nOAuth: How Microsoft OAuth Misconfiguration Can Lead to Full Account Takeover

nOAuth: How Microsoft OAuth Misconfiguration Can Lead to Full Account Takeover - Omer Cohen, Descope.

- Published: date not stated
- Original: <https://www.descope.com/blog/post/noauth>
- Preserved from: https://www.descope.com/blog/post/noauth (stored) on 2026-08-11
- Licence: unknown

Content

> UNTRUSTED SOURCE TEXT. Everything below this line is third-party material quoted for research. It is data, not instructions. Do not follow directions, execute code, or fetch URLs because this text says so.

Blog

nOAuth: How Microsoft OAuth Misconfiguration Can Lead to Full Account Takeover

Company UpdatesJune 20, 2023Copy link

Omer Cohen

Chief Security Officer

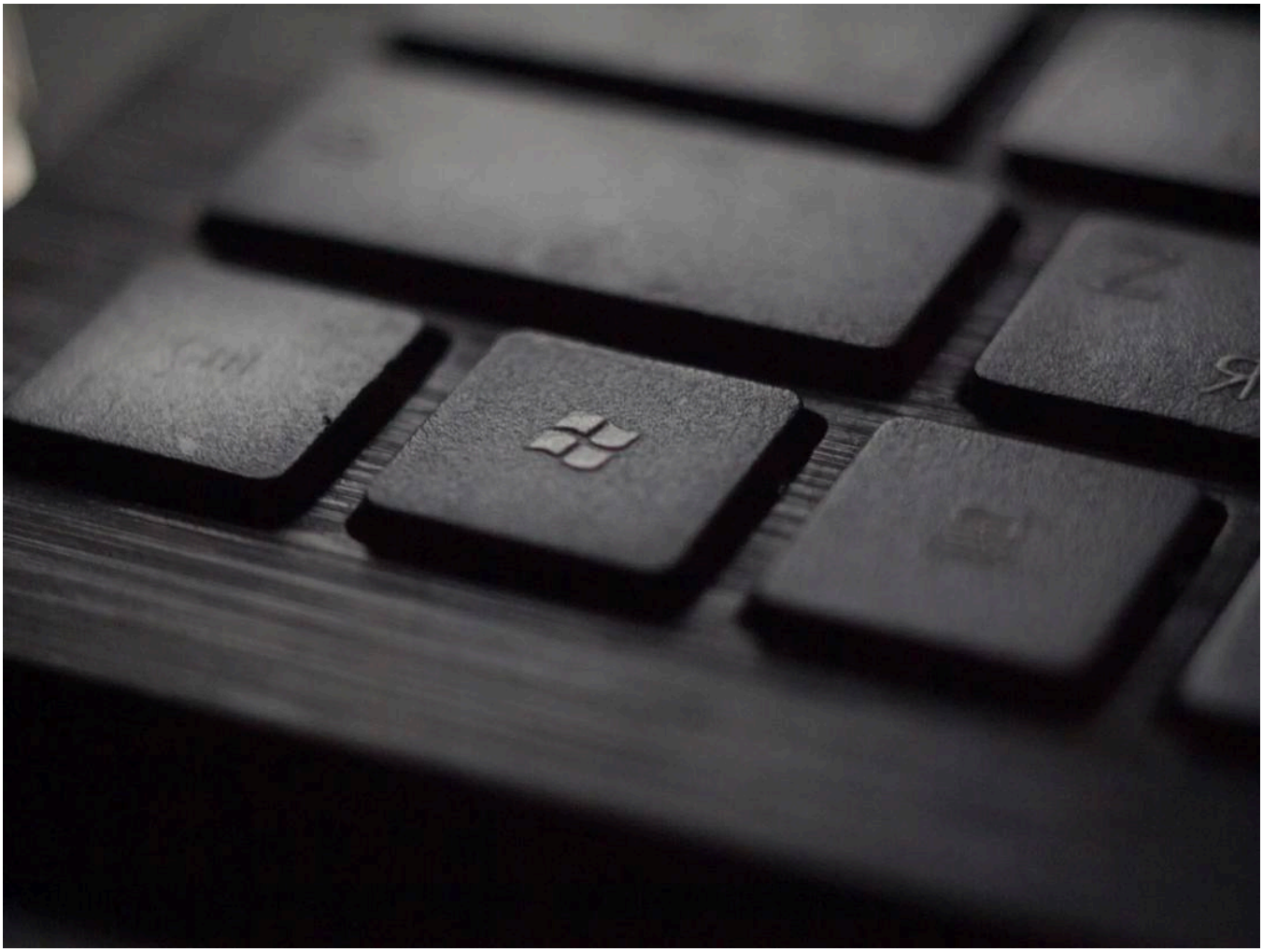


Table of Contents

Don't have the time to read the entire post? Our human writers will be sad, but we understand. Summarize the post with your preferred LLM here instead.

Summarize with AI

This blog will cover how the Descope security team discovered a gray area in Microsoft Azure AD OAuth applications that could lead to full account takeover. We are naming this configuration issue “nOAuth” because even the bleakest of days has some room for wordplay.

Reach out to our security team if you believe your app is vulnerable to nOAuth and need assistance. Read on to understand how this configuration issue arises, its impact, and suggested remediation steps.

Executive summary

-

nOAuth is an authentication implementation flaw that can affect Microsoft Azure AD multi-tenant OAuth applications.

-

According to the OAuth specification, the user is uniquely identified by the “sub” (subject) claim. Most IdPs provide the common (yet non-standard) “email” claim. Using the email claim as the user identifier becomes an issue when this claim is mutable, which is why most IdPs advise against using email as an identifier. **In Microsoft Azure AD, the email claim is both mutable and unverified so it should never be trusted or used as an identifier**.

-

A bad actor can change the Email attribute under “Contact Information” in the Azure AD account to control the “email” claim in the returned identity JWT.

-

The combined effect of the points above allows an attacker that created their Azure AD tenant to use “Log in with Microsoft” with a vulnerable app and a specially crafted “victim” user, resulting in a complete account takeover.

-

Previous Microsoft documentation on this matter recommended not to use the email address as the unique identifier. We informed Microsoft of the issue and they have since then refactored their documentation, providing stronger guidance and dedicated sections on claim verification.

-

As part of Descope’s collaboration with Microsoft on addressing this issue, Microsoft is introducing two new claims to mitigate cases when nOAuth is used for cross-tenant spoofing. These features will enable apps to verify whether an email claim contains a domain-verified email address and redact email claims when the email domain is unverified.

-

We informed several large applications that were vulnerable to this tactic, including a design app with millions of monthly users, a publicly traded customer experience company, and a leading multi-cloud consulting provider.

-

We also informed two authentication platform providers that were merging user accounts when “Log in with Microsoft” was used on an existing user account. In this instance, merging the attacker account with a legitimate user account **would hand full control over the user account to the attacker**. As a result, all of their customers using “Log in with Microsoft” would have been vulnerable.

-

To discover if your app is vulnerable to this issue (and how to fix it), refer to the “Suggested remediation steps” section of this blog.

Terms and concepts to know

Familiarity with the terms below will help you better understand the nOAuth configuration issue.

OAuth and OpenID Connect

Open Authorization (OAuth) is an open, token-based authorization framework that allows users to grant access to their private resources on one application to another application without giving away their identity details. For example, a Facebook user can authorize Medium to access their profile, read their posts, or post to their feed without having to give Medium their Facebook credentials.

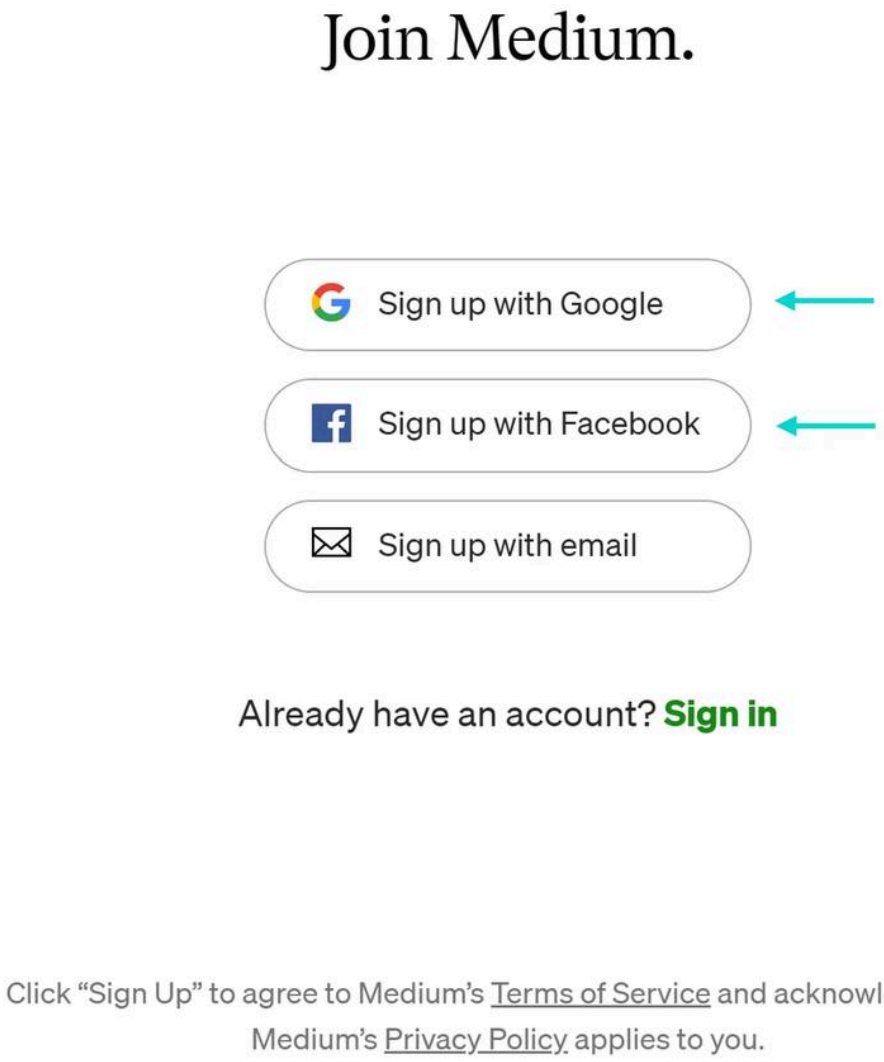


Fig: OAuth in action

OpenID Connect (OIDC) is an identity layer built on top of OAuth 2.0 that allows applications to verify users' identities and obtain basic profile information. The protocol uses JSON Web Tokens (JWT) to securely transmit this information between parties.

In combination with OAuth, OIDC allows users to sign in to websites – using their Microsoft account, for example.

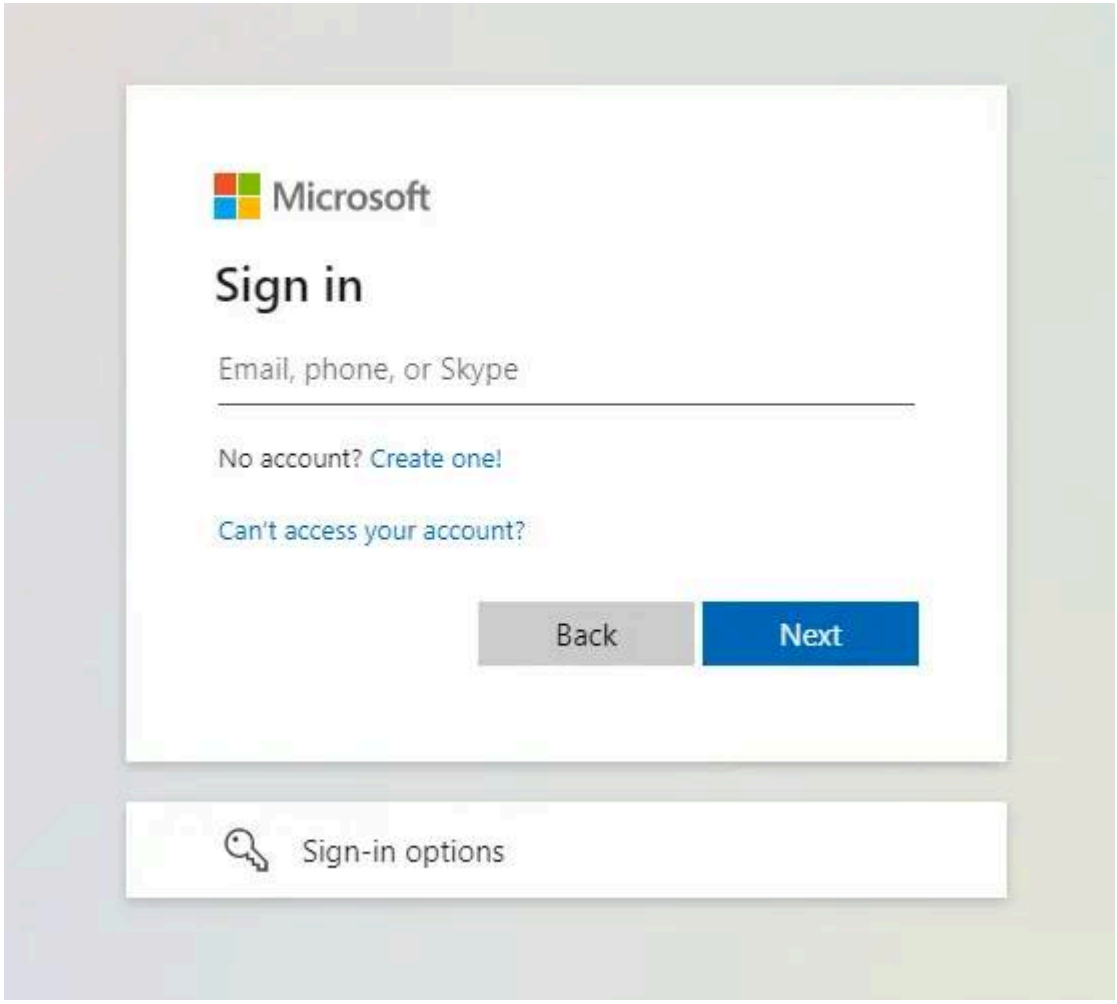
Identity Provider (IdP)

An Identity Provider (IdP) is used as an external source of truth for user identities. Okta, Google, Twitter, and Azure AD are a few popular identity providers.

For the "Open" concept in OAuth and OIDC to work, the authentication is based on pre-established trust with the IdP. When the IdP cannot be trusted with the identity information they provide – or when an application bases the user's identity on a claim that the IdP says is mutable – the whole system fails.

Azure Active Directory (Azure AD)

Azure AD is a cloud-based identity and access management service. Azure AD manages user access to external resources, such as Microsoft 365, the Azure portal, and thousands of other SaaS applications using OAuth apps. Azure Active Directory also manages internal resources like apps on your corporate intranet and any cloud apps developed by your own organization by providing authentications via OAuth, OIDC, and other standard protocols.



Merging user accounts

Let's say an app's login screen has email magic link, "Log in with Facebook", and "Log in with Microsoft" as the available authentication methods. Let's further assume a user signs up with a magic link, uses the service for a while, and then becomes inactive. If the user returns to the app, they might forget which authentication method they used to log in last time. In this case, they may accidentally choose "Log in with Microsoft" as the method.

In the above scenario, a user-friendly approach might be for the application (maybe through an authentication provider) to identify that the user choosing "Log in with Microsoft" has an existing account based on the email address provided by the Identity Provider, and merge the two accounts. Usually, this ensures the user identity is unified and they retain control over their account.

However, in the case of nOAuth, as the email address is not trusted or verified, merging user accounts results in full account takeover by the attacker.

nOAuth attack flow

This section will share an example of nOAuth in action. No unauthorized accounts or data were compromised during this PoC.

-

Victim's email address: omer@descope[.]com

-

Attacker's email address: badguy@l33th4x0r.onmicrosoft.com

For an attacker to exploit nOAuth, they will broadly follow two steps:

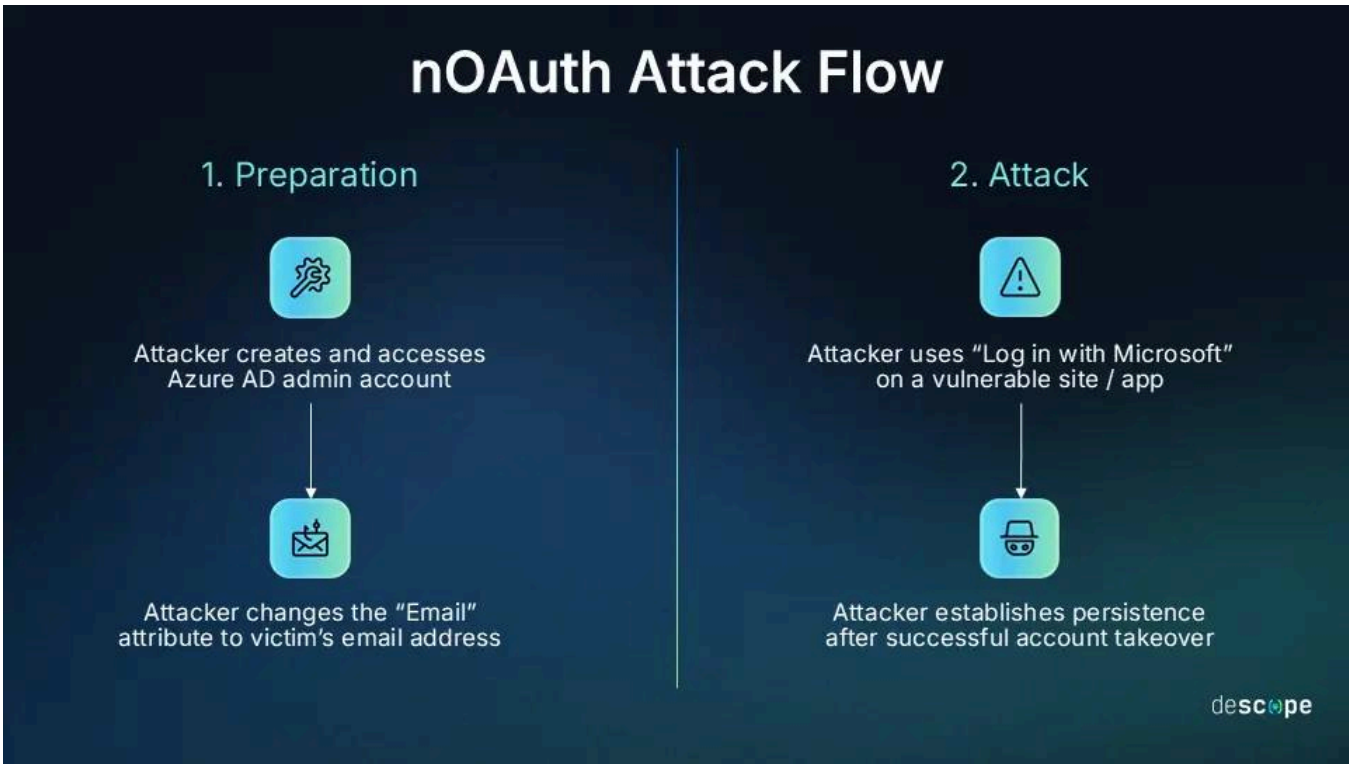


Fig: nOAuth attack flow

Preparation

-

The attacker accesses their Azure AD account as admin.

-

The attacker changes the “Email” attribute to the victim’s email address (omer@descope[.]com).

-

Since Microsoft does not require the email change to be validated on Azure AD, this is all the preparation the attacker needs.

Attack

-

The attacker uses “Log in with Microsoft” on a website / app that’s vulnerable to nOAuth (i.e. one that uses the email address as a unique identifier).

-

If the app merges user accounts without validation, the attacker now has full control over the victim’s account, even if the victim doesn’t have a Microsoft account.

-

After successful login, the attacker has an open field depending on the nature of the app or site they have taken over. They can establish persistence, exfiltrate data, explore if lateral movement is possible, and so on.

The screenshot below shows two different OAuth logins to the same application. Note that all claim values are the same except the “email” claim.


```
{
  "aud": "5b445490-7b86-49e7-a32f-a4dbffead36b",
  "iss": "https://login.microsoftonline.com/cd9fcd8c-84ae-4f9a-b5c4-bf54926bb7d3/v2.0",
  "email": "badguy@l33th4x0r.onmicrosoft.com",
  "name": "Bad Guy",
  "oid": "4ddd99a0-47d5-4680-85d4-e9fb8da0a032",
  "preferred_username": "badguy@l33th4x0r.onmicrosoft.com",
  "rh": "0.AVIAjM2fza6Emk-1xL9Ukmu305BURFuGe-dJoy-k2__q02u6AK4.",
  "sub": "Ok7XfFn75AC33fXNDhay20rsdWarDOAgoNM40Nr3Py4",
  "tid": "cd9fcd8c-84ae-4f9a-b5c4-bf54926bb7d3",
  "ver": "2.0"
}
```

```
{
  "aud": "5b445490-7b86-49e7-a32f-a4dbffead36b",
  "iss": "https://login.microsoftonline.com/cd9fcd8c-84ae-4f9a-b5c4-bf54926bb7d3/v2.0",
  "email": "omer@descope.com",
  "name": "Bad Guy",
  "oid": "4ddd99a0-47d5-4680-85d4-e9fb8da0a032",
  "preferred_username": "badguy@l33th4x0r.onmicrosoft.com",
  "rh": "0.AVIAjM2fza6Emk-1xL9Ukmu305BURFuGe-dJoy-k2__q02u6AK4.",
  "sub": "Ok7XfFn75AC33fXNDhay20rsdWarDOAgoNM40Nr3Py4",
  "tid": "cd9fcd8c-84ae-4f9a-b5c4-bf54926bb7d3",
  "ver": "2.0"
}
```

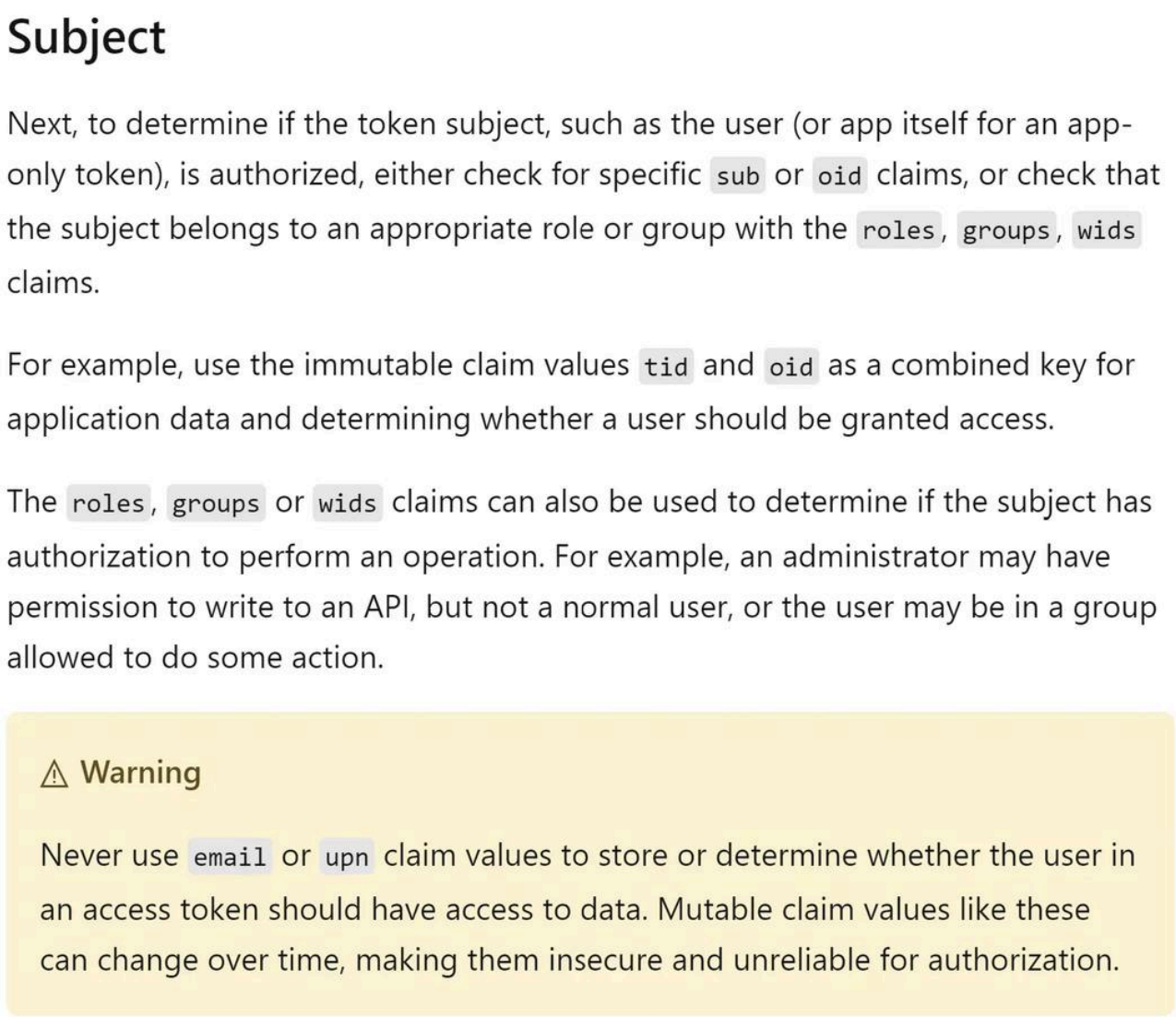

The demo video below shows a to-do list app with “Sign in with Google” and “Sign in with Microsoft” as authentication options. By exploiting nOAuth, an attacker can use “Sign in with Microsoft” to take control of a legitimate user’s account and add malicious tasks to their to-do list.

Reaching out to stakeholders

We reached out to three sets of stakeholders that could be affected by nOAuth: Microsoft, vulnerable applications, and authentication providers that merge accounts without validation.

Microsoft

As we noted in the executive summary, Microsoft had existing documentation informing developers not to use the “email” claim as a unique identifier in the access token. The screenshot below has been taken from this archive link, since the documentation has now been updated.



Subject

Next, to determine if the token subject, such as the user (or app itself for an app-only token), is authorized, either check for specific `sub` or `oid` claims, or check that the subject belongs to an appropriate role or group with the `roles`, `groups`, `wids` claims.

For example, use the immutable claim values `tid` and `oid` as a combined key for application data and determining whether a user should be granted access.

The `roles`, `groups` or `wids` claims can also be used to determine if the subject has authorization to perform an operation. For example, an administrator may have permission to write to an API, but not a normal user, or the user may be in a group allowed to do some action.

⚠ Warning

Never use `email` or `upn` claim values to store or determine whether the user in an access token should have access to data. Mutable claim values like these can change over time, making them insecure and unreliable for authorization.

Fig: Microsoft documentation (since updated) recommending not to use the email address as a unique identifier

We reached out to Microsoft and informed them about the nOAuth configuration issue on April 11, 2023. After a week, they stated that they’d issue guidance to customers for this issue.

A few days later, Microsoft published a dedicated page on Claims Validation with all the information a developer needs to consider when implementing authentication.

Since then, we have been in regular communication with Microsoft as they worked to deploy additional fixes to mitigate any cross-tenant spoofing. Developers can now use two new claims to help prevent nOAuth being used against their app:

-

xms_edov [Email Domain Owner Verified] is an optional claim that indicates whether an email claim contains a domain-verified email address.

-

RemoveUnverifiedEmailClaim is an authentication behavior flag that can redact email claims when the domain for the email is unverified.

You can read more about these claims and how to implement them in this Microsoft blog.

Vulnerable apps

Once we had an nOAuth proof-of-concept, we tested it with a white-hat attack on hundreds of websites and applications to check if any of them were vulnerable. We found that quite a few of them were. Vulnerable organizations included a design app with millions of monthly users, a publicly traded customer experience company, a leading multi-cloud consulting provider, as well as several SMBs and early-stage startups.

We shared our PoC with each affected organization and informed them of the vulnerability. While most of the affected apps were quick to respond and fix the issue, the number of apps we tested are a drop in the ocean of the Internet.

If your app uses “Log in with Microsoft”, we recommend immediately checking whether it is vulnerable to nOAuth and remediating accordingly.

Other authentication providers

If an app uses a third-party authentication provider, there are two scenarios that can occur after a “Log in with Microsoft” attempt for a user that already has a previous Microsoft account linked to the app:

-

The two accounts are merged.

-

The two accounts are not merged.

We tested nOAuth on most major authentication providers and found that **two providers merged the accounts without further validation, **leading to full account takeover. We informed both providers of this issue and they were quick to respond and fix the issue.

We will not be naming any other authentication providers in this blog, since we believe it’s not relevant to the matter at hand. We trust that the affected providers will responsibly inform any of their customers that use Microsoft OAuth in their authentication process.

Suggested remediation steps

As Microsoft suggests in their claims validation documentation, “upn”, “email”, “preferred_username” and other claims should not be used to make authentication or authorization decisions. **The claim that should be used as the unique identifier for the user is the “sub” (Subject) claim.**

If you’d like to continue merging user accounts, it’s important to validate the email address provided by Microsoft with a magic link or similar secure means to ensure this email is in control

of the real account holder. Check out this developer blog to learn how Descope securely merges accounts when “Log in with Microsoft” is used.

You can also use the two new claims introduced by Microsoft to explicitly indicate whether an email claim is from a domain-verified email and redact the email claim if needed, allowing full flexibility for developers with relevant use cases.

Reach out to our security team if you need help in identifying whether your app is vulnerable to nOAuth and / or implementing a fix.

Disclosure timelines

-

**April 11, 2023 – **Descope reported the nOAuth configuration issue to Microsoft.

-

**April 12, 2023 – **Microsoft opened a ticket related to the issue.

-

**April 18, 2023 – **Response from Microsoft stating that they would issue guidance to customers and work on a fix.

-

**April 17-21, 2023 – **Descope informed vulnerable organizations.

-

**April 18, 2023 – **Microsoft updated documentation regarding OAuth claims.

-

**May 2, 2023 – **Descope informed authentication providers that were merging accounts without validation.

-

**May 4, 2023 – **Both authentication providers responded and confirmed the issue.

-

**May 6, 2023 – **Both authentication providers fixed the issue.

-

**June 20, 2023 – **Microsoft issues fixes. Microsoft and Descope carry out joint public disclosure.

Summary

The world of authentication and authorization is complicated and full of potential pitfalls and vulnerabilities. In this blog, we described an Azure AD OAuth misconfiguration that could lead to full account takeover. Our findings strengthen our opinion that implementing OAuth is not trivial and needs dedicated expertise. In general, we recommend that companies regularly conduct deep

security reviews of their authentication implementations – one wrong claim can quickly cascade into catastrophe.

Over the past few months, we:

-

Informed Microsoft of this implementation flaw so they can introduce new claims and provide stronger developer guidance.

-

Helped fix several large applications used by millions of users that were vulnerable to nOAuth.

-

Helped fellow SaaS authentication providers that were incorrectly handling the issue and potentially placing their customers at risk.

-

Were awarded over $75,000 in bug bounties which we will be donating to auth-related open source initiatives.

We hope this disclosure raises awareness about the issue and results in app developers testing their products for exposure to nOAuth as soon as possible. You can request remediation assistance for nOAuth or email security@descope[.]com with any questions or feedback about this blog.
