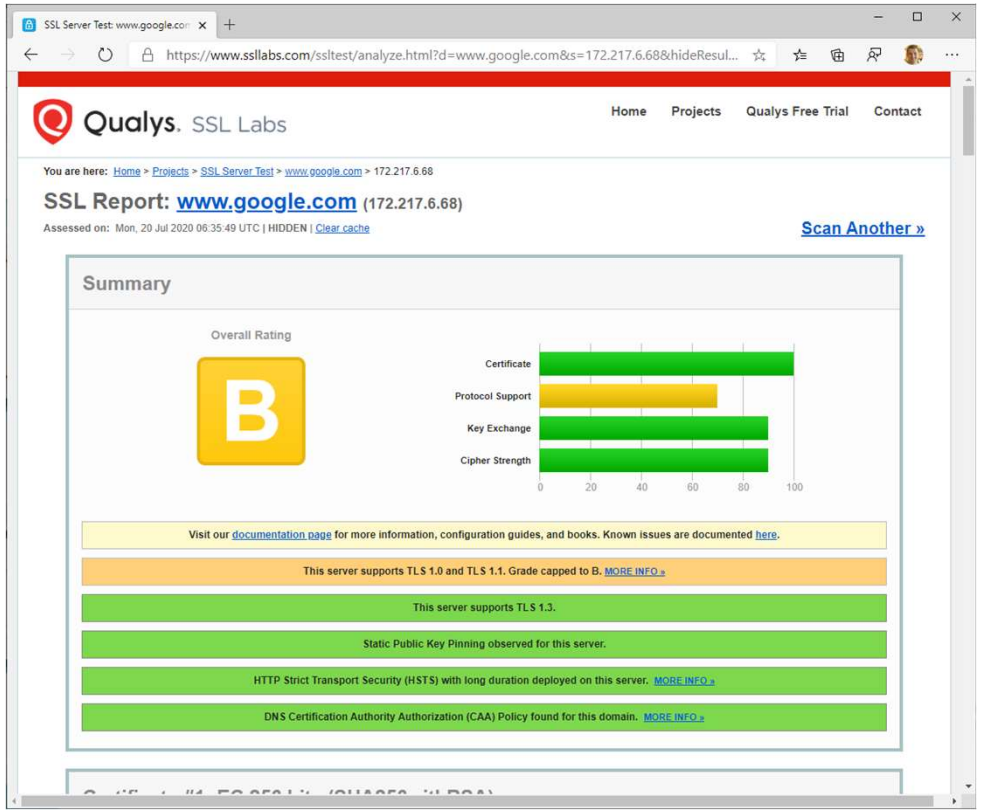
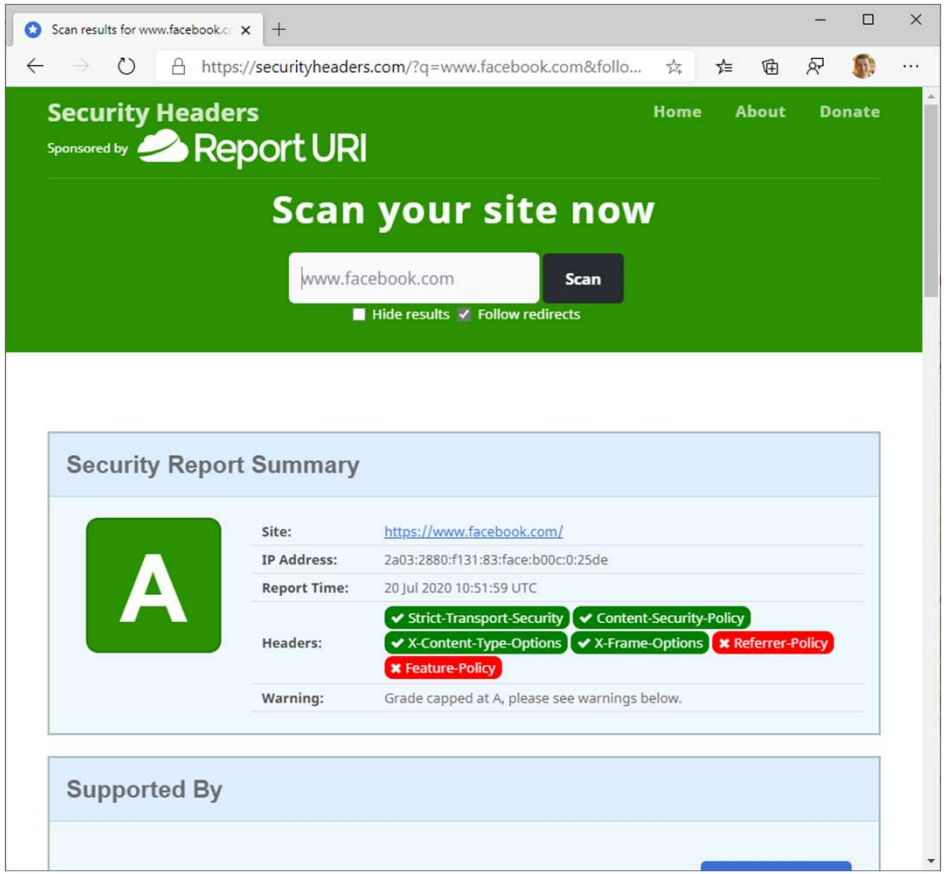
OAuch

Analyzing the Security Best Practices in the OAuth 2.0 Ecosystem

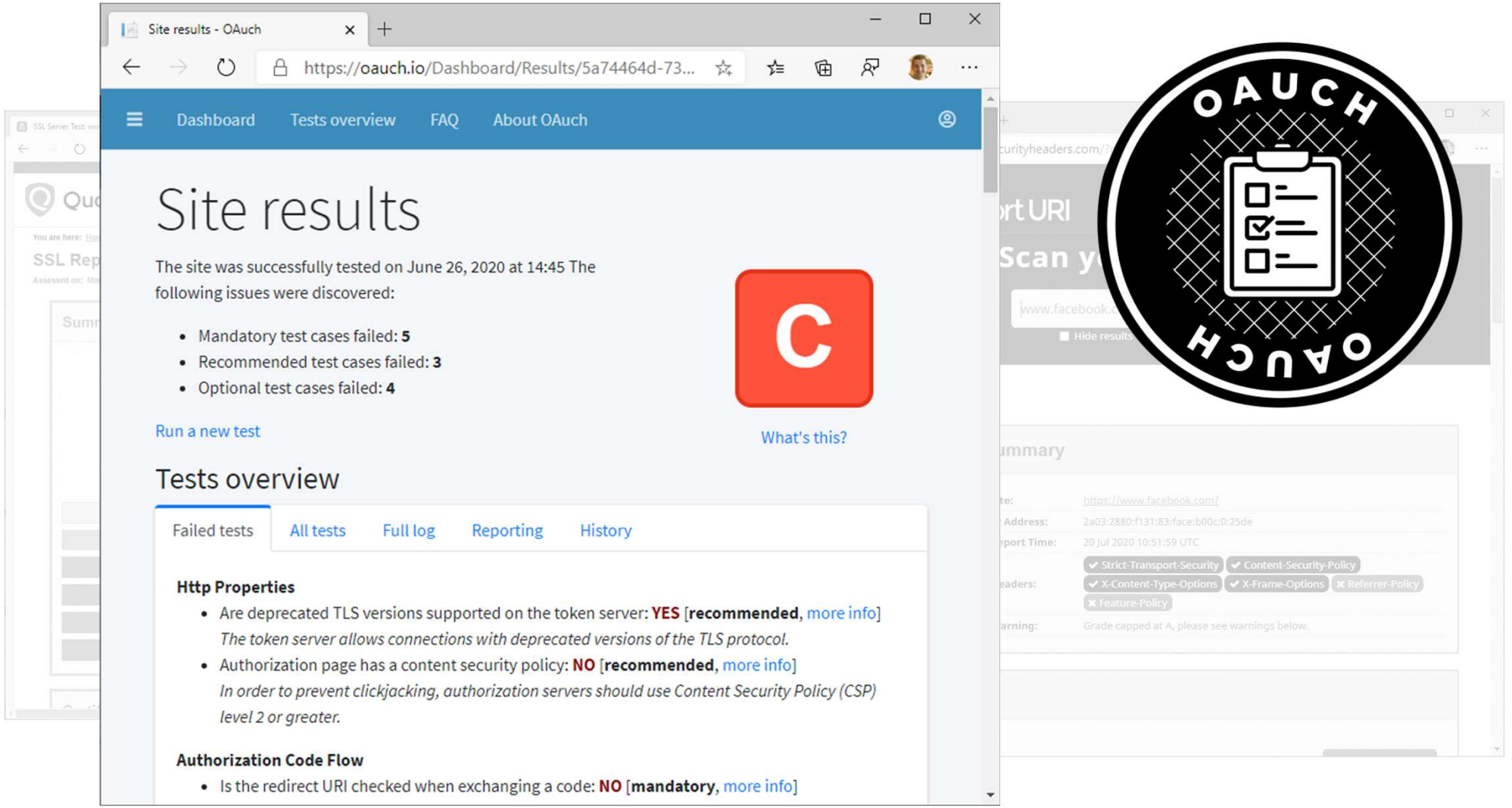
Pieter Philippaerts

“Once you have implemented OAuth2, how do you know you have implemented it securely?”

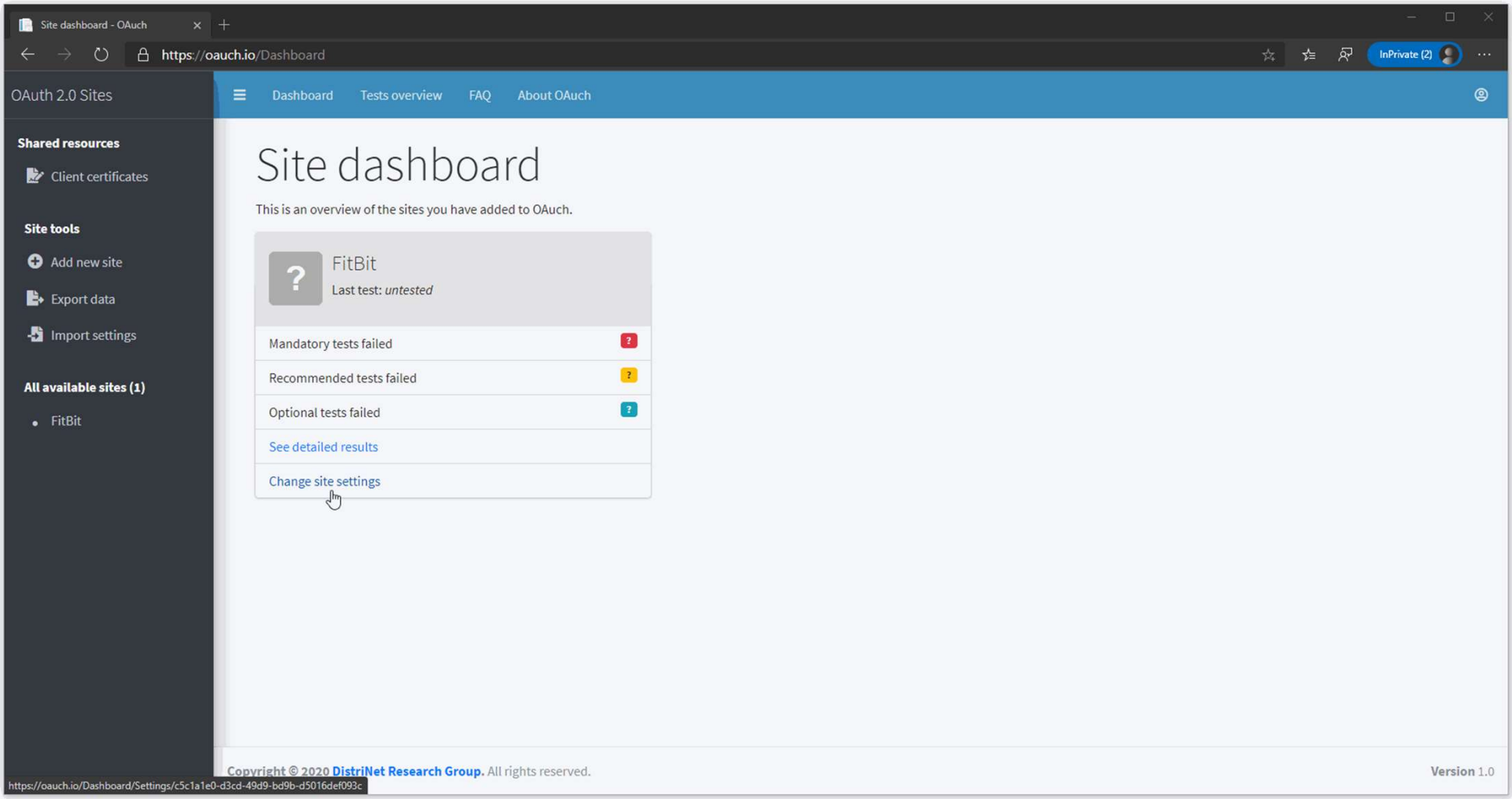




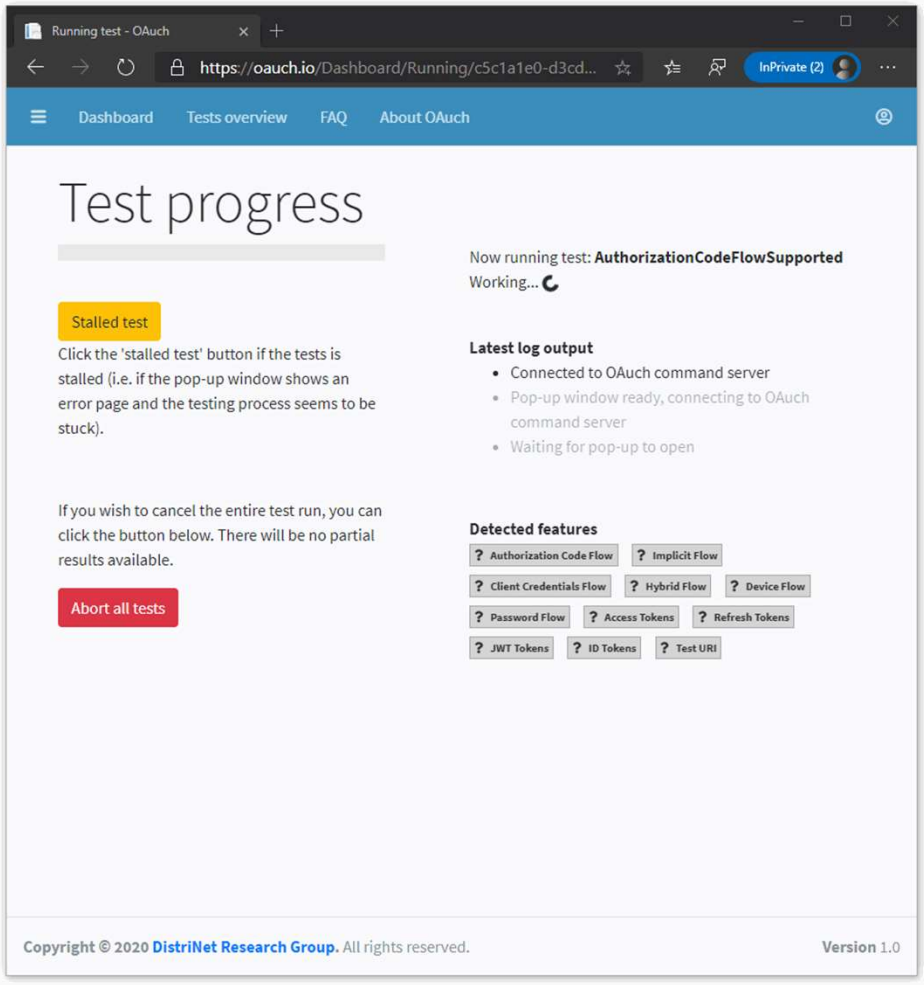
KU LEUVEN DistriNet



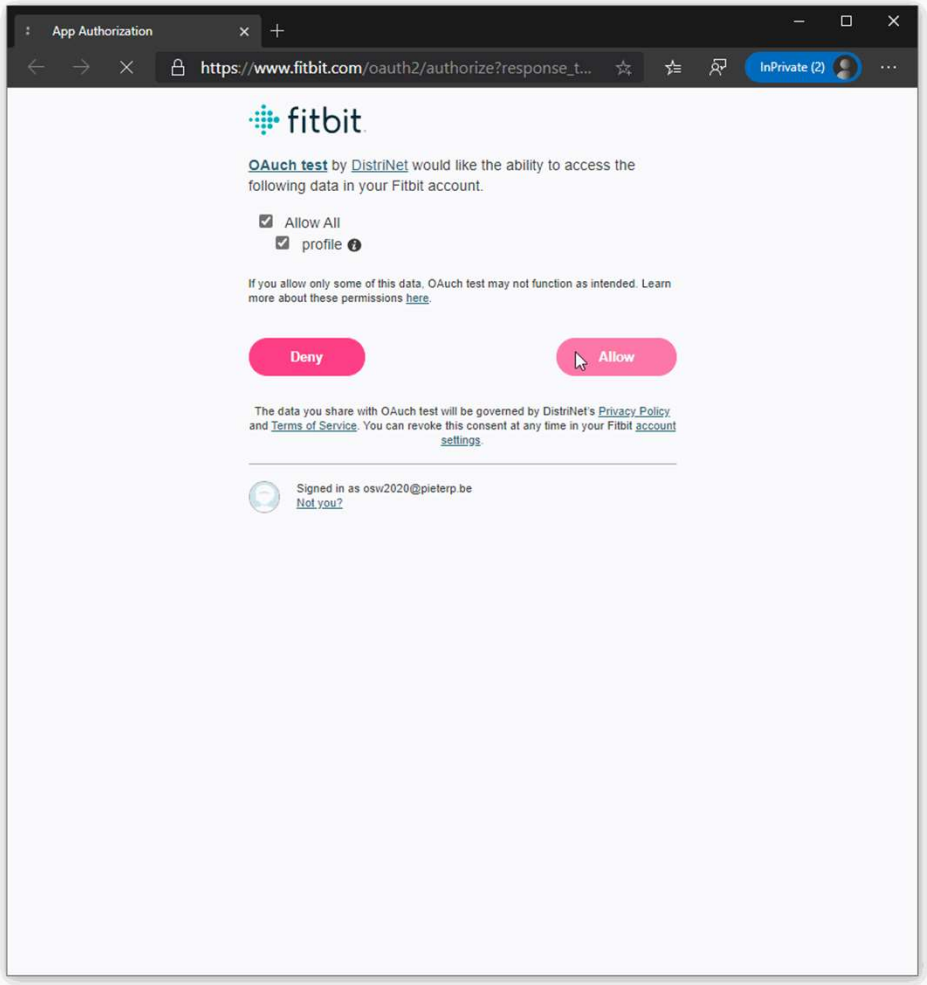
KU LEUVEN DistriNet



KU LEUVEN DistriNet

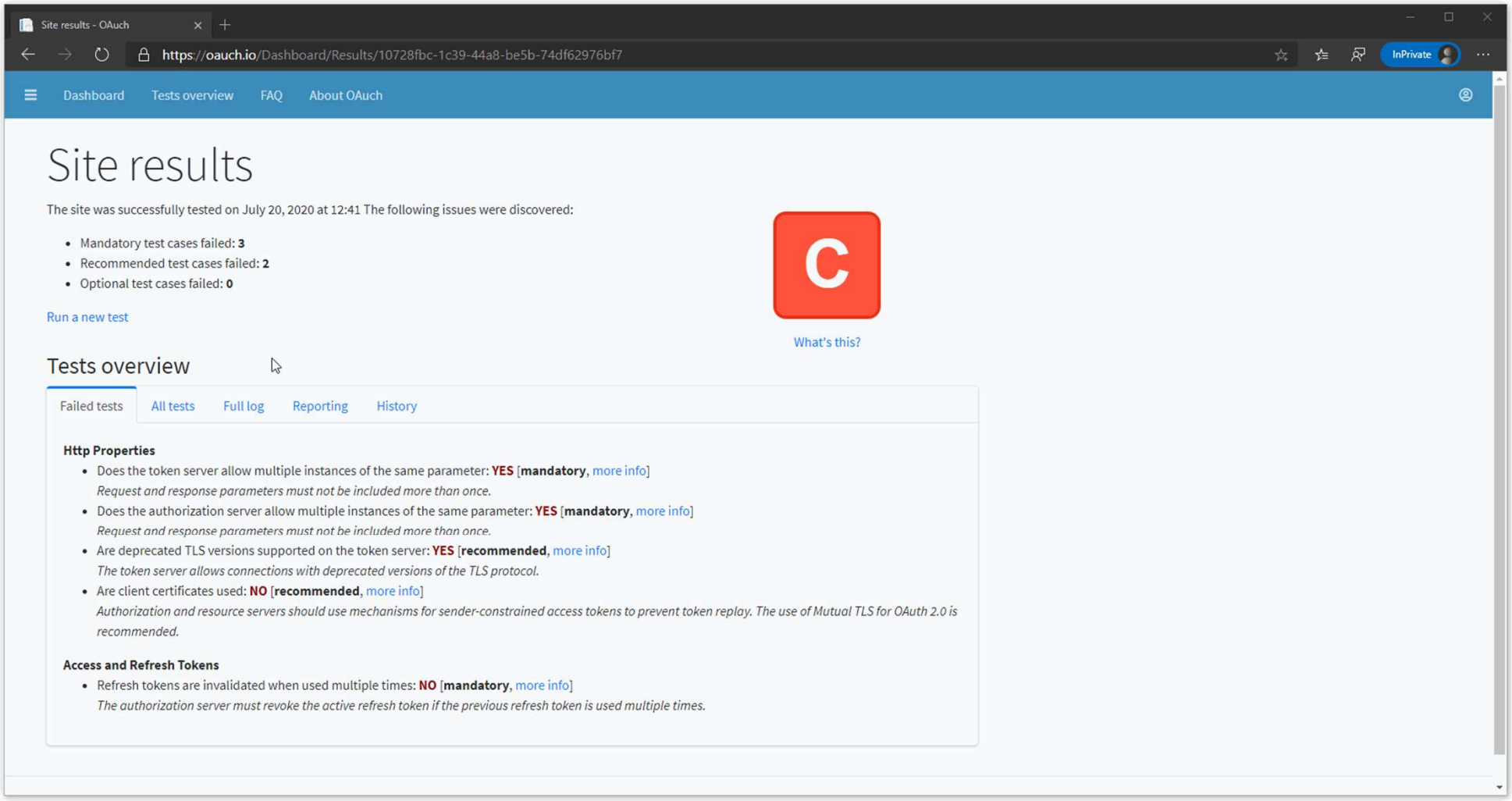


Test Overview



Authorization and Callback Window

KU LEUVEN DistriNet



KU LEUVEN DistriNet

Analyzing the OAuth 2.0 Ecosystem

What we did

- We tested 100+ OAuth implementations
  - 94 deployed and publicly available services
  - 17 OIDC providers, 77 OAuth 2.0 API providers
- We drew statistics over the sites and over the individual countermeasures

Supported Flows

API Providers

- 94% support Authorization Code flow
- 44% support Implicit flow
- 30% support Client Credentials flow
- 3% support Password flow

OIDC Providers

- 100% support Authorization Code flow
- 35% support Client Credentials flow
- 24% support Implicit flow
- 24% support Hybrid flow
- 6% support Device flow

Failure Rates

API Providers

- 38.0% average failure rate (±6.9%)
  - 31% *must* failures
  - 40% *should* failures
  - 85% *may* failures

OIDC Providers

- 28.0% average failure rate (±7.0%)
  - 22% *must* failures

Client Authentication

Client Type

- 1% support only public clients
- 1% support confidential clients (crypto key)
- 98% support confidential client (password)
  - However, 12% do not use/require the password

Client Authentication

Authorization servers must support the *Authorization* header

- Support is mandatory, but only 69% support it
- Other sites use form POST

Proof Key for Code Exchange

Authorization servers must support PKCE

- Only 12% of API providers support PKCE
  - Mostly ignored
  - Sometimes disallowed

Proof Key for Code Exchange

For the API providers supporting PKCE:

- None required PKCE
- 33% supported *plain* PKCE
- 44% allowed very short verifiers
- 56% were vulnerable to PKCE sidestep attack[1]

[1] https://mailarchive.ietf.org/arch/msg/oauth/qrLAf3nWRt8HAFkO49qGrPRuelo/

Redirect URI Matching

Callback URIs must be precisely matched

- Only 48% of sites do this

Token endpoint must compare the callback URI with the one received in the authorization request

- Only 43% of sites do this

Authorization Codes

Authorization codes must only be used once

- 76% disallow code exchange
- 12% disallow code exchange and revoke previously granted access tokens
- 12% allow multiple code exchanges

Access Tokens

- Are mostly opaque (only 15% JWT)
- Are long (85% over 128 bits of entropy)
- Can often be used as URI query parameter (44%)

Refresh Tokens

- Are used by 66% of sites

- When *refresh token rotation* is used, refresh tokens must be single use
  - Of these sites, only 34% prohibited exchanging the same refresh token multiple times
  - Active refresh tokens were never revoked

Access Tokens and Refresh Tokens

If refresh tokens are used, access token lifetime should be short

- < 1 hour: 36%
- < 8 hours and > 1 hour: 27%
- < 24 hours and > 8 hours: 10%
- > 24 hours: 27%

Some of the other results

- 26% allow authorization pages to be framed *(mandatory)*
- 29% allow the caching of sensitive values *(mandatory)*
- 70% do not suppress the referrer header *(optional)*
- 94% do not support *form post response mode* *(optional)*
- 85% allow parameters to be included multiple times *(mandatory)*
- 60% of OIDC servers do not support POST authorization requests *(mandatory)*
- 50% of OIDC servers did not require a *nonce* for the implicit flow *(mandatory)*
- 83% do not support token revocation *(optional)*
  - Of those that did, 42% accept revoked refresh tokens *(mandatory)*
- …

KU LEUVEN DistriNet

Work in progress…

- These results are a work-in-progress
  - The full analysis will hopefully be published soon

- The OAuch tool will be available at https://oauch.io/ (early September)
  - Offline download by the end of the year

KU LEUVEN DistriNet

Conclusions

- Having a formal verification of the OAuth2 protocol is great (and necessary)!
  - … but we also need tools to verify practical implementations
- A lot of sites can benefit from implementing missing countermeasures

Thank you!

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

Pieter.Philippaerts@kuleuven.be