How (Not) to Use OAuth

in 2024

Daniel Fett

About me: Daniel Fett

- Coauthor of the **OAuth Security Best Current Practice RFC**
- Standardization activities: IETF OAuth, OpenID Foundation
- PhD on web protocol security (formal security analysis)
- Product owner in the German EUDI Wallet project @ SPRIN-D

In this Talk

What is OAuth 2.0? Quick recap!

Security Challenges for OAuth

The three most important recommendations in the Security BCP

… and why you don’t have to remember them

Who is familiar with OAuth?

OAuth is a standard
for federated authorization

Authorization

User

authorizes

Banking App

Client

to access

Online Banking Account

Authorization Server & Resource Server

Authentication

User

authenticates to

airbnb

Relying Party

using identity from

Identity Provider

Authorization

User

authorizes

Banking App

Client

to access

Online Banking Account

Authorization Server & Resource Server

Authentication

User

authenticates to

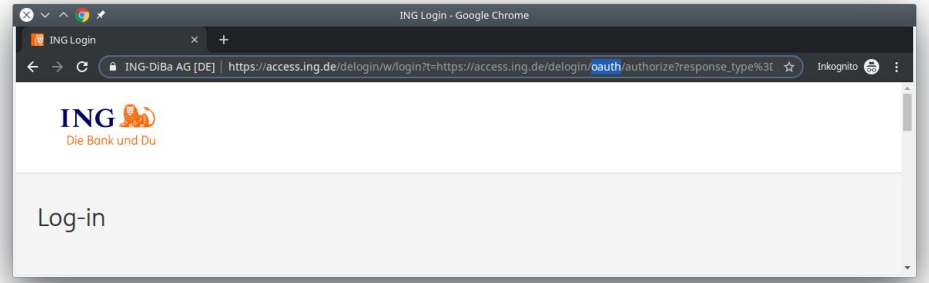
airbnb

Relying Party

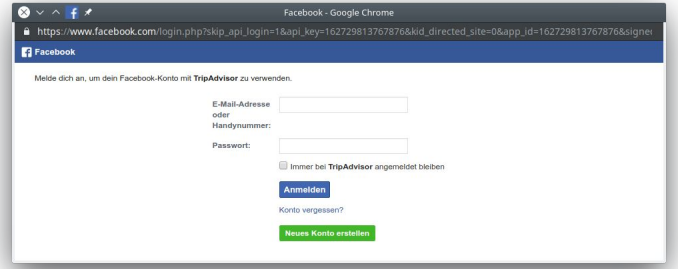
using identity from

Identity Provider

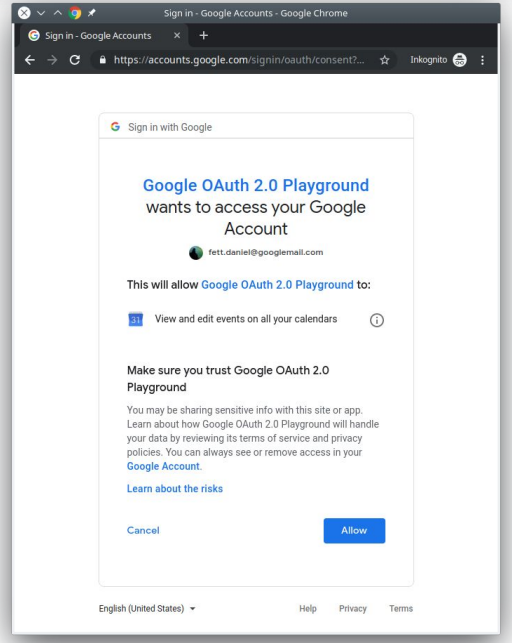
OAuth & friends in the Wild



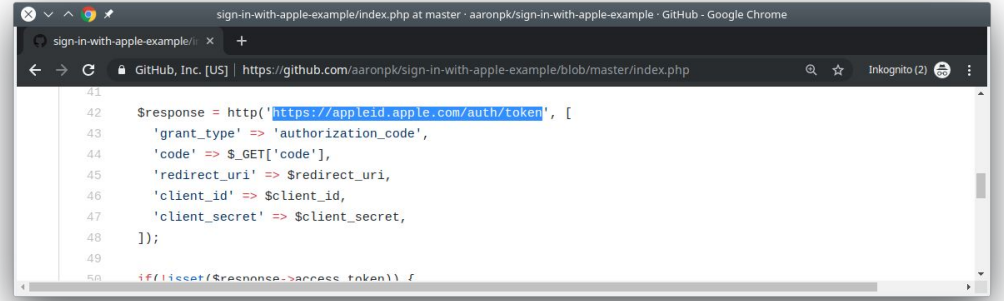
Banking



Facebook



Google



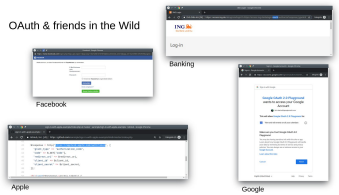
Apple

e-health

open banking

e-signing

open insurance



OAuth 2.0!

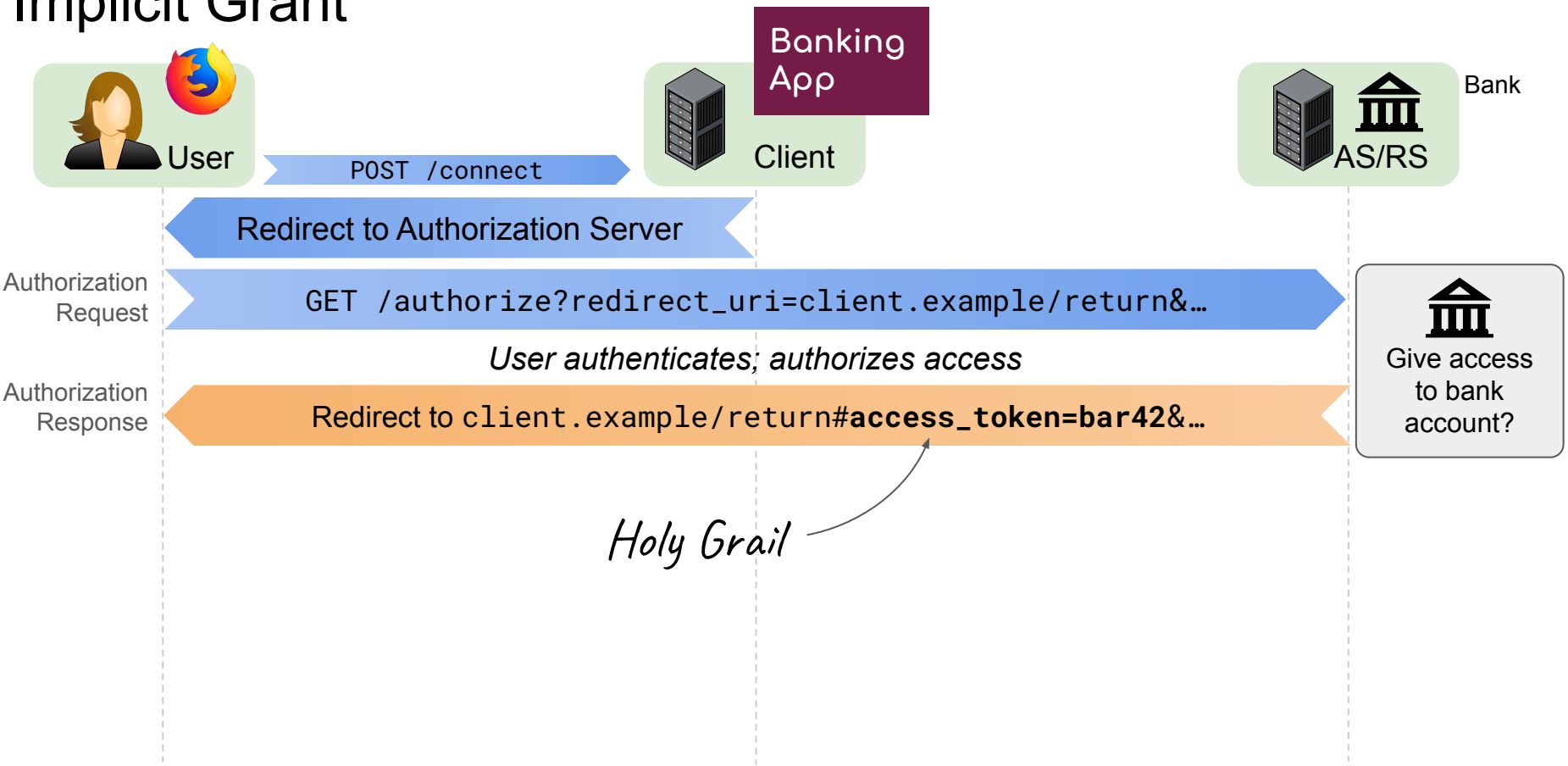
open finance

e-government

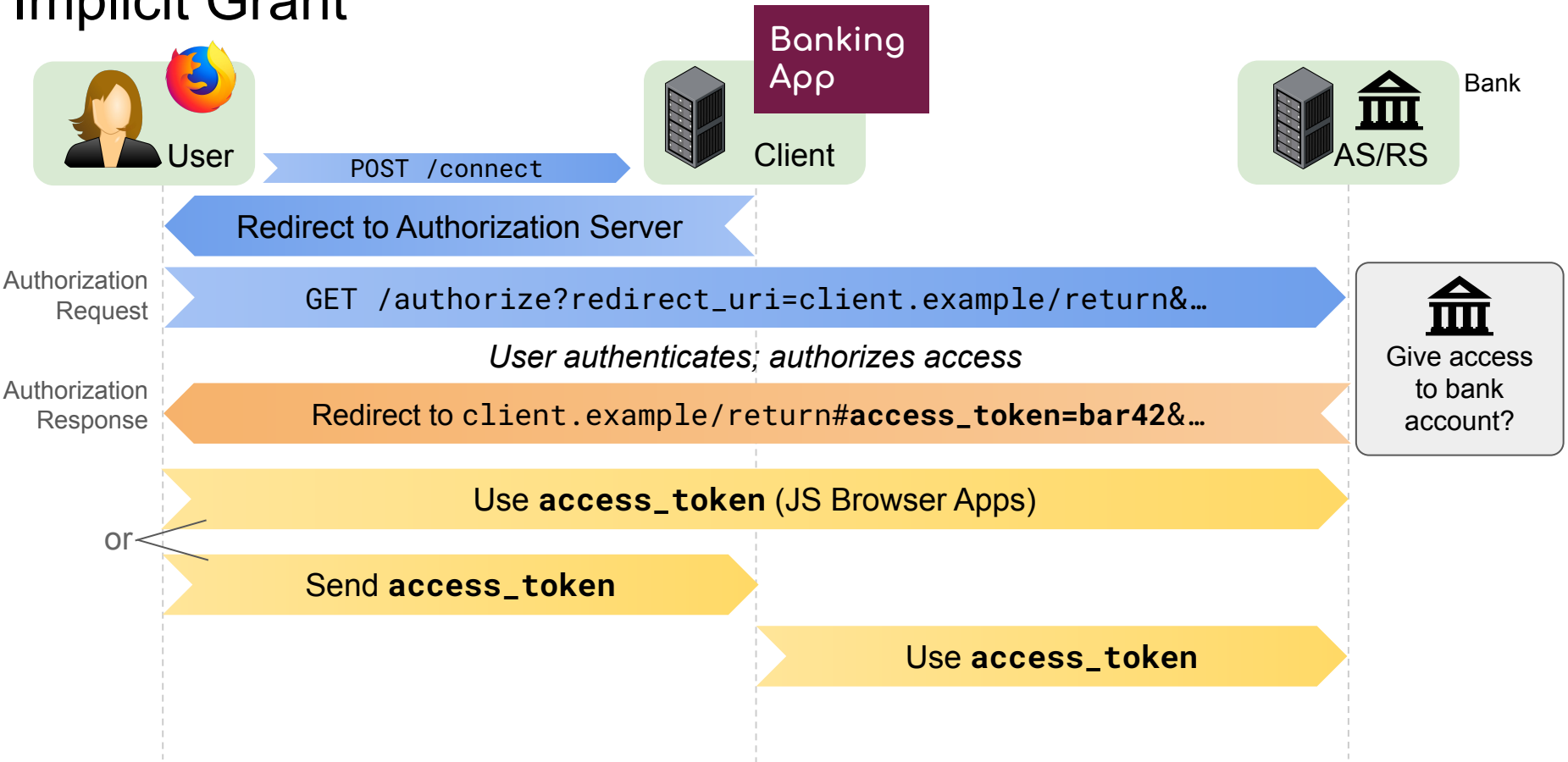
open consumer data

digital identity ecosystems

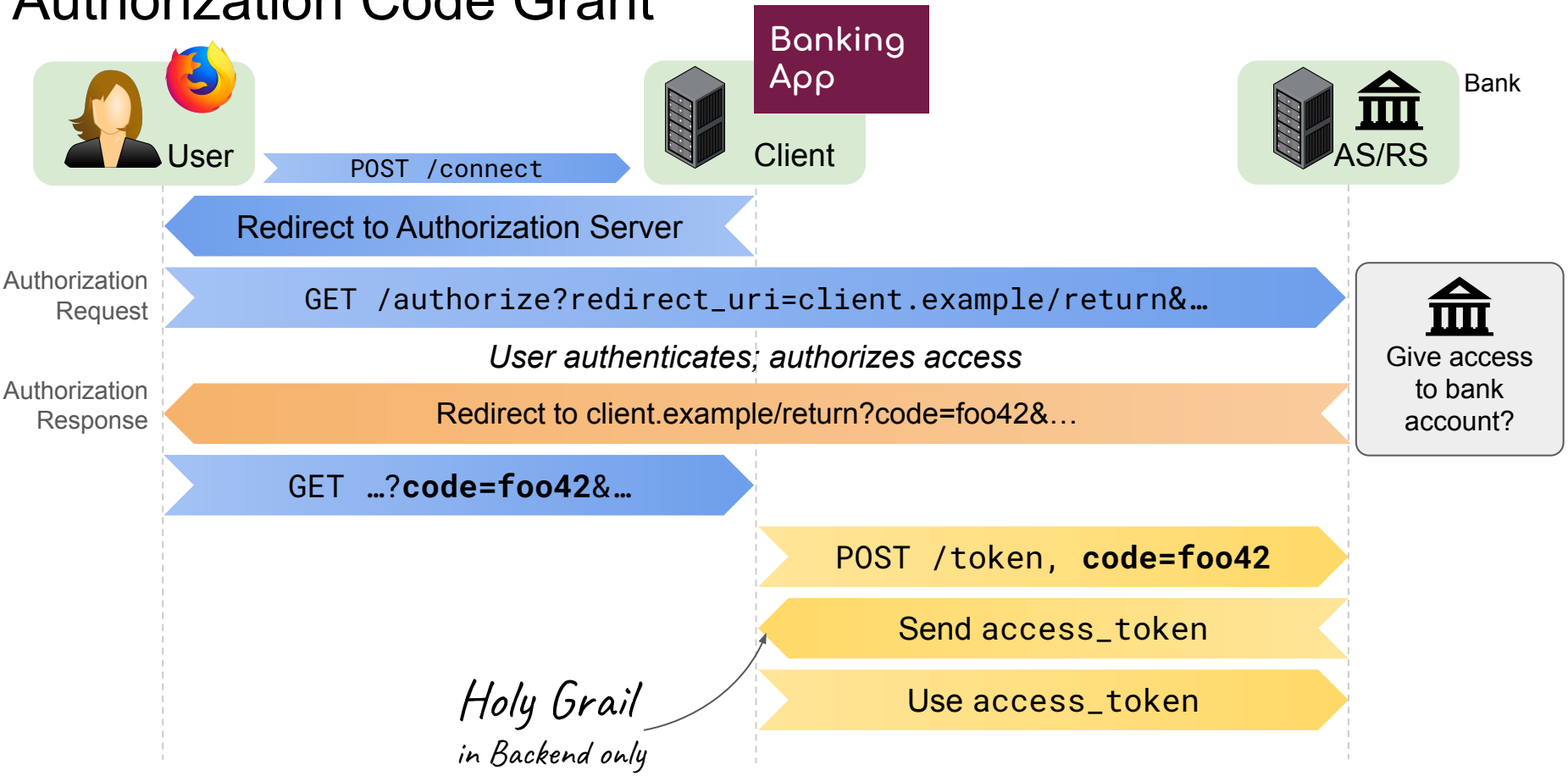
Implicit Grant



Implicit Grant



Authorization Code Grant



Twelve Years after RFC6749:

Security Challenges for OAuth

Challenge 1: Implementation Flaws

- We still see many implementation flaws
- Known anti-patterns are still used
  - Insufficient redirect URI checking (code/token is redirected to attacker)
  - `state` parameter is not used properly to defend against CSRF
  - …
- Clients worse than authorization/resource servers

- [Li et al., 2014]
  60 chinese clients, **more than half** vulnerable to CSRF
- [Yang et al., 2016]
  Out of 405 clients, **55%** do not handle `state` (CSRF protection) correctly
- [Shebab et al., 2015]
  25% of OAuth clients in Alexa Top 10000 vulnerable to CSRF
- [Chen et al., 2014]
  89 of 149 mobile clients vulnerable to one or more attacks
- [Wang et al., 2013]
  Vulnerabilities in Facebook PHP SDK and other OAuth SDKs
- [Sun et al., 2012]
  96 Clients, **almost all** vulnerable to one or more attacks

Challenge 2: High-Stakes Environments

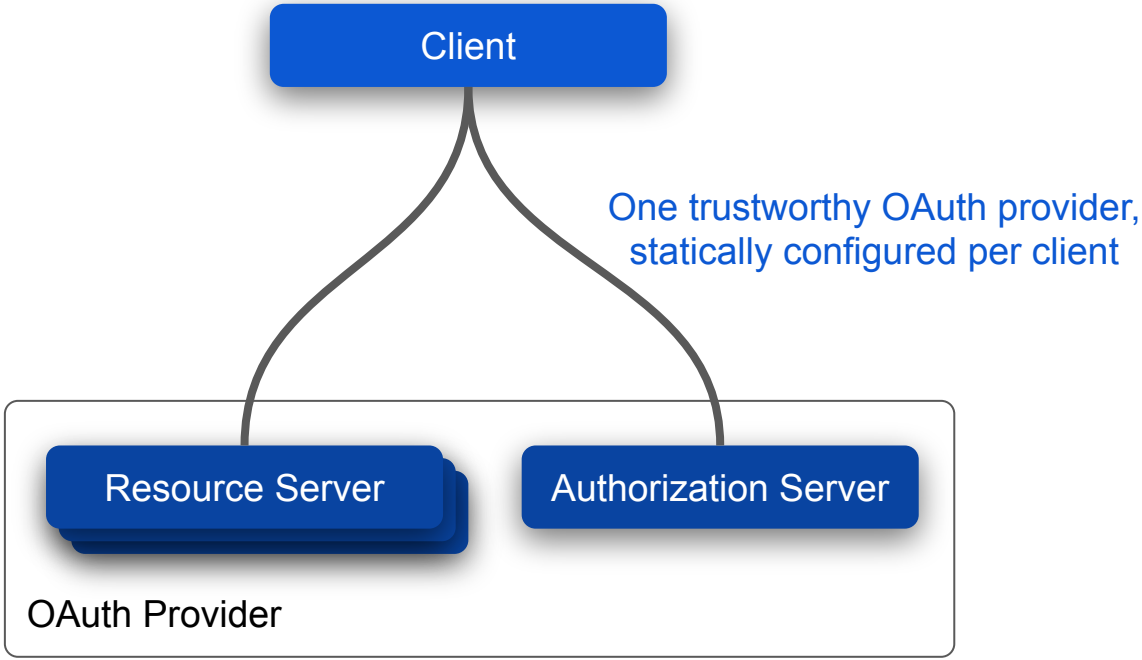
New Use Cases require a very high level of security

- **Open Banking:** Account access, payments, wire transfers
- **eHealth:** Access to health data
- **eSigning:** Legally binding digital signatures
- **Wallets (EU Digital Identity Wallets, eIDAS 2.0):**
  Identification on *Level of Assurance High* -> Kristina's talk

Far beyond the scope of the original security threat model!

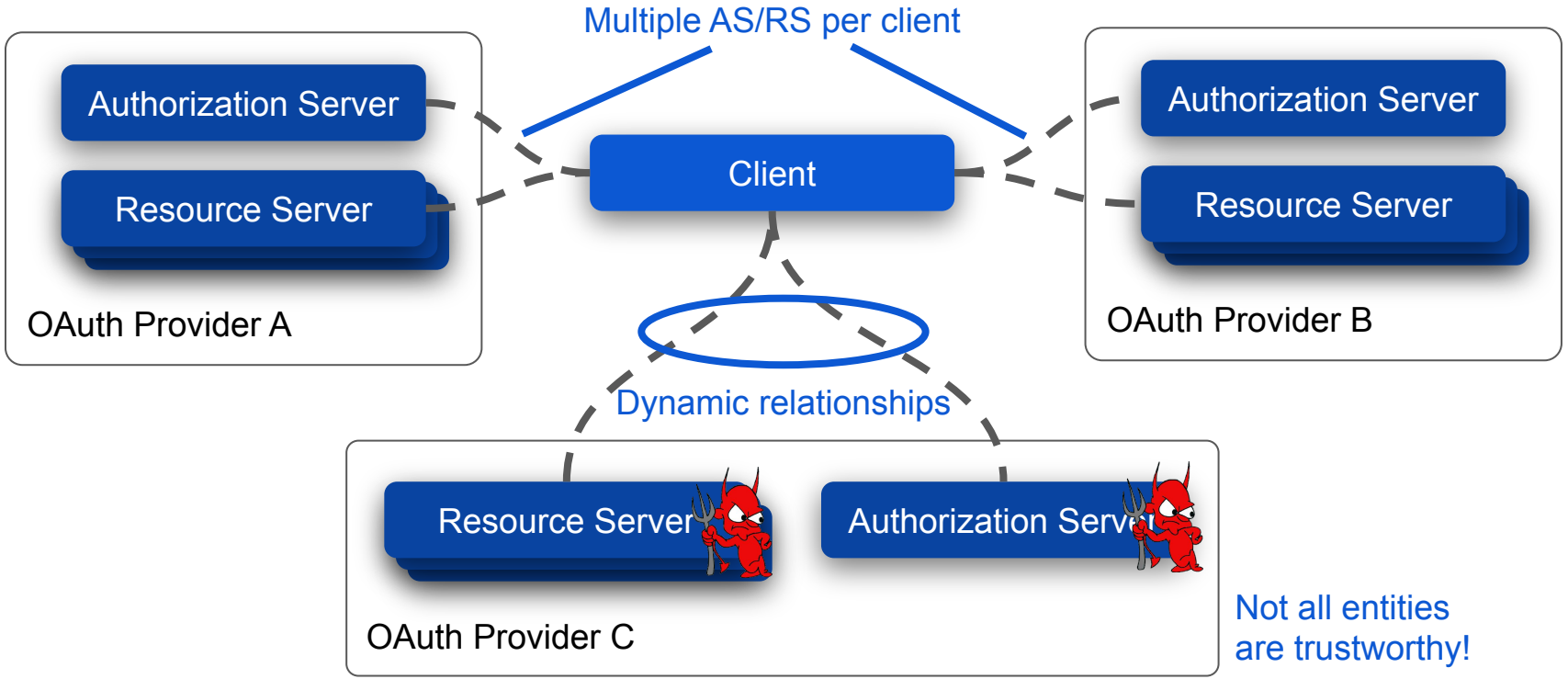
Challenge 3: Dynamic and Complex Setups

Originally anticipated:



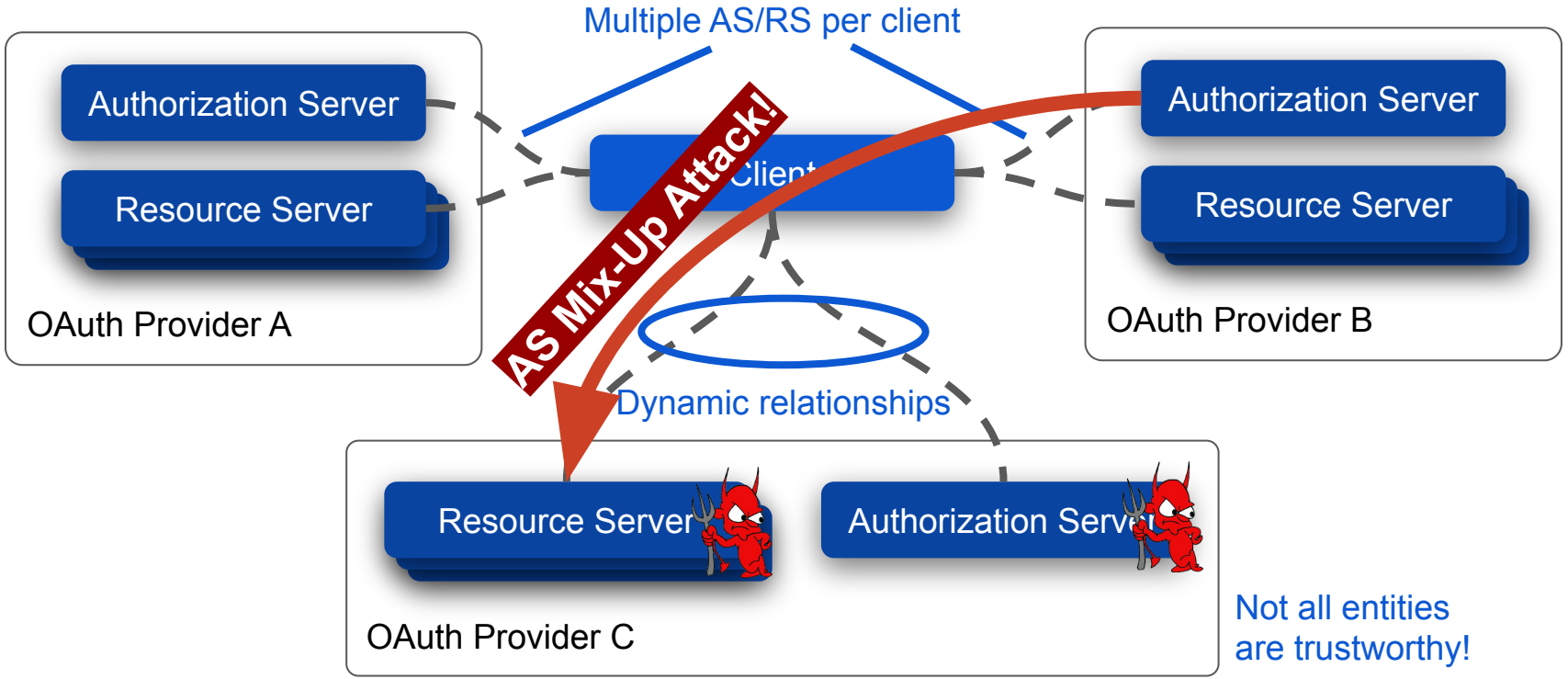
Challenge 3: Dynamic and Complex Setups

Today:



Challenge 3: Dynamic and Complex Setups

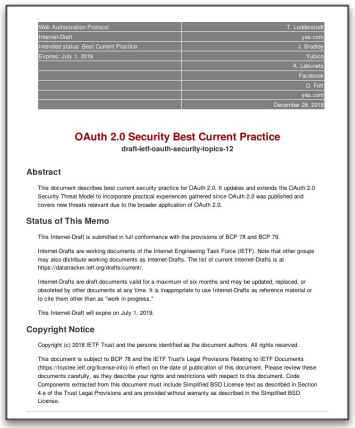
Today:



How to address these challenges?

OAuth 2.0 Security Best Current Practice RFC

- Under development at the IETF
- Refined and enhanced security guidance for OAuth 2.0 implementers
- Complements existing security guidance in RFCs 6749, 6750, and 6819



Web Authorization Protocol	T. Lodderstedt
Internet-Draft	yes.com
Intended status: Best Current Practice	J. Bradley
Expires: July 1, 2019	Yubico
	A. Labunets
	Facebook
	D. Fett
	yes.com
	December 28, 2018

OAuth 2.0 Security Best Current Practice

draft-ietf-oauth-security-topics-12

Abstract

This document describes best current security practice for OAuth 2.0. It updates and extends the OAuth 2.0 Security Threat Model to incorporate practical experiences gathered since OAuth 2.0 was published and covers new threats relevant due to the broader application of OAuth 2.0.

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 July 1, 2019.

Copyright Notice

Copyright (c) 2018 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 (https://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 include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

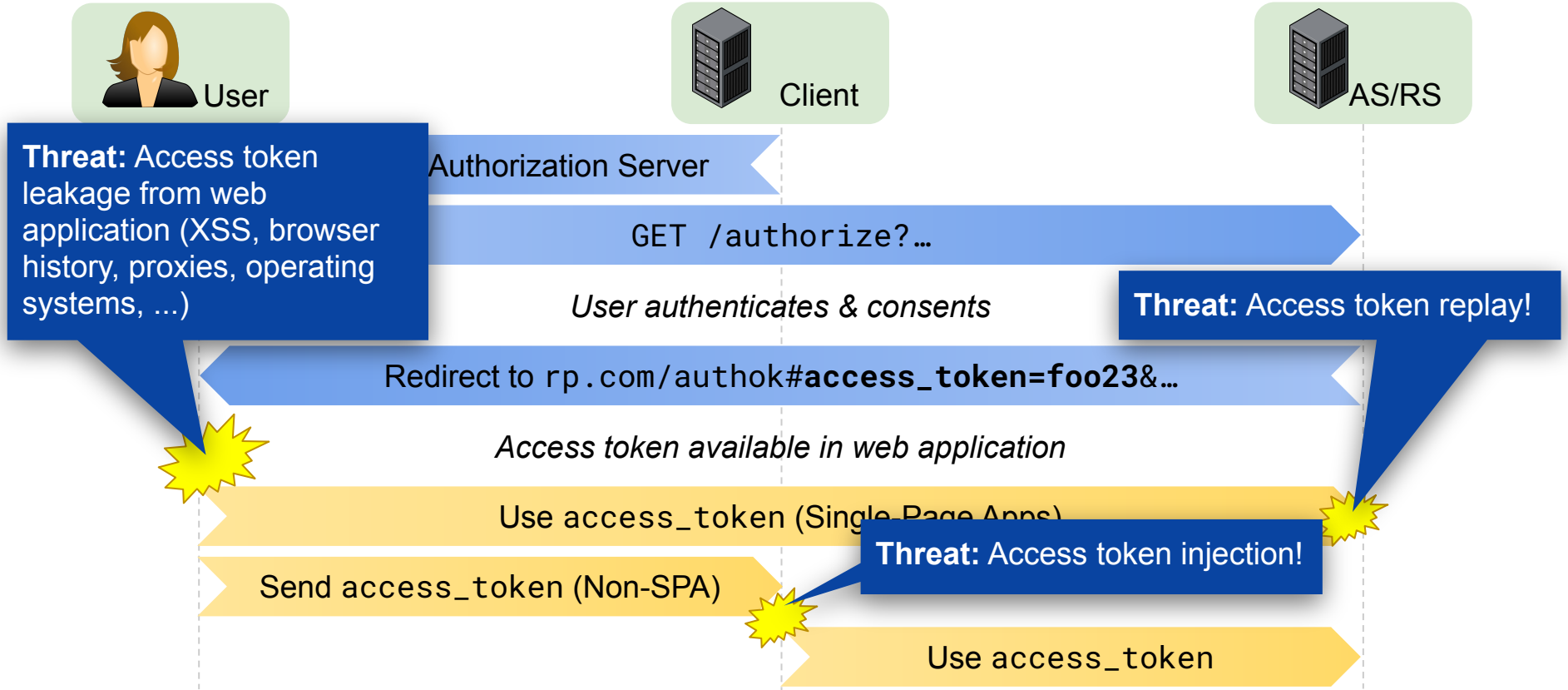
- Updated, more comprehensive Threat Model
- Description of Attacks and Mitigations
- Simple and actionable recommendations

Input from **practice** and **formal analysis**

IETF®

The Three Most Important Recommendations in the OAuth Security BCP

① Do not use the OAuth Implicit Grant any longer!



The Implicit Grant ...

- sends **powerful** and **potentially long-lived** tokens through the browser,
- lacks features for **sender-constraining** access tokens,
- provides no protection against access token **replay and injection**, and
- provides no **defense in depth** against XSS, URL leaks, etc.!

Why is Implicit even in RFC6749?

No Cross-Origin Resource Sharing in 2012!
⇒ No way of (easily) using OAuth in SPAs.

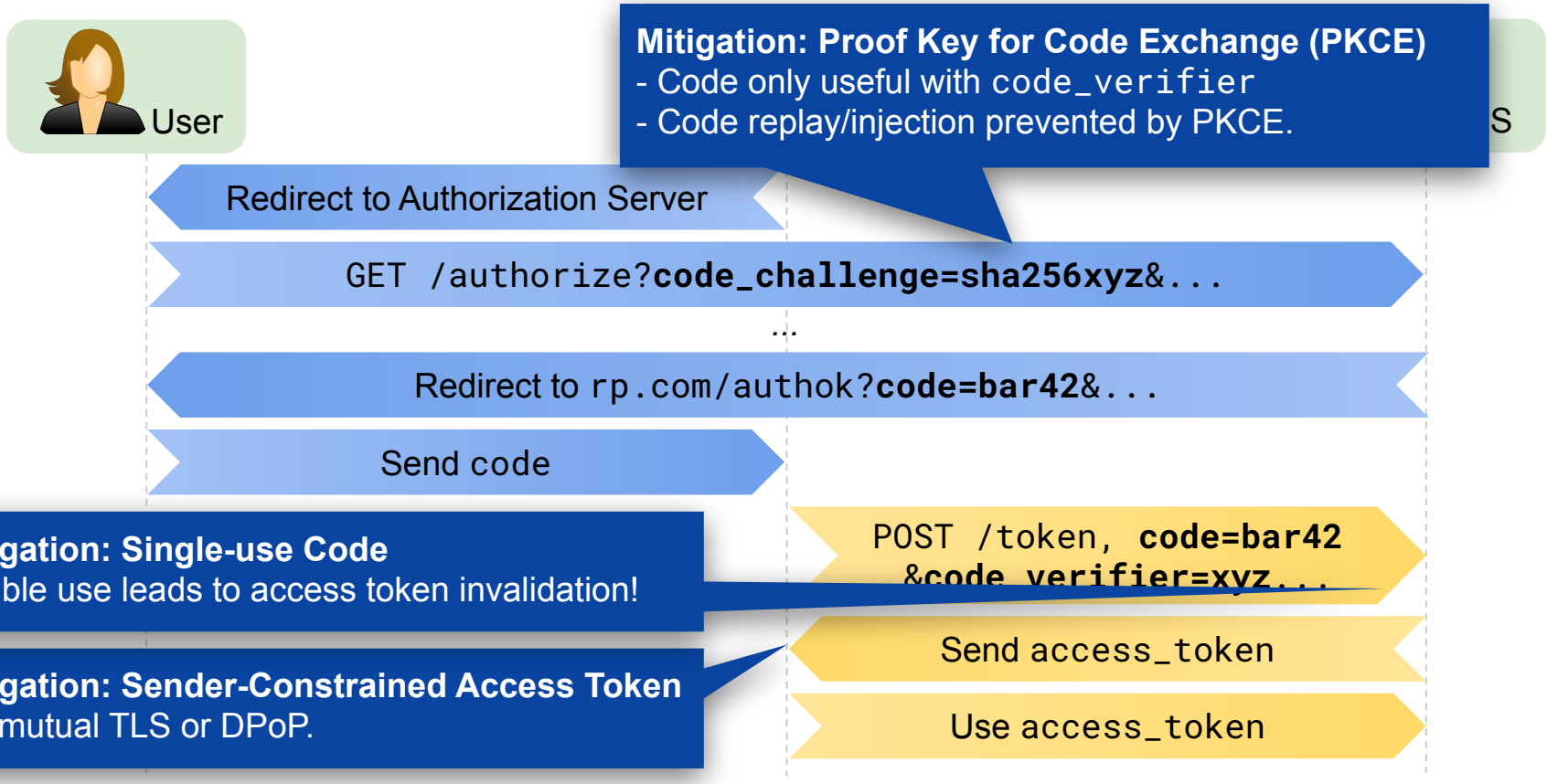
⇒ Not needed in 2024!

Recommendation

“Clients SHOULD NOT use the implicit grant [...]”

“Clients SHOULD instead use the response type code (aka authorization code grant type) [...]”

Use the Auth Code Grant with PKCE & DPoP/mTLS!



Authorization Code Grant with PKCE & DPoP/mTLS …

- protects against **code and token replay and injection,**
- supports **sender-constraining** of access tokens,
- protects against **CSRF** better than `state` does,
- provides **defense in depth!**

Recommendation

“Clients utilizing the authorization grant type MUST use PKCE [...]”

“Authorization servers SHOULD use TLS-based methods for sender-constrained access tokens [...]”

② Stop Redirects Gone Wild!

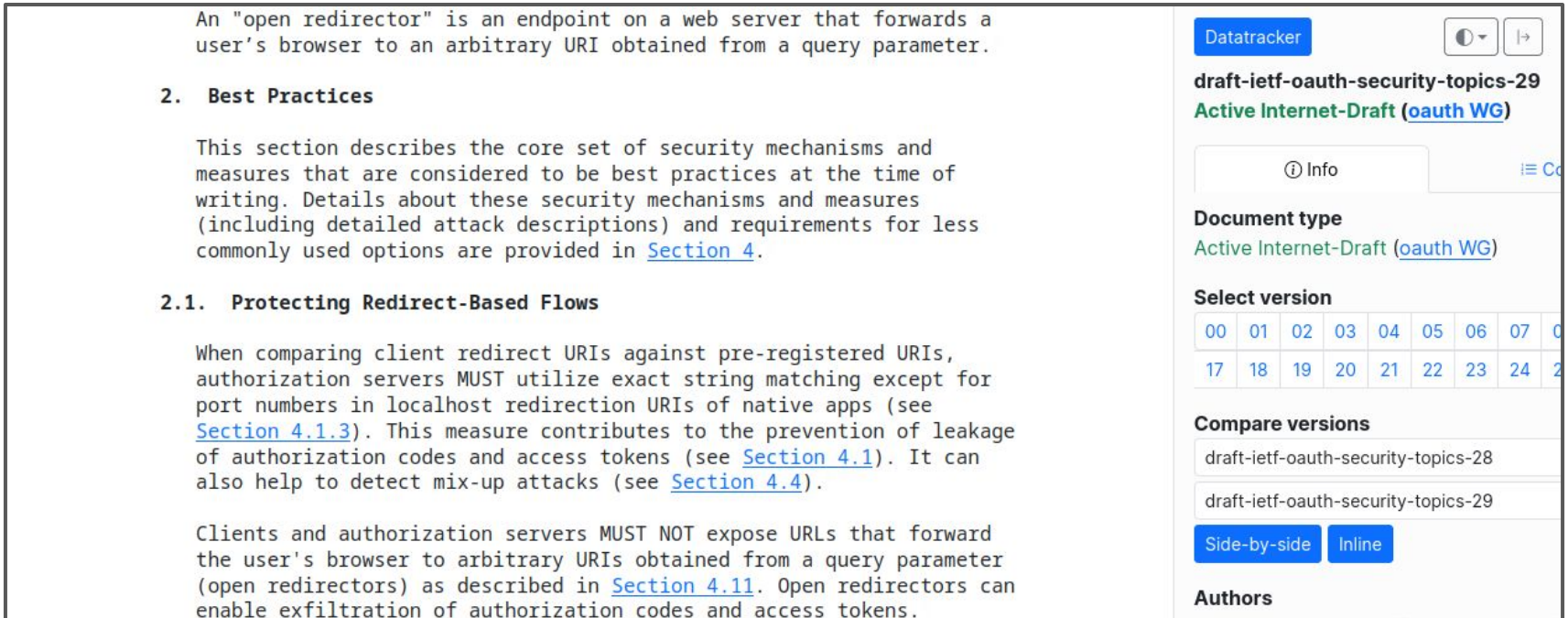
- Enforce exact redirect URI matching
  - Simpler to implement on AS side
  - Adds protection layer against open redirection
- Clients MUST avoid open redirectors!
  - Use whitelisting of target URLs
  - or authenticate redirection request

③ Limit Privileges of Access Tokens!

- Sender-constraining (mTLS or DPoP)
- Receiver-constraining (only valid for certain RS)
- Reduce scope and lifetime and use refresh tokens - defense in depth!

But wait, there’s more…

The treasure trove: Section 2 of draft-ietf-oauth-security-topics!



An "open redirector" is an endpoint on a web server that forwards a user's browser to an arbitrary URI obtained from a query parameter.

2. Best Practices

This section describes the core set of security mechanisms and measures that are considered to be best practices at the time of writing. Details about these security mechanisms and measures (including detailed attack descriptions) and requirements for less commonly used options are provided in Section 4.

2.1. Protecting Redirect-Based Flows

When comparing client redirect URIs against pre-registered URIs, authorization servers MUST utilize exact string matching except for port numbers in localhost redirection URIs of native apps (see Section 4.1.3). This measure contributes to the prevention of leakage of authorization codes and access tokens (see Section 4.1). It can also help to detect mix-up attacks (see Section 4.4).

Clients and authorization servers MUST NOT expose URLs that forward the user's browser to arbitrary URIs obtained from a query parameter (open redirectors) as described in Section 4.11. Open redirectors can enable exfiltration of authorization codes and access tokens.

Datatracker

draft-ietf-oauth-security-topics-29

Active Internet-Draft (oauth WG)

Info

Document type

Active Internet-Draft (oauth WG)

Select version

00 01 02 03 04 05 06 07

17 18 19 20 21 22 23 24

Compare versions

draft-ietf-oauth-security-topics-28

draft-ietf-oauth-security-topics-29

Side-by-side Inline

Authors

I’m confused…

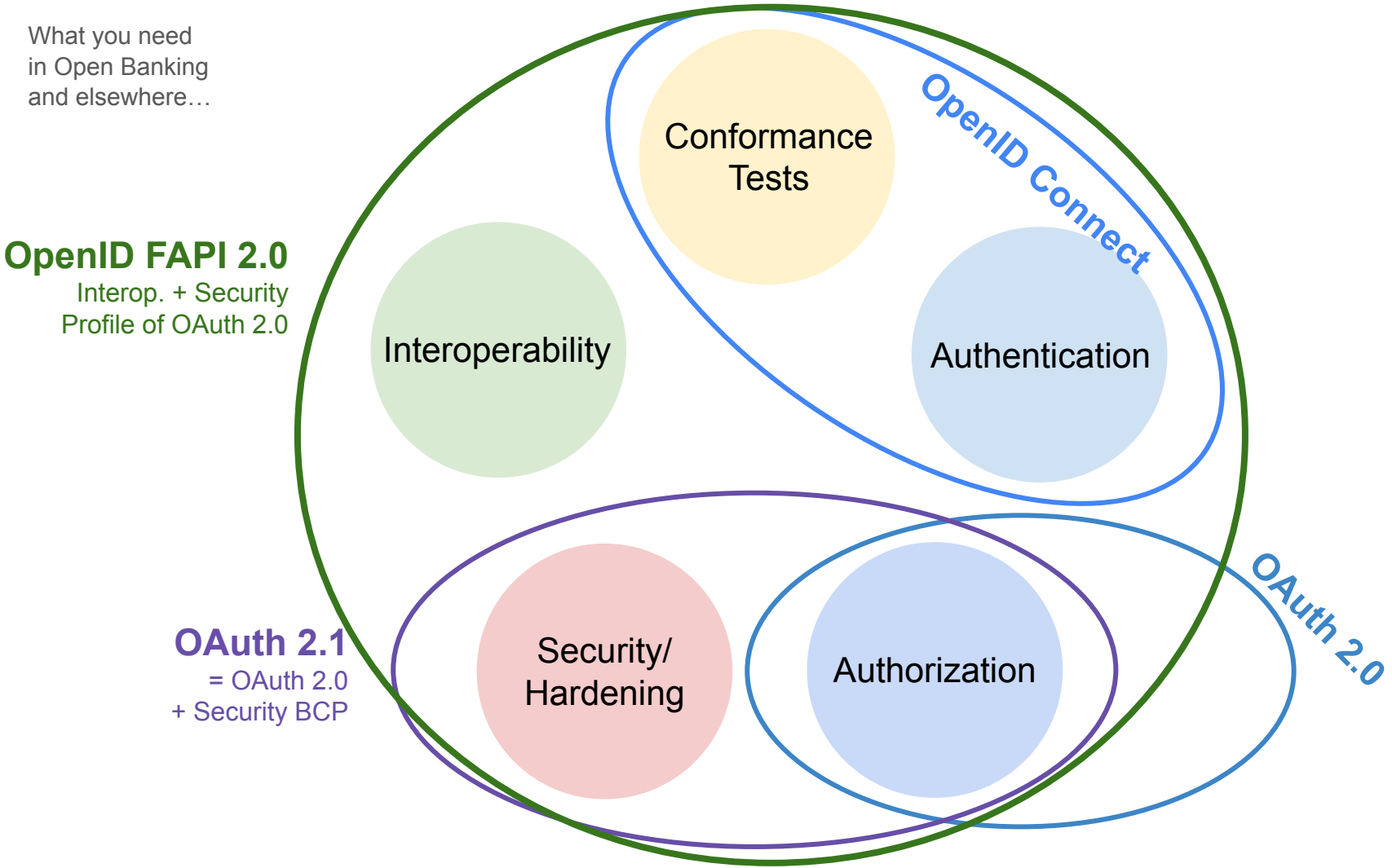
Should I Even Use OAuth?

Absolutely!

- Standards are good
  - Battle-proven libraries
  - Interoperability
- Years of experience, dozens of security analyses
- Custom-built solutions prone to repeat even the most basic vulnerabilities
- Protection against strong attackers
- Formal proof of security
- But:
  - Know your threat model
  - Read the security advice, including the BCP draft
  - Implement the latest security features

} **… or use OAuth 2.1 / FAPI 2.0**

What you need
in Open Banking
and elsewhere…



FAPI?

Financial API

FAPI?

~~Financial API~~

Financial API *Security Profile*

FAPI?

~~Financial API~~

~~Financial API *Security Profile*~~

Financial-*grade* API Security Profile

FAPI?

~~Financial API~~

~~Financial API *Security Profile*~~

~~Financial-*grade* API Security Profile~~

FAPI

FAPI!

Security, interoperability, and feature profile for OAuth 2.0

Implements all the security recommendations from the OAuth Security BCP

Usable for all APIs, including high-security applications.

FAPI 2.0: Latest version

Follow up

danielfett.de/publications

List of Drafts/Specifications

oauth.secworkshop.events

OAuth Security Workshop,
February 26-28, 2025,
Reykjavik

danielfett.de/publications

List of Drafts/Specifications,
Talk on FAPI 2.0

oauth.secworkshop.events

OAuth Security Workshop,
February 26-28, 2025, Reykjavik

Daniel Fett
SPRIN-D
mail@danielfett.de

Thank you!