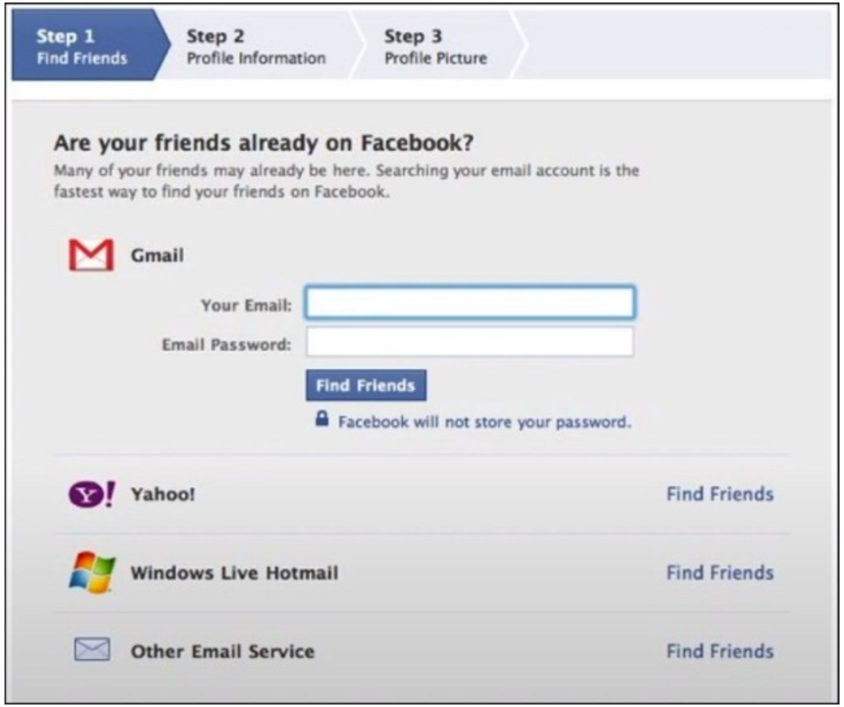
Keeping Pace with OAuth's Evolving Security Practices

Pieter Philippaerts
SECDES-meeting 18 October 2024

KU LEUVEN DistriNet


Step 1
Find Friends

Step 2
Profile Information

Step 3
Profile Picture

Are your friends already on Facebook?

Many of your friends may already be here. Searching your email account is the fastest way to find your friends on Facebook.

Gmail

Your Email:

Email Password:

Find Friends

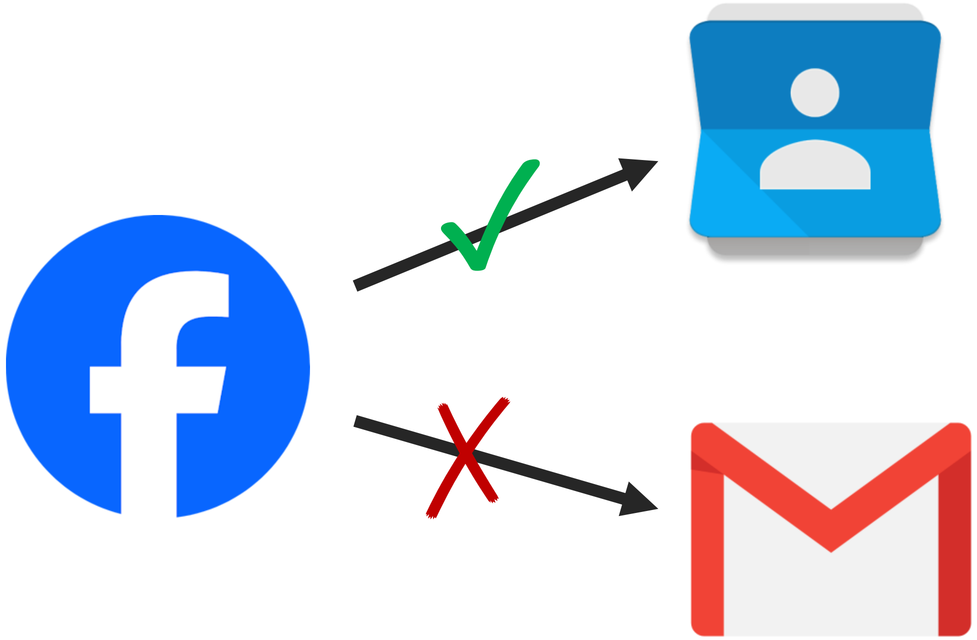
Facebook will not store your password.

Yahoo! Find Friends

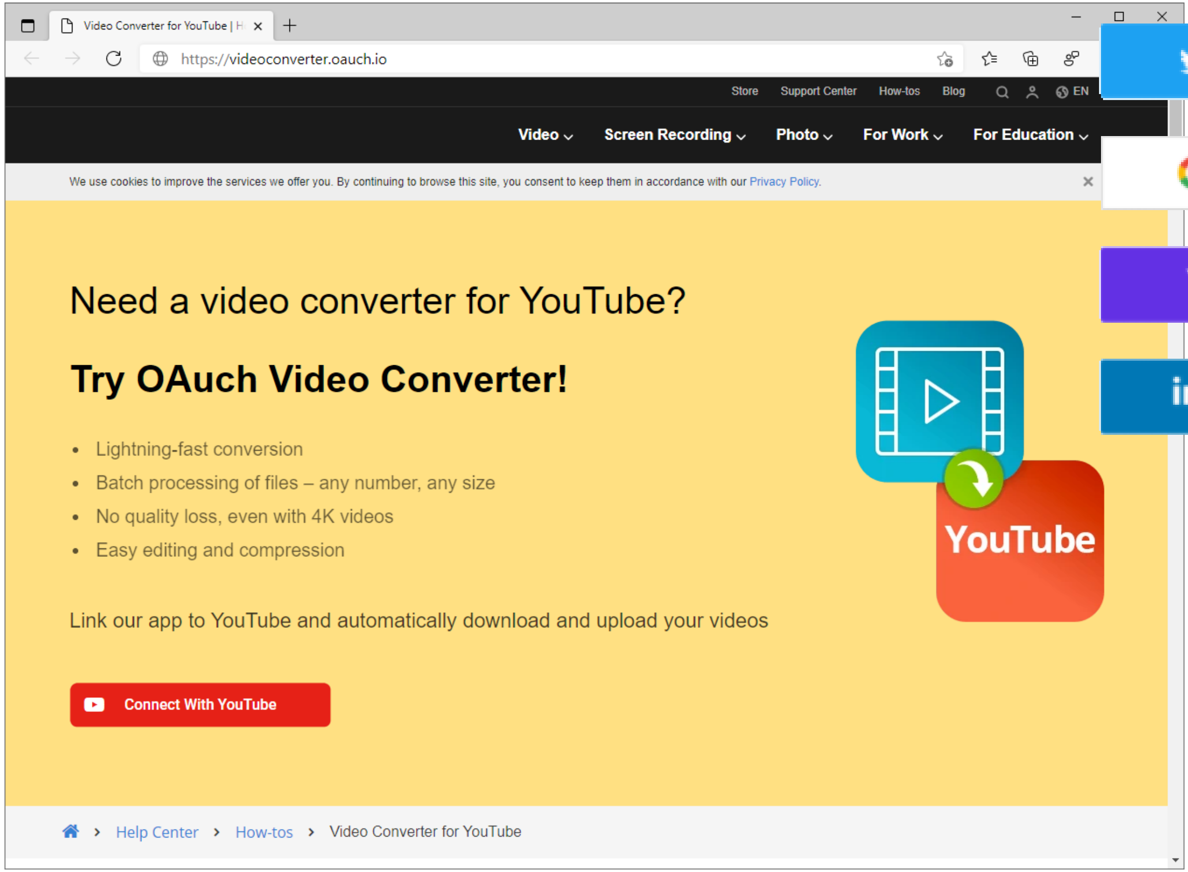
Windows Live Hotmail Find Friends

Other Email Service Find Friends

If a third party wanted access to an account, you'd give them your password



So…
how can I let an app
access my data
without giving it my password?



Video Converter for YouTube | H

https://videoconverter.oauch.io

We use cookies to improve the services we offer you. By continuing to browse this site, you consent to keep them in accordance with our Privacy Policy.

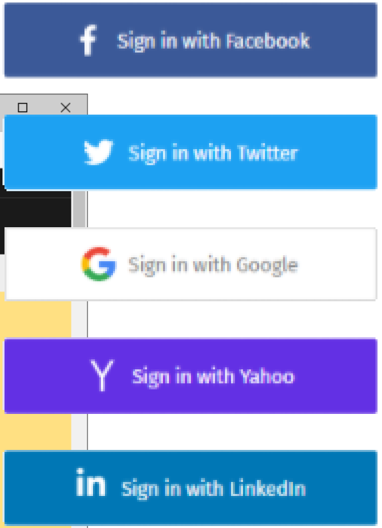
Need a video converter for YouTube?

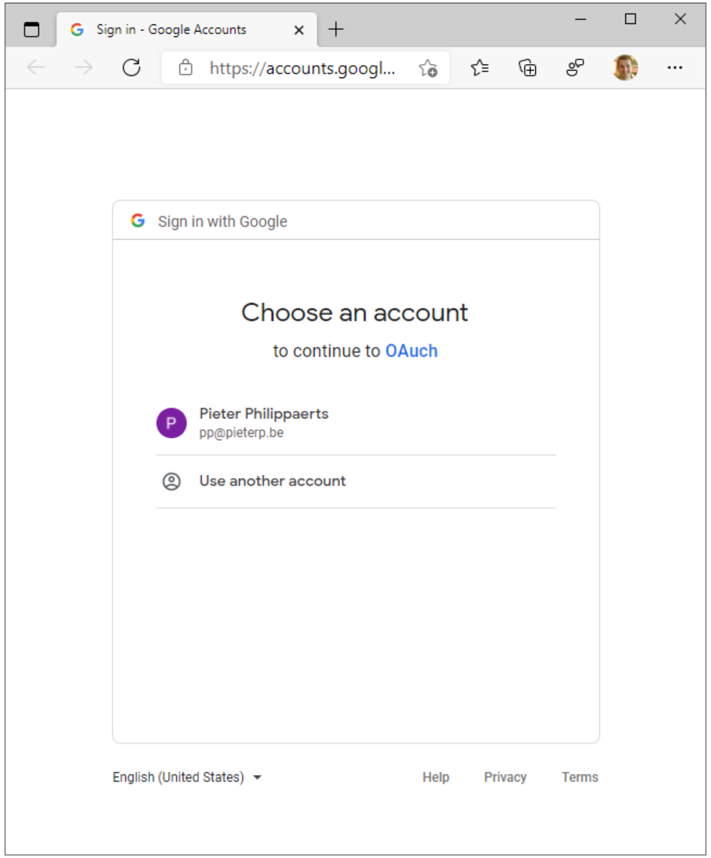
Try OAuch Video Converter!

- Lightning-fast conversion
- Batch processing of files – any number, any size
- No quality loss, even with 4K videos
- Easy editing and compression

Link our app to YouTube and automatically download and upload your videos

Help Center › How-tos › Video Converter for YouTube





Sign in - Google Accounts

https://accounts.googl...

Sign in with Google

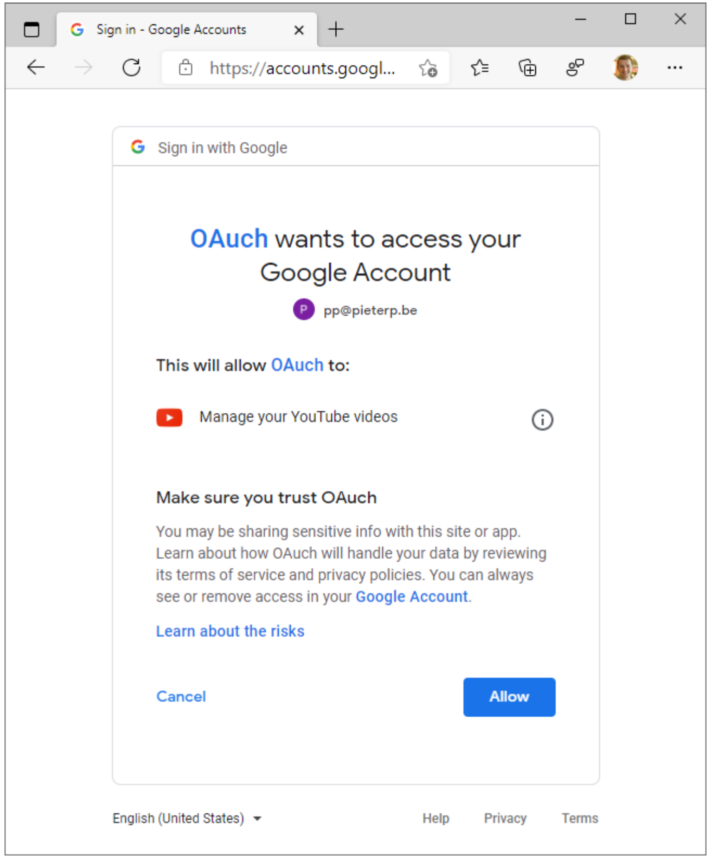
Choose an account

to continue to OAuch

Pieter Philippaerts
pp@pieterp.be

Use another account

English (United States) Help Privacy Terms



Sign in - Google Accounts

https://accounts.googl...

Sign in with Google

OAuch wants to access your Google Account

pp@pieterp.be

This will allow OAuch to:

Manage your YouTube videos

Make sure you trust OAuch

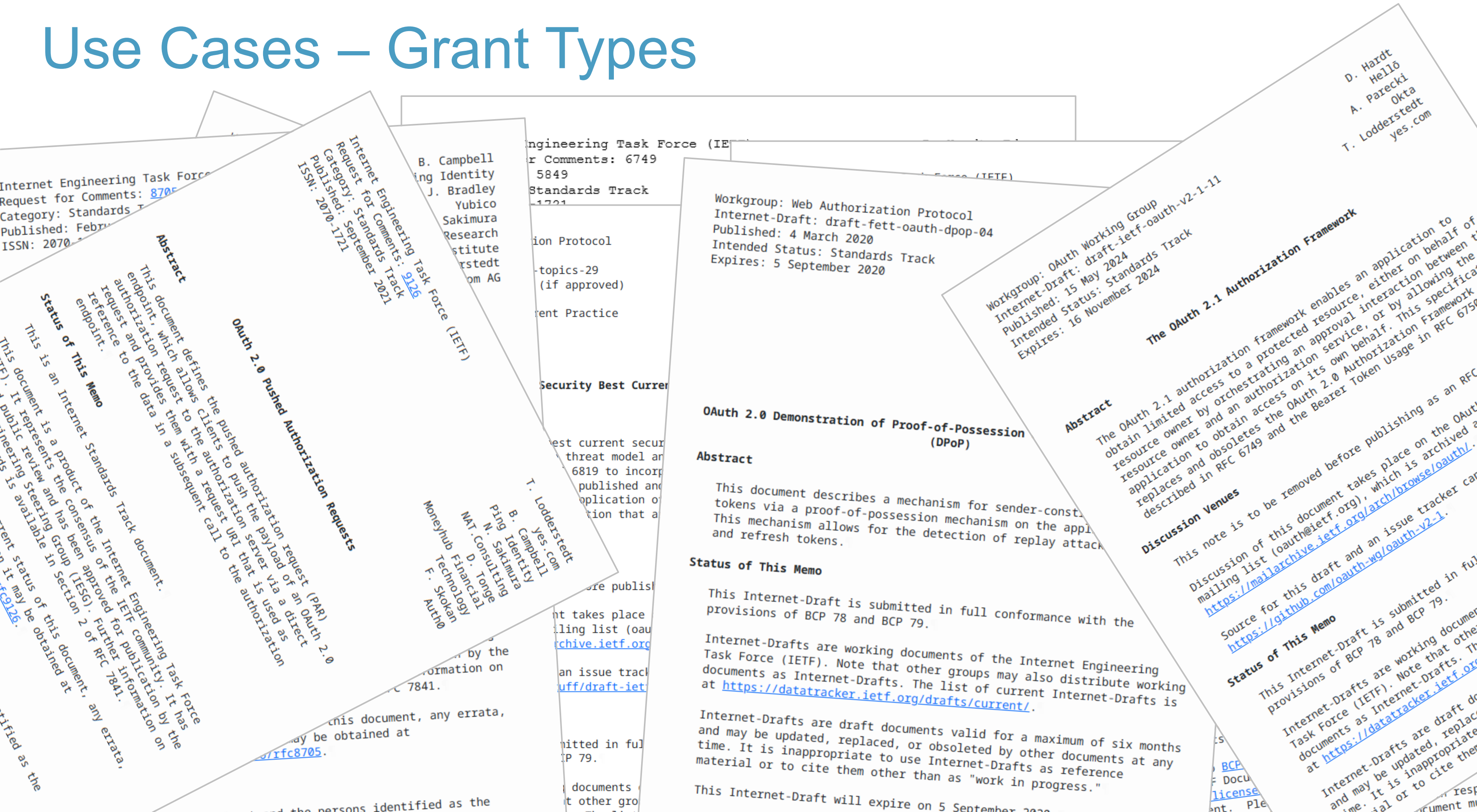
You may be sharing sensitive info with this site or app. Learn about how OAuch will handle your data by reviewing its terms of service and privacy policies. You can always see or remove access in your Google Account.

Learn about the risks

Cancel Allow

English (United States) Help Privacy Terms

Use Cases – Grant Types



Internet Engineering Task Force
Request for Comments: 8705
Category: Standards T
Published: Febru
ISSN: 2070-

by the
ormation on
7841.

this document, any errata,
ay be obtained at
/rfc8705.

the persons identified as the

Internet Engineering Task Force (IETF)
Request for Comments: 9126
Category: Standards Track
Published: September 2021
ISSN: 2070-1721

T. Lodderstedt
yes.com
B. Campbell
Ping Identity
N. Sakimura
NAT.Consulting
D. Tonge
Moneyhub Financial Technology
F. Skokan
Auth0

OAuth 2.0 Pushed Authorization Requests

Abstract

This document defines the pushed authorization request (PAR) endpoint, which allows clients to push the payload of an OAuth 2.0 authorization request to the authorization server via a direct request and provides them with a request URI that is used as reference to the data in a subsequent call to the authorization endpoint.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

rent status of this document, any errata, it may be obtained at rfc9126.

B. Campbell
ing Identity
J. Bradley
Yubico
Sakimura
Research
stitute
rstedt
om AG

ngineering Task Force (IETF)
r Comments: 6749
5849
Standards Track
-1721

ion Protocol
-topics-29
(if approved)
rent Practice

Security Best Curren

est current secur
threat model ar
6819 to incorp
published and
pplication of
tion that a

re publish
nt takes place
ling list (oau
rchive.ietf.org
an issue track
uff/draft-iet

itted in ful
P 79.

documents
t other gro

Workgroup: Web Authorization Protocol
Internet-Draft: draft-fett-oauth-dpop-04
Published: 4 March 2020
Intended Status: Standards Track
Expires: 5 September 2020

OAuth 2.0 Demonstration of Proof-of-Possession
(DPoP)

Abstract

This document describes a mechanism for sender-constr tokens via a proof-of-possession mechanism on the appl This mechanism allows for the detection of replay attack and refresh tokens.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 5 September 2020.

D. Hardt
Hellō
A. Parecki
Okta
T. Lodderstedt
yes.com

Workgroup: OAuth Working Group
Internet-Draft: draft-ietf-oauth-v2-1-11
Published: 15 May 2024
Intended Status: Standards Track
Expires: 16 November 2024

The OAuth 2.1 Authorization Framework

Abstract

The OAuth 2.1 authorization framework enables an application to obtain limited access to a protected resource, either on behalf of resource owner by orchestrating an approval interaction between the resource owner and an authorization service, or by allowing the application to obtain access on its own behalf. This specification replaces and obsoletes the OAuth 2.0 Authorization Framework described in RFC 6749 and the Bearer Token Usage in RFC 6750.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the OAuth mailing list (oauth@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/oauth/.

Source for this draft and an issue tracker can https://github.com/oauth-wg/oauth-v2-1.

Status of This Memo

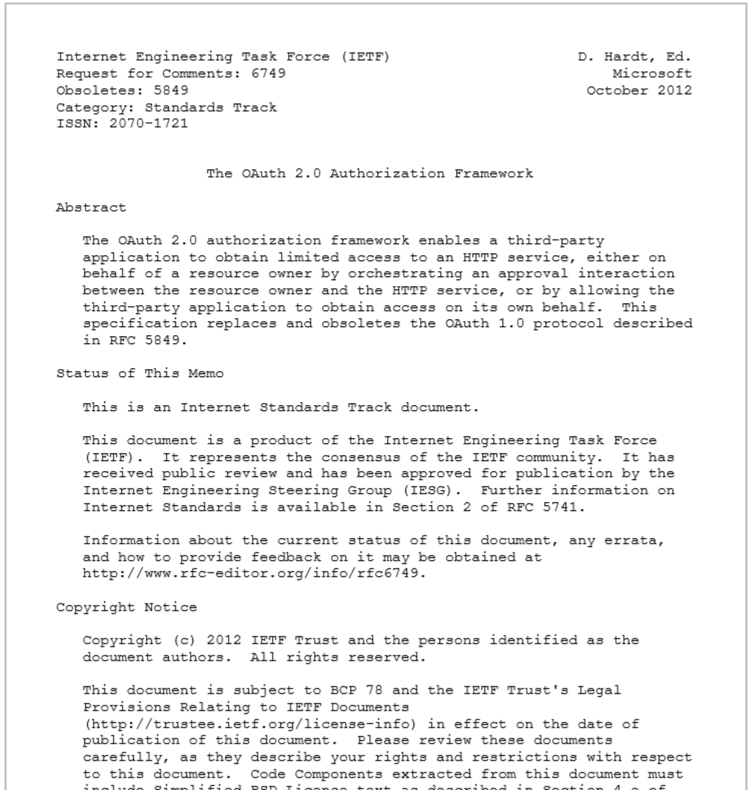
This Internet-Draft is submitted in full provisions of BCP 78 and BCP 79.

Internet-Drafts are working documen Task Force (IETF). Note that othe documents as Internet-Drafts. Th at https://datatracker.ietf.or

Internet-Drafts are draft d and may be updated, replac time. It is inappropriate material or to cite them

BCP
Docu
license
ent. Ple

Use Cases

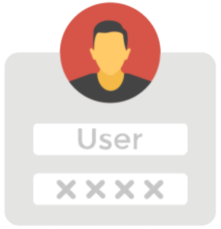


```
Internet Engineering Task Force (IETF)                    D. Hardt, Ed.
Request for Comments: 6749                                    Microsoft
Obsoletes: 5849                                            October 2012
Category: Standards Track
ISSN: 2070-1721


                 The OAuth 2.0 Authorization Framework

Abstract

   The OAuth 2.0 authorization framework enables a third-party
   application to obtain limited access to an HTTP service, either on
   behalf of a resource owner by orchestrating an approval interaction
   between the resource owner and the HTTP service, or by allowing the
   third-party application to obtain access on its own behalf.  This
   specification replaces and obsoletes the OAuth 1.0 protocol described
   in RFC 5849.

Status of This Memo

   This is an Internet Standards Track document.

   This document is a product of the Internet Engineering Task Force
   (IETF).  It represents the consensus of the IETF community.  It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG).  Further information on
   Internet Standards is available in Section 2 of RFC 5741.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   http://www.rfc-editor.org/info/rfc6749.

Copyright Notice

   Copyright (c) 2012 IETF Trust and the persons identified as the
   document authors.  All rights reserved.

   This document is subject to BCP 78 and the IETF Trust's Legal
   Provisions Relating to IETF Documents
   (http://trustee.ietf.org/license-info) in effect on the date of
   publication of this document.  Please review these documents
   carefully, as they describe your rights and restrictions with respect
   to this document.  Code Components extracted from this document must
```

Web-server apps

Browser-based apps

Mobile apps



Username/Password access

Application access

KU LEUVEN DistriNet

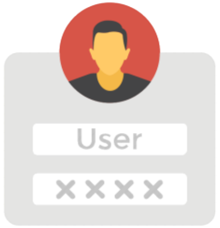
Use Cases – Grant Types

ANNO 2012

Web-server apps
authorization code

Browser-based apps
implicit

Mobile apps
implicit



Username/Password access
password

Application access
client credentials

KU LEUVEN DistriNet

OAuth 2.0 Roles

Resource Owner
“the user”

User-Agent
“the browser”

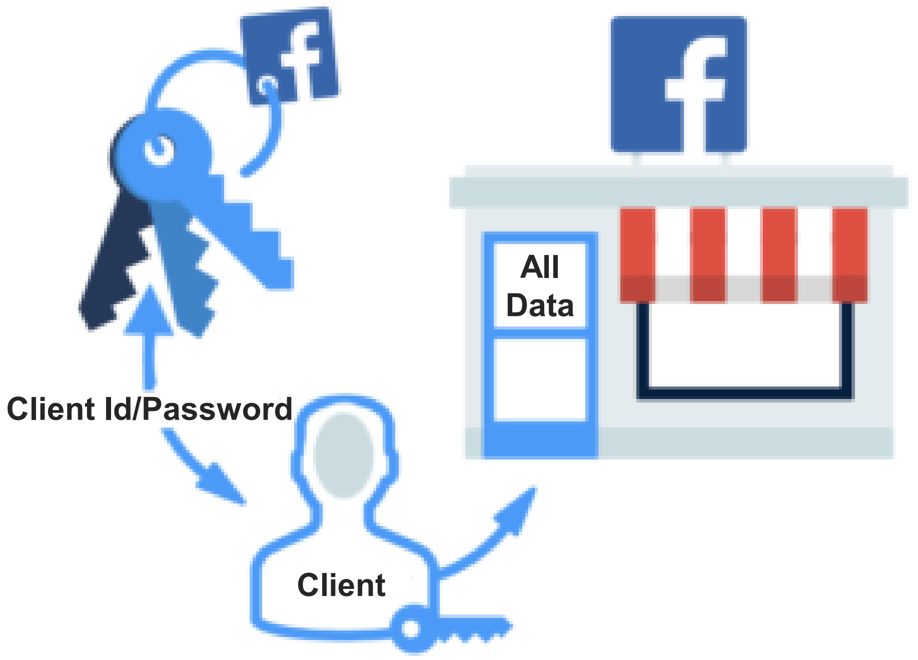
Client
“the app”

Resource Server
“the API”

Authorization Server

OAuth 2.0 Grant Types

Client Credentials Grant



https://blog.oauth.io/oauth2-flow-grant-types-in-pictures/

Client Credentials Grant

REQUEST

```
POST /token HTTP/1.1
Host: server.example.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials
```

Client ID & Password

RESPONSE

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
  "access_token":"2YotnFZFEjr1zCsicMWpAA",
  "token_type":"Bearer",
  "expires_in":3600
}
```

KU LEUVEN DistriNet

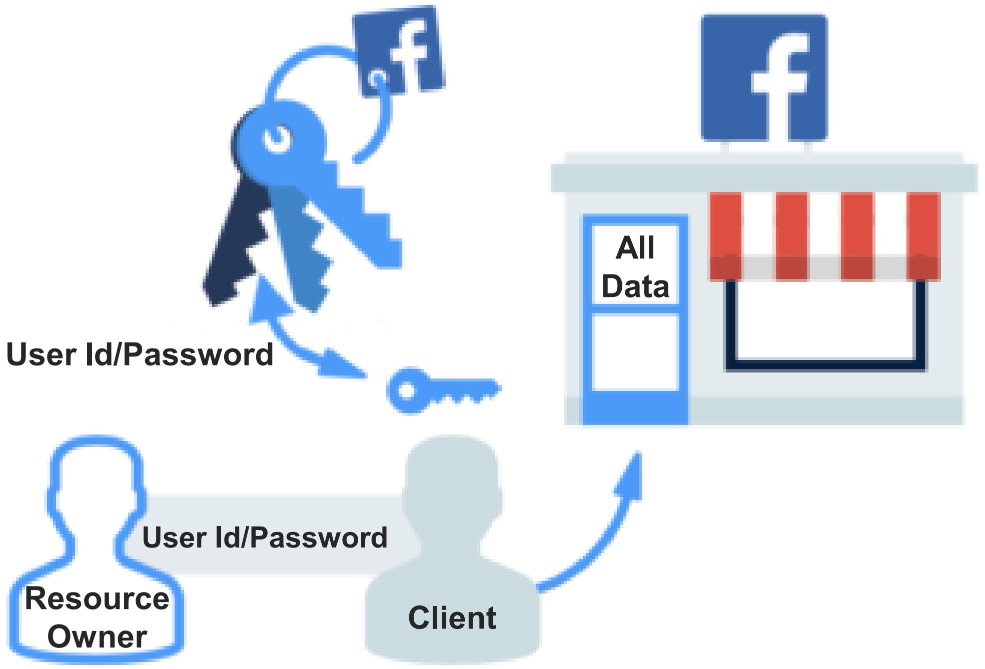
Client Credentials Grant

- Easy ✓
- Secure ✓
- Wide use case support ✗

BEST PRACTICE
BEST PRACTICE
BEST PRACTICE

Use the Client Credentials flow for Machine-to-machine authorization

Password Grant



https://blog.oauth.io/oauth2-flow-grant-types-in-pictures/

Password Grant

REQUEST

```
POST /token HTTP/1.1
Host: server.example.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=password&username=johndoe&password=A3ddj3w
```

Client ID & Password

Resource Owner Username & Password

RESPONSE

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
  "access_token":"2YotnFZFEjr1zCsicMWpAA",
  "token_type":"example",
  "expires_in":3600,
  "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA",
}
```

KU LEUVEN DistriNet

Password Grant

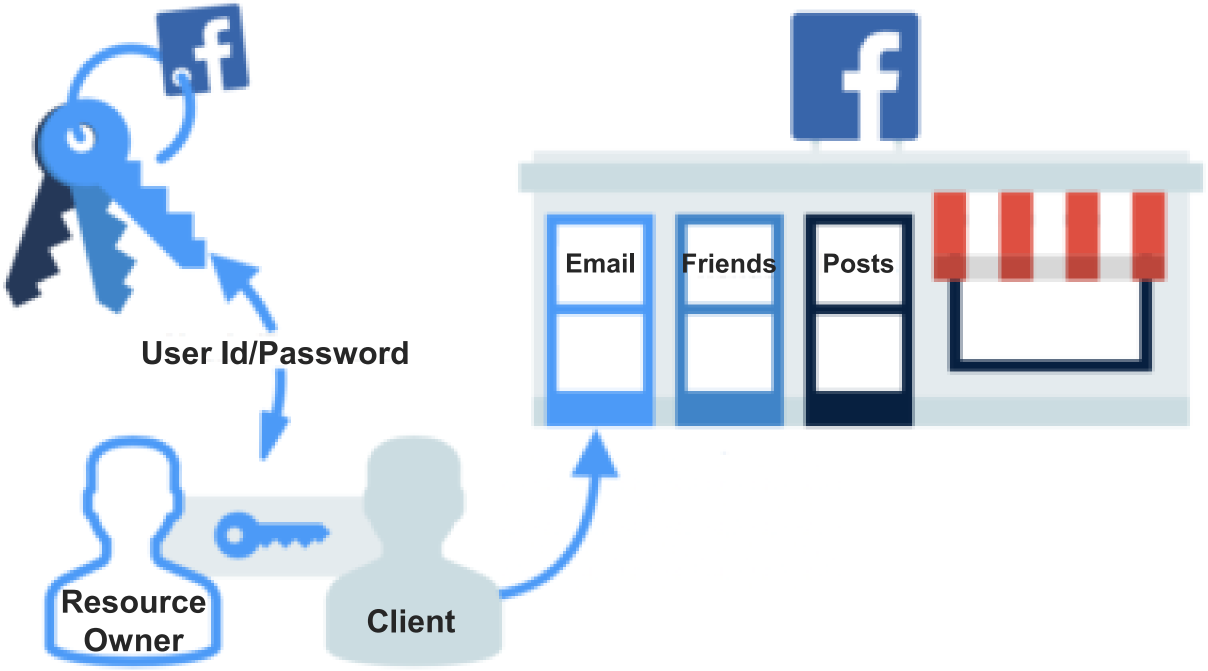
- Easy ✓
- Wide use case support ✓
- Secure ✗

Password Grant Threats

- Threat #1: Exposes the username and password
- Threat #2: No mechanism to limit scope
- Threat #3: Trains users that it’s okay to enter password in more than one place
- Threat #4: Difficult (or impossible) to add multifactor or passwordless authentication (WebCrypto, WebAuthn)

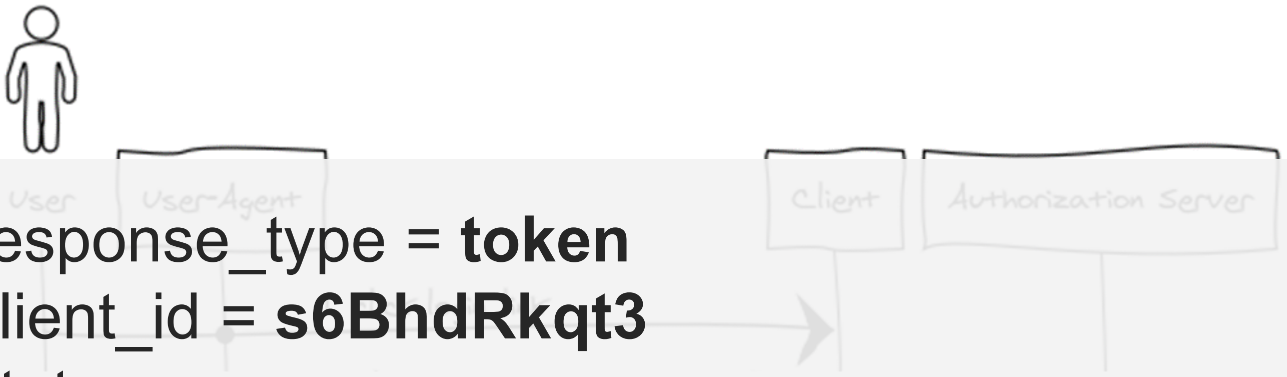
Do not use the Password grant

Implicit Grant



https://blog.oauth.io/oauth2-flow-grant-types-in-pictures/

Implicit Grant



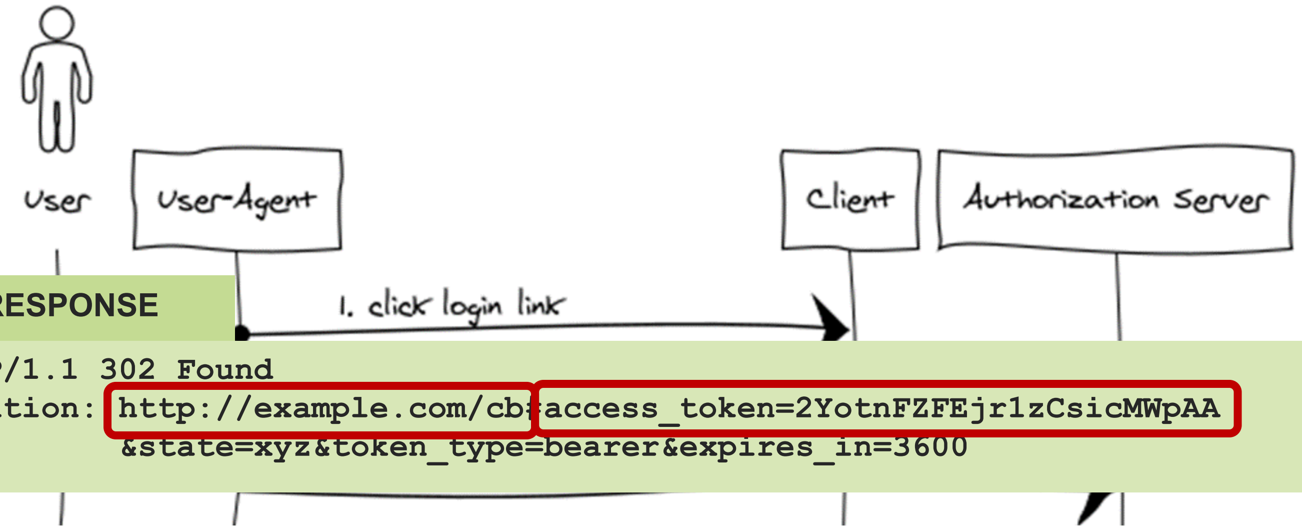
response_type = **token**
client_id = **s6BhdRkqt3**
state = **xyz**
redirect_uri = **https://client.example.com/cb**

REQUEST

```
GET /authorize?response_type=token&client_id=s6BhdRkqt3&state=xyz
      &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1
Host: server.example.com
```

Created with websequencediagrams.com

Implicit Grant



RESPONSE

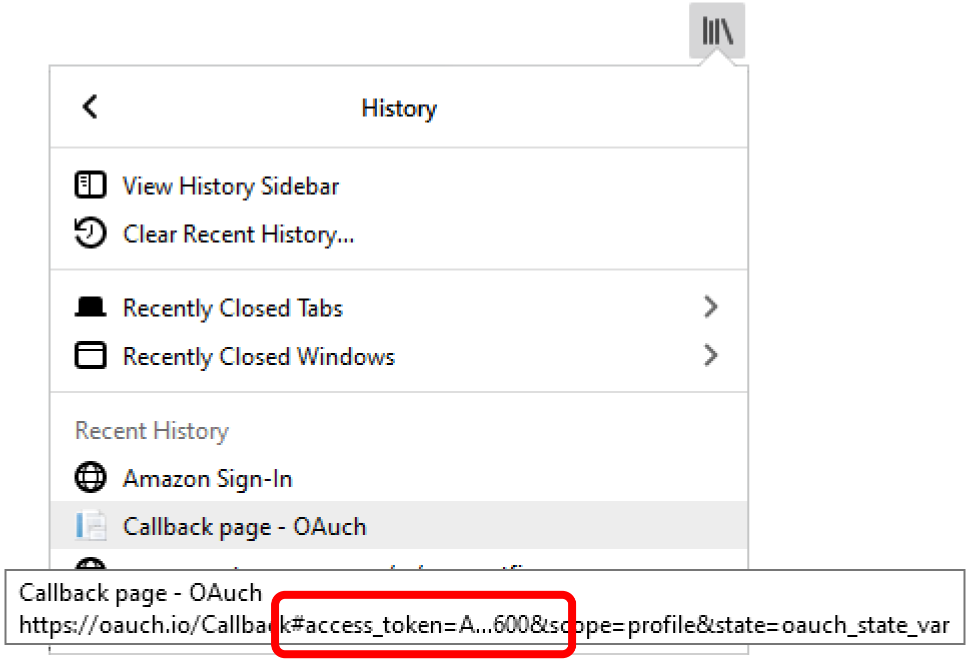
```
HTTP/1.1 302 Found
Location: http://example.com/cb#access_token=2YotnFZFEjr1zCsicMWpAA
          &state=xyz&token_type=bearer&expires_in=3600
```

Created with websequencediagrams.com

Implicit Grant

- Easy ?
- Wide use case support ✓
- Secure
  - Username and password are not exposed ✓
  - Scope can be limited ✓
  - User always uses official authorization page ✓
  - Possible to add multi-factor or passwordless authentication ✓
  - But…

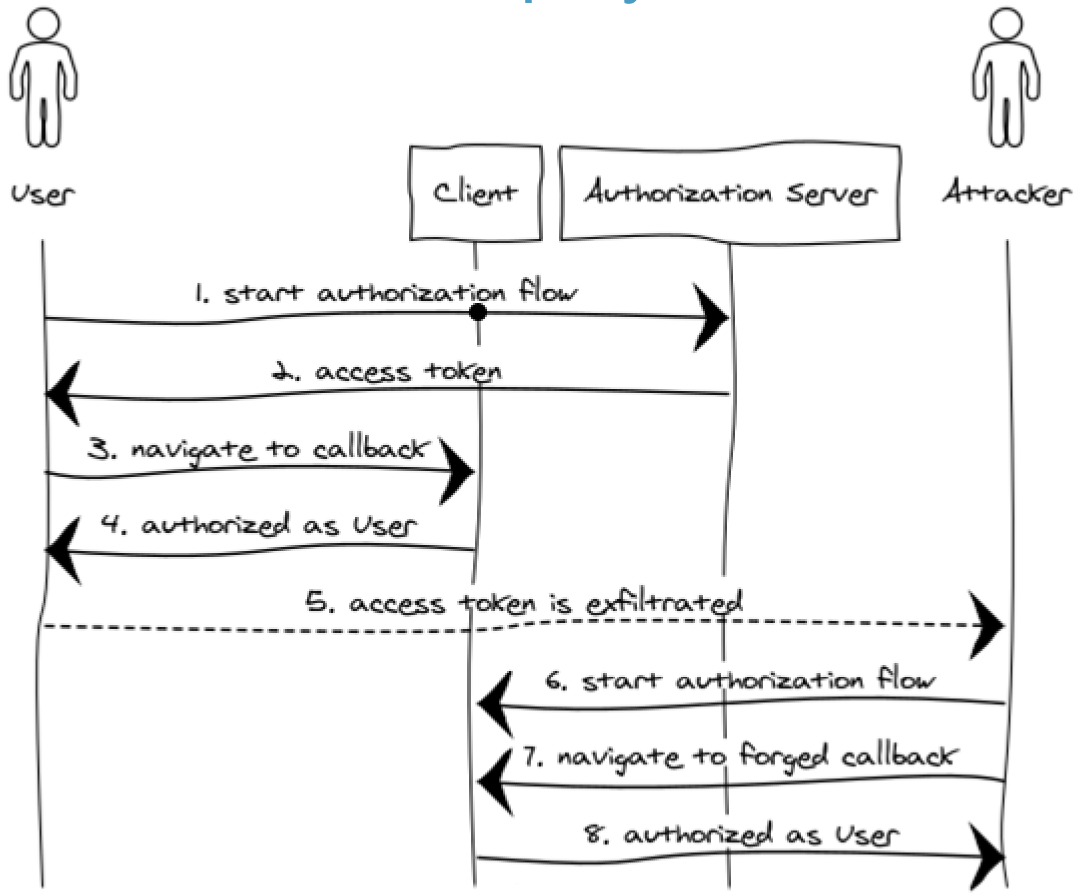
Threat #1: Access token leakage



Callback page - OAuch
https://oauch.io/Callback#access_token=A...600&scope=profile&state=oauch_state_var

KU LEUVEN DistriNet

Threat #2: Access token replay



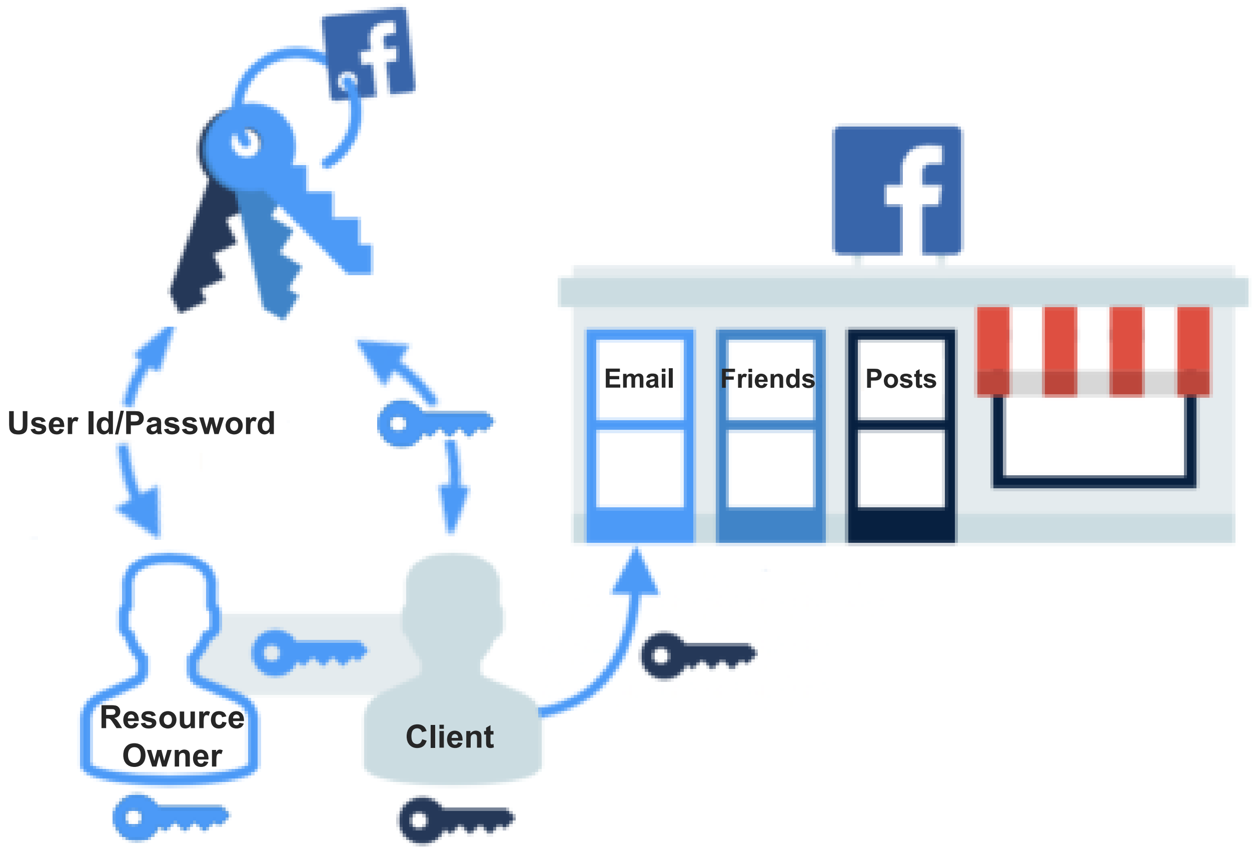
Created with websequencediagrams.com

Additional Shortcoming

- Tokens cannot be (cryptographically) bound to a client
  - Clients are not authenticated

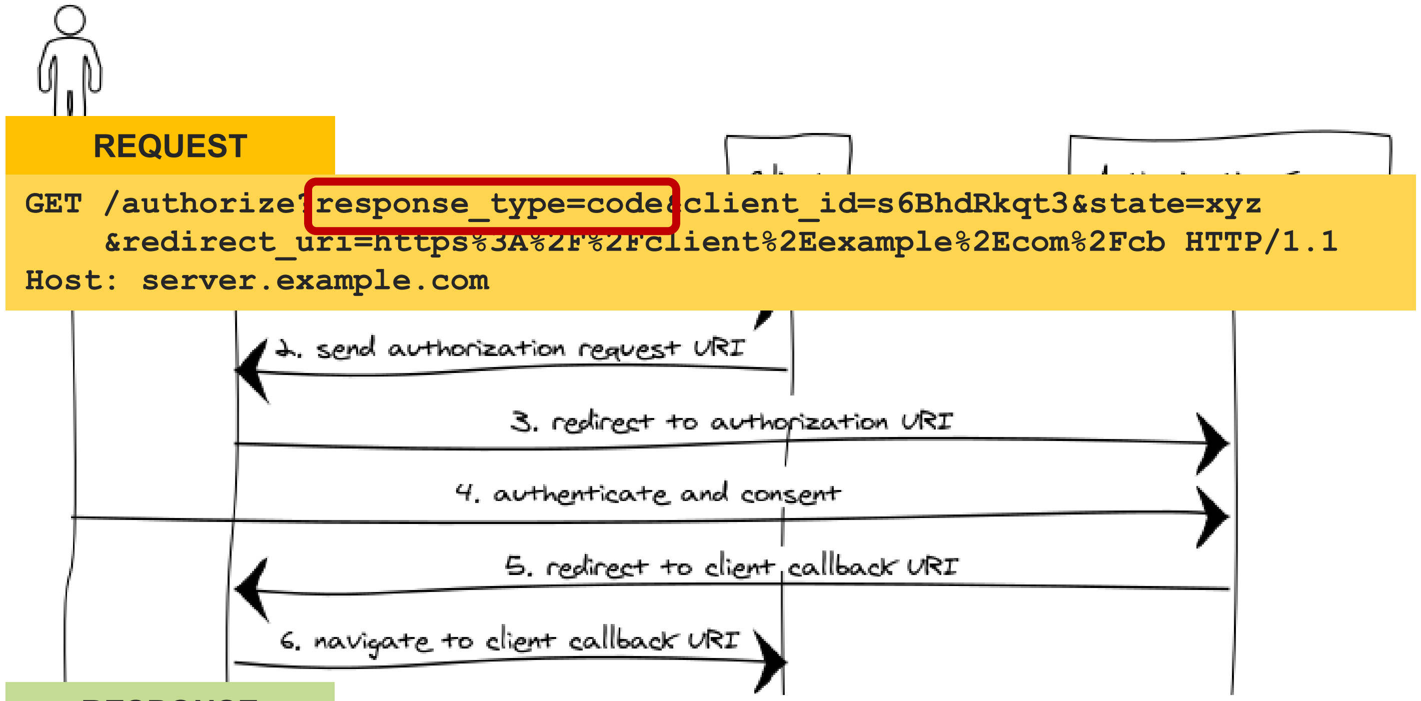
Do not use the Implicit grant

Authorization Code Grant



https://blog.oauth.io/oauth2-flow-grant-types-in-pictures/

Authorization Code Grant


2. send authorization request URI
3. redirect to authorization URI
4. authenticate and consent
5. redirect to client callback URI
6. navigate to client callback URI

REQUEST

```
GET /authorize?response_type=code&client_id=s6BhdRkqt3&state=xyz
    &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1
Host: server.example.com
```

RESPONSE

```
HTTP/1.1 302 Found
Location: https://client.example.com/cb?code=SplxlOBeZQQYbYS6WxSbIA
&state=xyz
```

Created with websequencediagrams.com

Authorization Code Grant

REQUEST

```
POST /token HTTP/1.1
Host: server.example.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&code=SplxlOBeZQQYbYS6WxSbIA
&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
```

RESPONSE

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8

{
  "access_token":"2YotnFZFEjr1zCsicMWpAA",
  "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA",
  …
}
```

Authorization Code Grant

- Easy ✗
- Wide use case support ✓
- Secure
  - All the benefits of the implicit flow ✓
  - Access tokens are not leaked ✓
  - Authorization codes cannot be replayed ✓
  - Clients can be authenticated ✓
  - But…

Threat #1: Insufficient Redirect URI Validation

- Some implementations allow redirect URI patterns
  - `https://*.benign.site/*`
  - Matches with `https://attacker.site/.benign.site/`

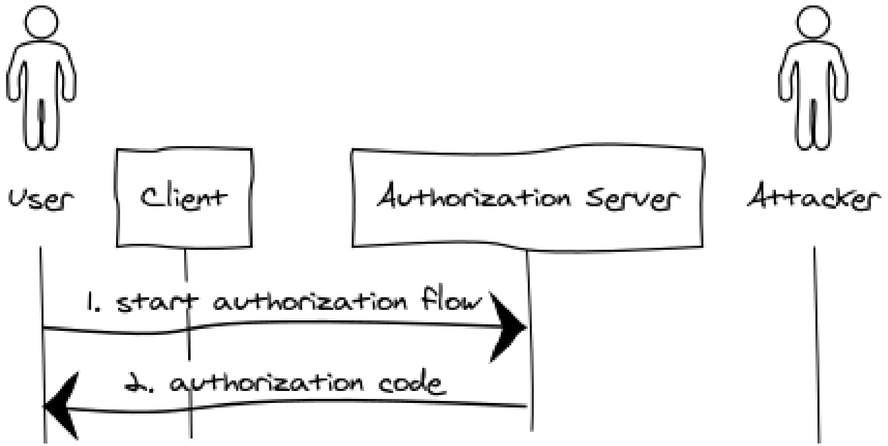
REQUEST

```
GET /authorize?response_type=code&client_id=s6BhdRkqt3&state=9ad67f13
     &redirect_uri=https%3A%2F%2Fattacker.site%2F.benign.site%2F
     HTTP/1.1
Host: server.somesite.example
```

Threat #1: Insufficient Redirect URI Validation

- Other problems exist (e.g. open redirectors, …)
- Always exactly match Redirect URIs with the registered values

Threat #2: Authorization Code Injection



Proof Key for Code Exchange (PKCE)

- Bind an authorization code to a client’s session
  - Client generates a random secret per authorization request
  - Client sends the hashed secret in the authorization request
  - When it exchanges the authorization code for an access token, it also sends the secret
    - The server can hash and compare the two hashes

Proof Key for Code Exchange (PKCE)

REQUEST

```
GET /authorize?response_type=code&client_id=s6BhdRkqt3&state=xyz
    &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
    &code_challenge=rLGaLy…5Z5Dc&code_challenge_method=S256 HTTP/1.1
Host: server.example.com
```

REQUEST

```
POST /token HTTP/1.1
Host: server.example.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&code=SplxlOBeZQQYbYS6WxSbIA
&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
&code_verifier=8WBGM8cbVT…bRzqts370
```

Use Authorization Code grant + PKCE when a user is involved

Use Cases – Grant Types

Web-server apps
authorization code + PKCE

Browser-based apps
~~implicit~~ authorization code + PKCE

Mobile apps
~~implicit~~ authorization code + PKCE

~~**Username/Password access**~~
~~password~~

Application access
client credentials

More Best Practices

- Clients *should* use sender-constrained access tokens
  - Mutual TLS for OAuth 2.0 (RFC8705)
  - OAuth 2.0 Demonstrating Proof of Possession (DPoP, RFC9449)

More Best Practices

- Clients *must not* pass access tokens in a URI query parameter
  - `https://myapi.com/posts/all?access_token=avGt23F8fWb`

More Best Practices

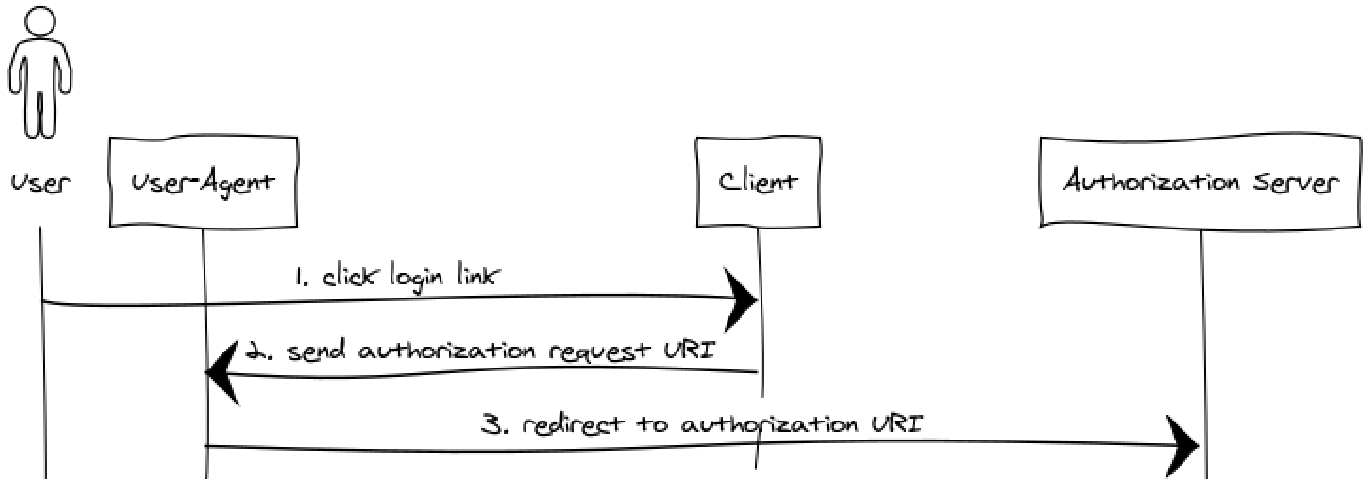
- Refresh tokens must either be sender-constrained or one-time use
  - Use refresh token rotation

Where Can I Find The Best Practices?

- OAuth 2.0 Security Best Current Practice
  - https://datatracker.ietf.org/doc/html/draft-ietf-oauth-security-topics
- The OAuth 2.1 Authorization Framework
  - https://datatracker.ietf.org/doc/html/draft-ietf-oauth-v2-1-11
  - Will be standardized soon (?)

What if you need
more
security?

Regular Authorization Requests

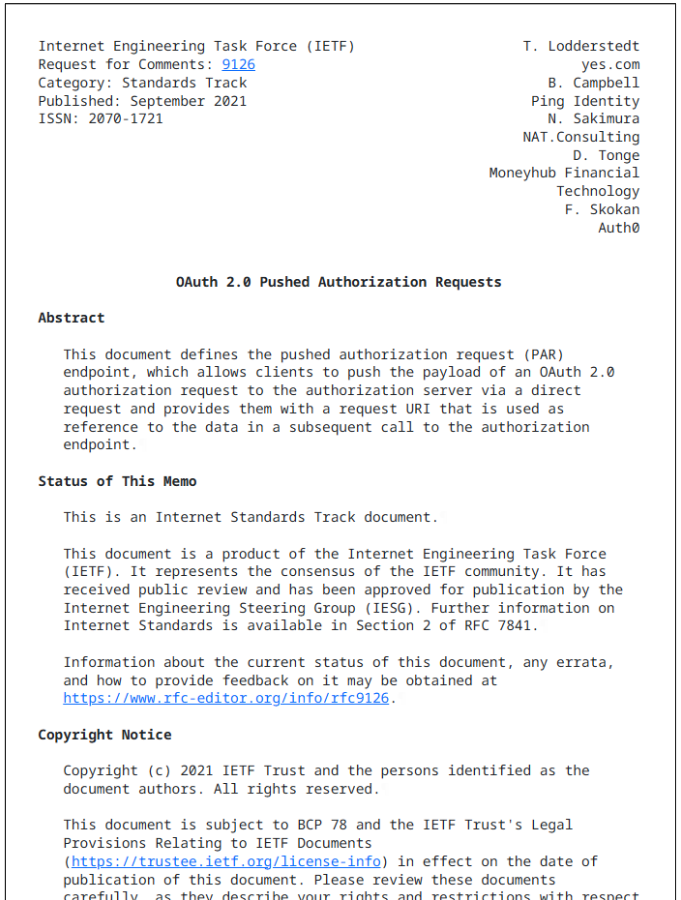

User
User-Agent
Client
Authorization Server
1. click login link
2. send authorization request URI
3. redirect to authorization URI

REQUEST

```
GET /authorize?response_type=code&client_id=CLIENT1234
    &state=duk681S8n00GsJpe7n9boxdzen&scope=profile
    &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Eorg%2Fcb
    &code_challenge=rLGaLy…5Z5Dc&code_challenge_method=S256 HTTP/1.1
Host: server.example.com
```

OAuth 2.0 Pushed Authorization Requests (RFC 9126)

https://datatracker.ietf.org/doc/html/rfc9126



```
Internet Engineering Task Force (IETF)                   T. Lodderstedt
Request for Comments: 9126                                      yes.com
Category: Standards Track                                    B. Campbell
Published: September 2021                                 Ping Identity
ISSN: 2070-1721                                              N. Sakimura
                                                          NAT.Consulting
                                                                D. Tonge
                                                      Moneyhub Financial
                                                              Technology
                                                               F. Skokan
                                                                   Auth0
```

OAuth 2.0 Pushed Authorization Requests

Abstract

This document defines the pushed authorization request (PAR) endpoint, which allows clients to push the payload of an OAuth 2.0 authorization request to the authorization server via a direct request and provides them with a request URI that is used as reference to the data in a subsequent call to the authorization endpoint.

Status of This Memo

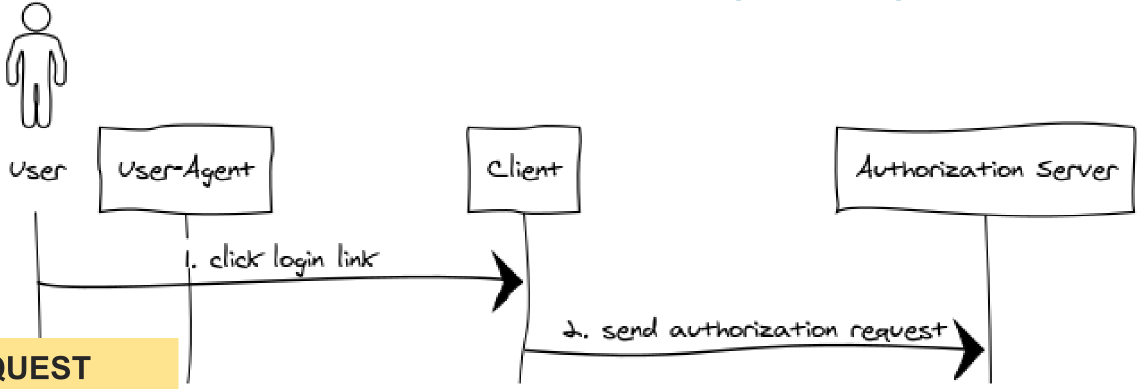
This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9126.

Copyright Notice

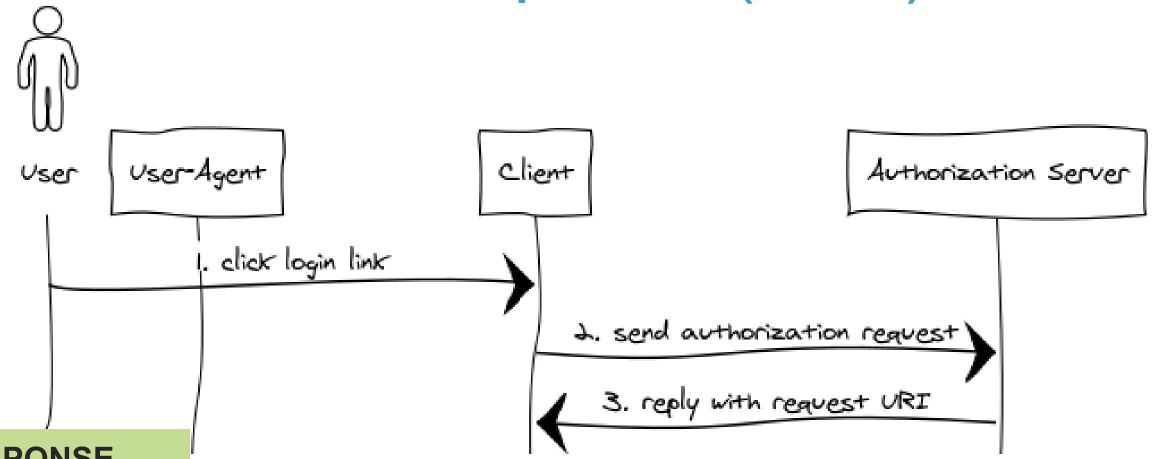
Pushed Authorization Requests (PAR)



REQUEST

```
POST /as/par HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded

response_type=code&client_id=CLIENT1234
&state=duk681S8n00GsJpe7n9boxdzen&scope=profile
&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Eorg%2Fcb
&code_challenge=rLGaLy…5Z5Dc&code_challenge_method=S256
&client_assertion_type=
  urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-type%3Ajwt-bearer
&client_assertion=eyJraWQiOiJ…dHBzOi8vc
```

Pushed Authorization Requests (PAR)

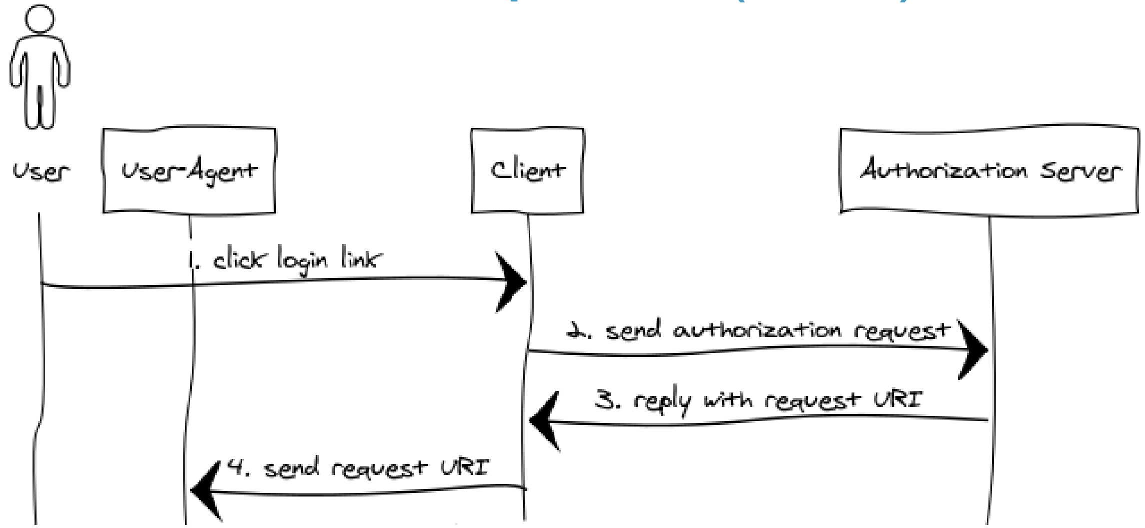


RESPONSE

```
HTTP/1.1 201 Created
Cache-Control: no-cache, no-store
Content-Type: application/json

{
  "request_uri": "urn:example:bwc4JK-ESC0w8acc191e-Y1LTC2",
  "expires_in": 90
}
```

KU LEUVEN DistriNet

Pushed Authorization Requests (PAR)

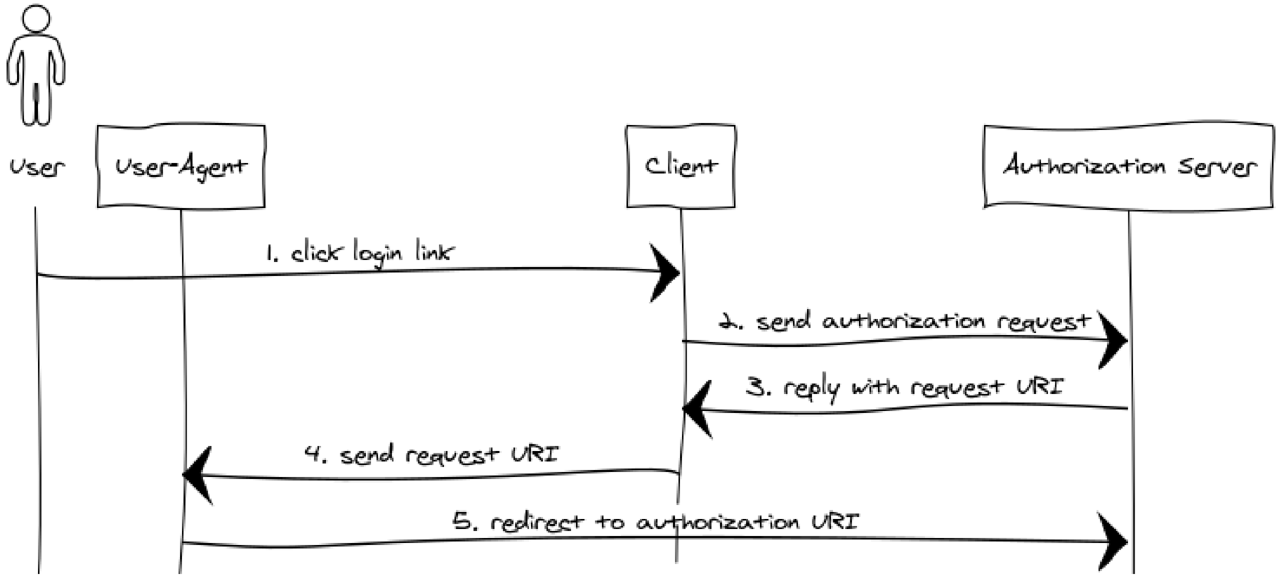


RESPONSE

```
HTTP/1.1 303 See other
Location: https://as.example.com/authorize?client_id=CLIENT1234
  &request_uri=urn%3Aexample%3Abwc4JK-ESC0w8acc191e-Y1LTC2
```

KU LEUVEN DistriNet

Pushed Authorization Requests (PAR)


User
User-Agent
Client
Authorization Server
1. click login link
2. send authorization request
3. reply with request URI
4. send request URI
5. redirect to authorization URI

Can we get
even more
secure?

The Financial-Grade API Security Profile

- Extension of OpenID Connect
  - OpenID Connect is an extension of OAuth 2.0
- Focus on high-security scenarios (e.g., banking apps)
- Gives additional requirements
  - E.g., which crypto algorithms to use, requiring asymmetric crypto instead of client passwords, …

The Financial-Grade API Security Profile

- Current standards:
  - Financial-grade API Security Profile (FAPI) 1.0 – Part 1: Baseline
  - Financial-grade API Security Profile (FAPI) 1.0 – Part 2: Advanced
- New specification coming up:
  - FAPI 2.0 Security Profile

Conclusion

Conclusion

- OAuth 2.0 is about **delegation**
  - Clients can ask permission to access protected resources on a resource owner's (user's) behalf
- OAuth 2.0 is a secure protocol ***if used correctly***
  - Most servers and clients do not follow the best practices

DistriNet

Thank you!

https://distrinet.cs.kuleuven.be/

Pieter.Philippaerts@kuleuven.be