New Phishing Attacks Exploiting OAuth Authorization Flows

August 7, 2021

Jenko Hwong
jhwong@netskope.com
@jenkohwong
$ az ad signed-in-user show

[
{
   "jobTitle": "Researcher",
   "department": "Threat Research Labs",
   "company": "Netskope, Inc.",
   "email": "jhwong@netskope.com",
   "twitter": "@jenkohwong",

   "background": "vulnerability scanning, AV/AS, pen-testing/exploits, L3/4 appliances, threat intel, windows security"
}
]
Phishing Evolution: smtp, fake domain, ssl cert, user/pwd

in the beginning...
Phishing Evolution: **apps**, fake domain, ssl cert, user/pwd

+mobile

1. attacker
   - phish
   - smtp, *sms, IM, chat*
   - steal username password

2. victim
   - browse, auth
   - http(s)

3. fake website
Phishing Evolution: apps, **fake domain, ssl cert, user/pwd**

+cloud

1. Phish
   - **smtp, sms, IM, chat...**

2. Browse, auth
   - **http(s)**

3. Steal username, password

**fake website hosted in cloud**
Phishing Evolution: apps, fake domain, ssl cert, user/pwd + cloud
Phishing Evolution: fake domain, apps, ssl cert, user/pwd

controls

1. phish

2. victim

3. steal username password

 attacker

fake website

smtp, sms, IM, chat...

http(s)

link analysis (domain/URLs/certs)

sender reputation

MFA

IP allow policies

content inspection (creds)
Phishing Evolution: OAuth 2.0 auth code grant

+cloud app authorization

1. Request Authorization
   - Request permissions (scopes)
   - Redirect user to Identity Platform (authorization service)

2. Authenticate and Authorize
   - Authenticate (MFA)
   - Authorize permissions (scopes)

3. OAuth Tokens
   - access token
   - refresh token

Identity Platform

Application (client, device)

User

Phishing Evolution: OAuth 2.0 auth code grant
+cloud app authorization: Payments

1 Item
Cart Subtotal

$229.99

CHECKOUT NOW

REP Sabre Olympic Bar - 20 kg
$229.99

Qty: 1 Remove

 VIEW AND EDIT CART

PAYMENT METHOD

- Credit Card
- PayPal

Pay with PayPal

+ Apply Discount Code
+ Apply Gift Card
Phishing Evolution: OAuth 2.0 auth code grant

+cloud app authorization: Payments
Phishing Evolution: OAuth 2.0 auth code grant

+cloud app authorization: Payments
Phishing Evolution: OAuth 2.0 auth code grant

+cloud app authorization: GCP CLI

$ gcloud auth login joeblogs@centeneo.com --launch-browser --force

Your browser has been opened to visit:

Phishing Evolution: OAuth 2.0 auth code grant

+cloud app authorization: GCP CLI
Phishing Evolution: OAuth 2.0 auth grant

Google Cloud SDK wants to access your Google Account

This will allow Google Cloud SDK to:

- See, edit, configure, and delete your Google Cloud Platform data
- View and manage your Google Compute Engine resources
- View and manage your applications deployed on Google App Engine

Make sure you trust Google Cloud SDK

You may be sharing sensitive info with this site or app. You can always see or remove access in your Google Account.

Learn how Google helps you share data safely.
See Google Cloud SDK's Privacy Policy and Terms of Service.

Cancel  Allow
Phishing Evolution

+cloud app authorization: GCP CLI
Phishing Evolution: OAuth 2.0 auth code grant

$ gcloud auth login joeblogs@centeneo.com --launch-browser --force

Your browser has been opened to visit:


You are now logged in as [joeblogs@centeneo.com].

$
Phishing Evolution: fake OAuth login
+cloud app authorization
Phishing Evolution: fake OAuth login, check creds

+cloud app authorization

- Real-time creds validation (APIs)[1]
- Based on pass/fail, redirect user to valid domains (stealth, creds validation upfront)

Phishing Evolution: fake OAuth login, check creds

+cloud app authorization

- Real-time creds validation (APIs)\[^1\]

- Controls
  - MFA, IP allow policies
  - link analysis (domain/URLs/certs)
  - content inspection (creds)
  - sender reputation

Phishing Evolution: OAuth 2.0 auth code grant

+cloud app authorization protocol -- why do we care?

1. Hijack session tokens, not creds
2. REST APIs <-> remote exploit vs endpoint
3. Authenticate and Authorize
   GET https://accounts.google.com/o/oauth2/v2/auth?
   client_id=32555940559.apps.googleusercontent.com&
   response_type=code&
   scope=https://www.googleapis.com/auth/cloud-platform&
   access_type=offline&redirect_uri=www.myapp.com:9000
   (authenticate, MFA, consent to scopes)

Application (client, device)

User

Request oauth tokens
POST https://www.googleapis.com/oauth2/v4/token
client_id=32555940559.apps.googleusercontent.com&
scope=https://www.googleapis.com/auth/cloud-platform&
client_secret=JqQXA298PB...&
code=AwABAAAAvPM1KaP...&
redirect_uri=www.myapp.com:9000

Login / Checkout / Install App

Redirect to Identity Platform

Redirect URL with Authorization Code
GET http://www.myapp.com:9000?
code=AwABAAAAvPM1KaP...

Identity Platform

OAuth tokens
{ "access_token": "ya29.a0ARrdaM9...",
"refresh_token": "1//06S3lSKyEHY...",
"scope": "https://www.googleapis.com...",
"expires_in": 3599,
"token_type": "Bearer"
}

1. Application (client, device)
2. Redirect to Identity Platform
3. Authenticate and Authorize
4. Redirect URL with Authorization Code
5. Request oauth tokens
6. Oauth tokens
Phishing Evolution: OAuth 2.0 illicit consent grants

1. Malicious registered application
2. Get user consent for wide scopes / permissions

OAuth tokens
{ "access_token": "ya29.a0ARrdaM9...",
"refresh_token": "1//06S3lSKyEHY...",
"scope": "https://www.googleapis...",
"expires_in": 3599,
"token_type": "Bearer"
}

+ cloud app authorization protocol

Identity Platform

1. Login / Checkout / Install App
2. Redirect to Identity Platform
3. Authenticate and Authorize
   
   GET https://accounts.google.com/o/oauth2/v2/auth?
   client_id=32555940559.apps.googleusercontent.com&
   response_type=code&
   scope=https://www.googleapis.com/auth/cloud-platform&
   access_type=offline&redirect_uri=www.myapp.com:9000

   (authenticate, MFA, consent to scopes)

4. Redirect URL with Authorization Code
   
   GET http://www.myapp.com:9000?
   code=AwABAAAAvPM1KaP...

5. Request oauth tokens
   
   POST https://www.googleapis.com/oauth2/v4/token
   client_id=32555940559.apps.googleusercontent.com&
   scope=https://www.googleapis.com/auth/cloud-platform&
   access_type=offline&redirect_uri=www.myapp.com:9000

6. Azure AD

User

Application (client, device)
Phishing Evolution: OAuth 2.0 illicit consent grants[1]

+cloud app authorization protocol


1. Malicious registered application
2. Get user consent for wide scopes / permissions

Controls
1. Prevent users from registering apps in AD
2. Prevent users from consenting
Phishing Evolution: OAuth 2.0 device code authorization[1]

What’s the purpose? To provide easier authentication/authorization on limited input devices e.g. smart TVs

“I think there's an RFC for that.”

From: draft-ietf-oauth-device-flow-15
Internet Engineering Task Force (IETF)
Request for Comments: 8628
Category: Standards Track
ISSN: 2070-1721

OAuth 2.0 Device Authorization Grant
which, when implemented, looks something like this on your TV.
with the real sign-in on a computer or mobile phone

Enter the code displayed on your TV.

If your device generates an activation code, you will need to enter that code on our website by doing the following:

1. Navigate to Netflix.com/activate.
2. After signing in, select the profile you would like to watch Netflix from.
3. Enter the code in the Enter code field. Click Activate.
Unusability is the father of insecurity
Instruct user to login on computer/smartphone

1. Go to www.google.com/device
2. Enter user code: ZLGG-LOSP

Phishing Evolution: OAuth 2.0 device code authorization

+cloud app authorization protocol

1. Login
2. Get user/device codes
3. Instruct user to login on computer/smartphone
   *1. Go to www.google.com/device
   2. Enter user code: ZLGG-LOSP*
4. Authenticate and Authorize
   1. Goes to www.google.com/device
   2. Enters: ZLGG-LOSP
   3. Authenticates, including MFA
5. Retrieve oauth tokens
   *client_id, device_code*

Demo: OAuth 2.0 device code authorization

- Dr. Nestori Syynimaa: https://o365blog.com/post/phishing/
- Usability => insecurity
- A different auth flow => opportunity
- Implementation quirks
Phishing Evolution: OAuth 2.0 device code authorization

+cloud app authorization protocol

Get user/device codes
POST
https://login.microsoftonline.com/common/oauth2/devicecode?api-version=1.0
client_id=d3590ed6-52b3-4102-aeff-aad2292ab01c
resource=https://outlook.office365.com

User/device codes
{ "device_code": "AH-1NgM6boio...", "verification_url": "https://www.google.com/device", "user_code": "ZLGG-LQSP", "expires_in": 1800, "interval": 5 }

Poll for oauth tokens
client_id
device_code

OAuth tokens
{ "access_token": "ya29.a0ARrdaM9...", "refresh_token": "1//06S3lSKyEHY...", }

Authenticate and Authorize
1. Goes to www.google.com/device
2. Enters: ZLGG-LOSP
3. Authenticates, including MFA

Manual instructions:
"1. Go to www.google.com/device
2. Enter: ZLGG-LOSP"

Device (client, app)

Identity Platform

Azure AD

Google Identity

User
Phishing Evolution: OAuth 2.0 device code authorization

+cloud app authorization protocol

Device (client, app)

User/device codes
POST
https://login.microsoftonline.com/common/oauth2/devicecode?api-version=1.0
client_id=d3590ed6-52b3-4102-aeff-aad2292ab01c&
resource=https://outlook.office365.com

Get user/device codes

User/device codes
{ "device_code": "AH-1NgM6boio...", "verification_url": "https://www.google.com/device", "user_code": "ZLGG-LQSP", "expires_in": 1800, "interval": 5 }

Poll for oauth tokens

Oauth tokens
{ "access_token": "ya29.a0ARrdaM9...", "refresh_token": "1//06S3lSKyEHY...", }

Phish

"Here's your promotional product code:
1. Go to www.google.com/device
2. Enter: ZLGG-LOSP"

Authenticate and Authorize

1. Goes to www.google.com/device
2. Enters: ZLGG-LOSP
3. Authenticates, including MFA

Login

1

Identity Platform

Azure AD

Google Identity

2

3

4

5

Phishing Evolution: OAuth 2.0 device code authorization

microsoft phish
Phishing Evolution: OAuth 2.0 device code authorization protocol

Access Token
{
  "scope": "user_impersonation",
  "resource": "https://management.azure.com",
  "access_token": "eyJ0eXAiOiJKV1QiLCJhbG...",
  "refresh_token": "0.AUYAknJ93kbWUyXs2..."
}

Device (client, app)

Identity Platform

Use refresh token to get new access token for Azure
{
  "refresh_token": "1/06S3iSKyEHY...",
  "scope": "openid",
  "grant_type": "refresh_token"
  "resource": "https://management.azure.com",
  "client_id": "d3590ed6-52b3-4102-aeff-aad2292ab01c",
}
Phishing Evolution: OAuth 2.0 device code authorization

**microsoft phish**

1. No server infrastructure
2. No registered application, use existing vendor client app
3. No consent screen

---

**Get user/device codes**

POST
https://login.microsoftonline.com/common/oauth2/devicecode?api-version=1.0
client_id=d3590ed6-52b3-4102-aeff-aad2292ab01c&
resource=https://outlook.office365.com

**Poll for oauth tokens**

user/device codes

```
{ "device_code": "AH-1NgM6boio...", "verification_url": "https://www.google.com/device", "user_code": "ZLGG-LOSP", "expires_in": 1800, "interval": 5 }
```

**Phish**

"Here’s your promotional product code:
1. Go to www.google.com/device
2. Enter: ZLGG-LOSP"

---

**Oauth tokens**

```
{ "access_token": "ya29.a0ARrdaM9...", "refresh_token": "1//06S3lSKyEHY...", }
```

---

**Authenticate and Authorize**

1. Goes to www.google.com/device
2. Enters: ZLGG-LOSP
3. Authenticates, including MFA
Phishing Evolution: OAuth 2.0 device code authorization

+cloud app authorization protocol

Get user/device codes
POST
https://login.microsoftonline.com/common/oauth2/devicecode?api-version=1.0
client_id=d3590ed6-52b3-4102-aeff-aad2292ab01c
resource=https://outlook.office365.com

User/device codes
{ "device_code": "AH-1NgM6boio...", "verification_url": "https://www.google.com/device", "user_code": "ZLGG-LOSP", "expires_in": 1800, "interval": 5 }

Poll for oauth tokens
client_id
device_code

Oauth tokens
{ "access_token": "ya29.a0ARrdaM9...", "refresh_token": "1/06S3ISkYEHY...", }

Phish
"Here's your promotional product code:
1. Go to www.google.com/device
2. Enter: ZLGG-LOSP"

Authenticate and Authorize
1. Goes to www.google.com/device
2. Enters: ZLGG-LOSP
3. Authenticates, including MFA

1. No server infrastructure
2. No registered application, use existing vendor client app
3. No consent screen
4. Implicit, default scopes

microsoft phish

Device
(client, app)

User
Phishing Evolution: OAuth 2.0 device code authorization protocol

+cloud app authorization protocol

1. No server infrastructure
2. No registered application, use existing vendor client app
3. No consent screen
4. Implicit, default scopes
5. Move laterally to other services
6. Logging limited (initial token logged as sign-in, but lateral move is not)
Phishing Evolution: OAuth 2.0

+ cloud app authorization protocol

1. No server infrastructure
2. No registered application, use existing vendor client app
3. No consent screen
4. Implicit, default scopes
5. Move laterally to other services
6. Logging limited (initial token logged as sign-in, but lateral move is not)
Phishing Evolution: OAuth 2.0 device code authorization

controls

1. Prevent: block verification URIs, use conditional access policies
   - https://oauth2.googleapis.com/device/code
   - https://microsoft.com/devicelogin
   - https://login.microsoftonline.com/common/oauth2/deviceauth
   - block access based on IP, location, endpoint characteristics

2. Detect
   - Difficult

3. Remediate
   - API to revoke all oauth tokens for a user

   1. No server infrastructure
   2. No registered application, use existing vendor client app
   3. No consent screen
   4. Implicit, default scopes
   5. Move laterally to other services
   6. Logging limited (initial token logged as sign-in, but lateral move is not)
Phishing Evolution: OAuth 2.0 **device code authorization**

**controls**

1. **Prevent:** block verification URIs, use conditional access policies
   - https://oauth2.googleapis.com/device/code
   - https://microsoft.com/devicelogin
   - https://login.microsoftonline.com/common/oauth2/deviceauth
   - block access based on IP, location, endpoint characteristics

2. **Detect**
   - https://login.microsoftonline.com/common/oauth2/deviceauth

3. **Remediate**
   - API to revoke all oauth tokens for a user

**practical considerations**

- Short expiration of user/device codes (15-30mins)
  - phishing numbers game
  - incorporate hosted website, generate codes dynamically
  - use images for user code (no javascript allowed in email clients)

**microsoft phish**

1. No server infrastructure
2. No registered application, use existing vendor client app
3. No consent screen
4. Implicit, default scopes
5. Move laterally to other services
6. Logging limited (initial token logged as sign-in, but lateral move is not)
<table>
<thead>
<tr>
<th></th>
<th>Microsoft</th>
<th>Google</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Server infrastructure</strong></td>
<td>None required</td>
<td>None required</td>
</tr>
<tr>
<td><strong>Application registration</strong></td>
<td>None needed, can use large # of existing apps</td>
<td>Some limited vendor apps e.g. Chrome</td>
</tr>
<tr>
<td><strong>Consent screens</strong></td>
<td>No</td>
<td>Partial (limited vendor apps)</td>
</tr>
<tr>
<td><strong>Scopes</strong></td>
<td>Implicit, default scopes, wide-range</td>
<td><strong>Very limited</strong> (user profile, drive access to app files, youtube info)</td>
</tr>
<tr>
<td><strong>Lateral movement</strong></td>
<td>Easy to switch among large number of services</td>
<td><strong>No</strong>: strict limited scopes for device code flow</td>
</tr>
<tr>
<td><strong>Logging</strong></td>
<td>Partial (initial token access)</td>
<td>Partial</td>
</tr>
<tr>
<td><strong>Prevention</strong></td>
<td>block URIs, cond access</td>
<td>block URIs, VPC perimeters</td>
</tr>
<tr>
<td><strong>Detection</strong></td>
<td>Difficult</td>
<td>Difficult</td>
</tr>
<tr>
<td><strong>Remediation</strong></td>
<td>API to revoke user tokens</td>
<td>Delete/recreate user</td>
</tr>
</tbody>
</table>
Ongoing Research Areas

- Other flows\(^1\)
- Any usability "requirements"
- Bypass consent e.g. implicit grants
- Default scopes\(^2\)
- Consent\(^3\)
- Browser auto-login and scope expansion e.g. Google uberauth (2013)\(^4\)[\(^5\)]

---

4. Obtaining Authorization

4.1. Authorization Code Grant

4.1.1. Authorization Request
4.1.2. Authorization Response
4.1.3. Access Token Request
4.1.4. Access Token Response

4.2. Implicit Grant

4.2.1. Authorization Request
4.2.2. Access Token Request

4.3. Resource Owner Password Credentials Grant

4.3.1. Authorization Request and Response
4.3.2. Access Token Request
4.3.3. Access Token Response

4.4. Client Credentials Grant

4.4.1. Authorization Request and Response
4.4.2. Access Token Request
4.4.3. Access Token Response

---

4.4. Obtaining an Access Token
4.5. Using an Access Token
4.6. Revoking an Access Token
4.7. Access Token Scopes

---

**Getting access tokens silently in the background**

This part of the implicit flow is unlikely to work for your application as it's used across different browsers due to the removal of third party cookies by default. While this still currently works in Chromium-based browsers that are not in Incognito, developers should reconsider using this part of the flow. In browsers that do not support third party cookies, you will receive an error indicating that no users are signed in, as the login page's session cookies were removed by the browser.

---

**Incremental and dynamic user consent**

With the Microsoft identity platform endpoint, you can ignore the static permissions defined in the app registration information in the Azure portal and request permissions incrementally instead. You can ask for a bare minimum set of permissions upfront and request more over time as the customer uses additional app features. To do so, you can specify the scopes your app needs at any time by including the new scopes in the **scope** parameter when requesting an access token - without the need to pre-define them in the application registration information. If the user hasn't yet consented to new scopes added to the request, they'll be prompted to consent only to the new permissions. Incremental, or dynamic consent, only applies to delegated permissions and not to application permissions.

---

\(^3\) [https://docs.microsoft.com/en-us/azure/active-directory/develop/v2-permissions-and-consent](https://docs.microsoft.com/en-us/azure/active-directory/develop/v2-permissions-and-consent)
\(^4\) [https://gist.github.com/arirubinstein/fd5453537436a8757266f908c3e41538](https://gist.github.com/arirubinstein/fd5453537436a8757266f908c3e41538)
\(^5\) [https://duo.com/blog/beyond-the-vulnerabilities-of-the-application-specific-password-exploiting-google-chrome-s-oauth2-tokens](https://duo.com/blog/beyond-the-vulnerabilities-of-the-application-specific-password-exploiting-google-chrome-s-oauth2-tokens)
Thank you

Questions

Open Source Tools

- Repo: https://github.com/netskopeoss/phish_oauth
- License: BSD-3-Clause

Contact

- jhwong@netskope.com
- @jenkohwong
References

1.0 Evolving Phishing Attacks

1.1 A Big Catch: Cloud Phishing from Google App Engine and Azure App Service:
https://www.netskope.com/blog/a-big-catch-cloud-phishing-from-google-app-engine-and-azure-app-service

1.2 Microsoft Seizes Malicious Domains Used in Mass Office 365 Attacks:

1.3 Phishing Attack Hijacks Office 365 Accounts Using OAuth Apps:

1.4 Office 365 Phishing Attack Leverages Real-Time Active Directory Validation:

1.5 Demonstration - Illicit Consent Grant Attack in Azure AD:
https://www.nixu.com/blog/demonstration-illicit-consent-grant-attack-azure-ad-office-365

1.6 Detection and Mitigation of Illicit Consent Grant Attacks in Azure AD:
https://www.cloud-architekt.net/detection-and-mitigation-consent-grant-attacks-azuread/

1.7 HelSec Azure AD write-up: Phishing on Steroids with Azure AD Consent Extractor:
https://securecloud.blog/2019/12/17/helsec-azure-ad-write-up-phishing-on-steroids-with-azure-ad-consent-extractor/

1.8 Pawn Storm Abuses OAuth In Social Engineering Attack:

2.0 OAuth Device Code Flow

2.1 OAuth 2.0 RFC: https://tools.ietf.org/html/rfc6749

2.2 OAuth 2.0 Device Authorization Grant RFC: https://datatracker.ietf.org/doc/html/rfc8628

2.3 OAuth 2.0 for TV and Limited-Input Device Applications:
https://developers.google.com/identity/protocols/oauth2/limited-input-device

2.4 OAuth 2.0 Scopes for Google APIs:
https://developers.google.com/identity/protocols/oauth2/scopes

2.5 Introducing a new phishing technique for compromising Office 365 accounts:
https://o365blog.com/post/phishing/#oauth-consent

2.6 Office Device Code Phishing:
https://gist.github.com/Mr-Un1k0d3r/afef5a80cb72dfeaa78d14465fb0d333

3.0 Additional OAuth Research Areas

3.1 Poor OAuth implementation leaves millions at risk of stolen data:

3.2 How did a full access OAuth token get issued to the Pokémon GO app?:
https://searchsecurity.techtarget.com/answer/How-did-a-full-access-OAuth-token-get-issued-to-the-Pokemon-GO-app