**ISO #####-#:####(X)**

ISO TC ###/SC ##/WG #

Secretariat: XXXX

**Information technology — FAPI 2.0 Security Profile**

PAS Submission

© ISO 2024

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO’s member body in the country of the requester.

ISO copyright office

CP 401 • Ch. de Blandonnet 8

CH-1214 Vernier, Geneva

Phone: +41 22 749 01 11

Email: copyright@iso.org

Website: www.iso.org

Published in Switzerland

|  |  |
| --- | --- |
| Workgroup:  Internet-Draft:  Published:  Intended Status:  Author: | fapi  fapi-2\_0-security-profile-03  7 December 2022  Standards Track  D. Fett  *yes.com* |

### FAPI 2.0 Security Profile

### Foreword

The OpenID Foundation (OIDF) promotes, protects and nurtures the OpenID community and technologies. As a non-profit international standardizing body, it is comprised by over 160 participating entities (workgroup participant). The work of preparing implementer drafts and final international standards is carried out through OIDF workgroups in accordance with the OpenID Process. Participants interested in a subject for which a workgroup has been established have the right to be represented in that workgroup. International organizations, governmental and non-governmental, in liaison with OIDF, also take part in the work. OIDF collaborates closely with other standardizing bodies in the related fields.

Final drafts adopted by the Workgroup through consensus are circulated publicly for the public review for 60 days and for the OIDF members for voting. Publication as an OIDF Standard requires approval by at least 50% of the members casting a vote. There is a possibility that some of the elements of this document may be the subject to patent rights. OIDF shall not be held responsible for identifying any or all such patent rights.

### Introduction

The FAPI 2.0 Security Profile is an API security profile based on the OAuth 2.0 Authorization Framework [[RFC6749](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC6749)] and related specifications suitable for protecting APIs in high-value scenarios. While the security profile was initially developed with a focus on financial applications, it is designed to be universally applicable for protecting APIs exposing high-value and sensitive (personal and other) data, for example, in e-health and e-government applications.

### Warning

This document is not an OIDF International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

### Notational Conventions

The keywords "shall", "shall not", "should", "should not", "may", and "can" in this document are to be interpreted as described in ISO Directive Part 2 [[ISODIR2](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#ISODIR2)]. These keywords are not used as dictionary terms such that any occurrence of them shall be interpreted as keywords and are not to be interpreted with their natural language meanings.

### Table of Contents

* [1.  Scope](#_1._Scope)
* [2.  Normative references](#_2._Normative_references)
* [3.  Terms and definitions](#_3._Terms_and)
* [4.  Symbols and Abbreviated terms](#_4._Symbols_and)
* [5.  Security Profile](#_5._Security_Profile)
  + [5.1.  Introduction](#_5.1._Introduction)
    - [5.1.1.  Profiling this specification](#_5.1.1._Profiling_this)
  + [5.2.  Network Layer Protections](#_5.2._Network_Layer)
    - [5.2.1.  Requirements for all endpoints](#_5.2.1._Requirements_for)
    - [5.2.2.  Requirements for endpoints not used by web browsers](#_5.2.2._Requirements_for)
    - [5.2.3.  Requirements for endpoints used by web browsers](#_5.2.3._Requirements_for)
  + [5.3.  Profile](#_5.3._Profile)
    - [5.3.1.  Requirements for Authorization Servers](#_5.3.1._Requirements_for)
    - [5.3.2.  Requirements for Clients](#_5.3.2._Requirements_for)
    - [5.3.3.  Requirements for Resource Servers](#_5.3.3._Requirements_for)
  + [5.4.  Cryptography and Secrets](#_5.4._Cryptography_and)
  + [5.5.  Main Differences to FAPI 1.0](#_5.5._Main_Differences)
  + [5.6.  Security Considerations](#_5.6._Security_Considerations)
    - [5.6.1.  Access token lifetimes](#_5.6.1._Access_token)
    - [5.6.2.  DPoP Proof Replay](#_5.6.2._DPoP_Proof)
    - [5.6.3.  JWKS URIs](#_5.6.3._JWKS_URIs)
    - [5.6.4.  Duplicate Key Identifiers](#_5.6.4._Duplicate_Key)
    - [5.6.5. Injection of stolen access tokens](#_5.6.5._Injection_of)
    - [5.6.6.  Authorization Request Leaks lead to CSRF](#_5.6.6._Authorization_Request)
    - [5.6.7.  Browser-Swapping Attacks](#_5.6.7._Browser-Swapping_Attacks)
* [6.  Privacy considerations](#_6._Privacy_considerations)
* [7.  Acknowledgements](#_7._Acknowledgements)
* [8.  Normative References](#_8._Normative_References)
* [9.  Informative References](#_9._Informative_References)
* [Appendix A.  Notices](#_Appendix_A._Notices)
* [Author's Address](#_Author's_Address)

### 1. Scope

This document specifies the requirements for confidential Clients to securely obtain OAuth tokens from Authorization Servers and securely use those tokens to access REST APIs at Resource Servers.

### 2. Normative references

See section 8 for normative references.

### 3. Terms and definitions

For the purpose of this document, the terms defined in [[RFC6749](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC6749)], [[RFC6750](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC6750)], [[RFC7636](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC7636)], [[OIDC](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#OIDC)] and ISO29100 apply.

### 4. Symbols and Abbreviated terms

**API** - Application Programming Interface

**HTTP** - Hyper Text Transfer Protocol

**REST** - Representational State Transfer

**TLS** - Transport Layer Security

**DNS** - Domain Name System

**DNSSEC** - Domain Name System Security Extensions

**CAA** - Certificate Authority Authorization

**URI** - Uniform Resource Identifier

### 5. Security Profile

### 5.1. Introduction

The FAPI 2.0 Security Profile is an API security profile based on the OAuth 2.0 Authorization Framework [[RFC6749](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC6749)], that aims to reach the security goals laid out in the Attacker Model [[attackermodel](https://openid.net/specs/fapi-2_0-security-profile-ID2.html" \l "attackermodel)].

This profile is the base of the FAPI 2.0 Framework. Other specifications that are part of this framework and may be used together with this profile include:

1. FAPI Message Signing is recommended when messages are required to be signed for the purposes of non-repudiation.
2. FAPI CIBA is recommended when support is required for decoupled or cross device flows.
3. Grant Management is recommended for ecosystems that require interopable grant management.
4. OAuth 2.0 Rich Authorization Requests (RAR) [[I-D.ietf-oauth-rar](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#I-D.ietf-oauth-rar)] is recommended when the scope parameter is not expressive enough to convey the authorization that a client wants to obtain.

We are not currently aware of any mechanisms that would allow public clients to be secured to the same degree and hence their use is not within the scope of this specification.

### 5.1.1. Profiling this specification

This specification is a general purpose high security profile of OAuth 2.0 that has been proved by formal analysis to meet the stated attacker model.

This specification, and the underlying specifications, leave a number of choices open to implementors, deployers and/or ecosystems - with knowledge of the exact use cases, further reducing the number of choices may further improve security, or make implementation or interoperability easier.

However, for a profile to be compliant with this specification, the profile shall not remove or override mandatory behaviours - as doing so is likely to invalidate the formal security analysis and reduce security in potentially unpredictable ways.

### 5.2. Network Layer Protections

### 5.2.1. Requirements for all endpoints

TLS connections shall be protected against network attackers. To this end, clients, authorization servers, and resource servers:

1. shall only offer TLS protected endpoints and shall establish connections to other servers using TLS. TLS connections shall be set up to use TLS version 1.2 or later.
2. when using TLS 1.2, follow the recommendations for Secure Use of Transport Layer Security in [[RFC7525](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC7525)].
3. should use DNSSEC to protect against DNS spoofing attacks that can lead to the issuance of rogue domain-validated TLS certificates.
4. shall perform a TLS server certificate check, as per [[RFC6125](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC6125)].

**NOTE**: Even if an endpoint uses only organization validated (OV) or extended validation (EV) TLS certificates, rogue domain-validated certificates can be used to impersonate the endpoints and conduct man-in-the-middle attacks. CAA records [[RFC8659](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC8659)] can help to mitigate this risk.

### 5.2.2. Requirements for endpoints not used by web browsers

1. when using TLS 1.2, the server shall only permit the cipher suites listed in [Section 5.2.2.1](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#tls-12-ciphers)
2. when using TLS 1.2, the client should only permit the cipher suites listed in [Section 5.2.2.1](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#tls-12-ciphers)
3. When using the TLS\_DHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256 or TLS\_DHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384 cipher suites, key lengths of at least 2048 bits are required.

### 5.2.2.1. TLS 1.2 permitted cipher suites

* TLS\_DHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256
* TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256
* TLS\_DHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384
* TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384

### 5.2.3. Requirements for endpoints used by web browsers

Endpoints for the use by web browsers

1. shall use methods to ensure that connections cannot be downgraded using TLS Stripping attacks. A preloaded [[preload](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#preload)] HTTP Strict Transport Security policy [[RFC6797](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC6797)] can be used for this purpose. Some top-level domains, like .bank and .insurance, have set such a policy and therefore protect all second-level domains below them.
2. when using TLS 1.2, shall only use cipher suites allowed in [[RFC7525](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC7525)]

### 5.3. Profile

In the following, a profile of the following technologies is defined:

* OAuth 2.0 Authorization Framework [[RFC6749](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC6749)]
* OAuth 2.0 Bearer Tokens [[RFC6750](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC6750)]
* Proof Key for Code Exchange by OAuth Public Clients (PKCE) [[RFC7636](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC7636)]
* OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens (MTLS) [[RFC8705](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC8705)]
* OAuth 2.0 Demonstrating Proof-of-Possession at the Application Layer (DPoP) [[I-D.ietf-oauth-dpop](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#I-D.ietf-oauth-dpop)]
* OAuth 2.0 Pushed Authorization Requests (PAR) [[RFC9126](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC9126)]
* OAuth 2.0 Authorization Server Metadata [[RFC8414](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC8414)]
* OAuth 2.0 Authorization Server Issuer Identification [[RFC9207](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC9207)]
* OpenID Connect Core 1.0 incorporating errata set 1 [[OIDC](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#OIDC)]

### 5.3.1. Requirements for Authorization Servers

### 5.3.1.1. General Requirements

Authorization servers

1. shall distribute discovery metadata (such as the authorization endpoint) via the metadata document as specified in [[OIDD](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#OIDD)] and [[RFC8414](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC8414)]
2. shall reject requests using the resource owner password credentials grant or the implicit grant described in [[RFC6749](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC6749)] or the hybrid flow as described in [[OIDC](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#OIDC)]
3. shall support confidential clients as defined in [[RFC6749](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC6749)]
4. shall only issue sender-constrained access tokens,
5. shall use one of the following methods for sender-constrained access tokens:
   * MTLS as described in [[RFC8705](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC8705)]
   * DPoP as described in [[I-D.ietf-oauth-dpop](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#I-D.ietf-oauth-dpop)]
6. shall authenticate clients using one of the following methods:
   * MTLS as specified in section 2 of [[RFC8705](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC8705)]
   * private\_key\_jwt as specified in section 9 of [[OIDC](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#OIDC)]
7. shall not expose open redirectors (see section 4.10 of [[I-D.ietf-oauth-security-topics](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#I-D.ietf-oauth-security-topics)])
8. shall accept its issuer identifier value (as defined in [[RFC8414](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC8414)]) in the aud claim received in client authentication assertions.
9. shall not use refresh token rotation unless, in the case a response with a new refresh token is not received and stored by the client, retrying the request (with the previous refresh token) will succeed.
10. if using DPoP, may use the server provided nonce mechanism (as defined in section 8 of [[I-D.ietf-oauth-dpop](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#I-D.ietf-oauth-dpop)]).
11. shall issue authorization codes with a maximum lifetime of 60 seconds
12. if using DPoP, shall support "Authorization Code Binding to DPoP Key" (as required by section 10.1 of [[I-D.ietf-oauth-dpop](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#I-D.ietf-oauth-dpop)]).

**NOTE**: In order to facilitate interoperability the authorization server should also accept its token endpoint URL or the URL of the endpoint at which the assertion was received in the aud claim received in client authentication assertions.

**NOTE**: Refresh token rotation is an optional feature defined in [[RFC6749](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC6749)] section 6 where the Authorization Server issues a new refresh token to the client as part of the refresh\_token grant. This specification discourages the use of this feature as it doesn't bring any security benefits for confidential clients, and can cause significant operational issues. However to allow for operational agility, Authorization Servers may implement it providing they meet the requirement in clause 20.

**NOTE**: Other grants as appropriate may be supported, for example the client credentials grant, the Client Initiated Backchannel Authentication grant, etc.

### 5.3.1.2. Authorization Code Flow

For the Authorization Code flow, Authorization servers

1. shall support the authorization code grant (response\_type=code & grant\_type=authorization\_code) described in [[RFC6749](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC6749)]
2. shall support client-authenticated pushed authorization requests according to [[RFC9126](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC9126)]
3. shall reject authorization requests sent without [[RFC9126](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC9126)]
4. shall reject pushed authorization requests without client authentication
5. shall require PKCE [[RFC7636](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC7636)] with S256 as the code challenge method
6. shall require the redirect\_uri parameter in pushed authorization requests
7. shall return an iss parameter in the authorization response according to [[RFC9207](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC9207)]
8. shall not transmit authorization responses over unencrypted network connections, and, to this end, shall not allow redirect URIs that use the "http" scheme except for native clients that use Loopback Interface Redirection as described in [[RFC8252](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC8252)], Section 7.3,
9. shall reject an authorization code (section 1.3.1 of [[RFC6749](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC6749)]) if it has been previously used
10. shall not use the HTTP 307 status code when redirecting a request that contains user credentials to avoid forwarding the credentials to a third party accidentally (see section 4.11 of [I-D.ietf-oauth-security-topics]);
11. should use the HTTP 303 status code when redirecting the user agent using status codes;
12. shall issue pushed authorization requests request\_uri with expires\_in values of less than 600 seconds.

**NOTE**: If replay identification of the authorization code is not possible, it is desirable to set the validity period of the authorization code to one minute or a suitable short period of time. The validity period may act as a cache control indicator of when to clear the authorization code cache if one is used

**NOTE**: The request\_uri expires\_in time must be sufficient for the user's device to receive the link and the user to complete the process of opening the link. In many cases (poor network connection or where the user has to manually select the browser to be used) this can easily take over 30 seconds.

### 5.3.1.3. Returning Authenticated User's Identifier

If it is desired to provide the authenticated user's identifier to the client in the token response, the authorization server shall support OpenID Connect [[OIDC](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#OIDC)].

### 5.3.2. Requirements for Clients

### 5.3.2.1. General Requirements

Clients

1. shall support sender-constrained access tokens using one of the following methods:
   * MTLS as described in [[RFC8705](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC8705)]
   * DPoP as described in [[I-D.ietf-oauth-dpop](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#I-D.ietf-oauth-dpop)]
2. shall support client authentication using one of the following methods:
   * MTLS as specified in section 2 of [[RFC8705](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC8705)]
   * private\_key\_jwt as specified in section 9 of [[OIDC](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#OIDC)]
3. shall send access tokens in the HTTP header as in Section 2.1 of OAuth 2.0 Bearer Token Usage [[RFC6750](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC6750)]
4. shall not expose open redirectors (see section 4.10 of [[I-D.ietf-oauth-security-topics](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#I-D.ietf-oauth-security-topics)])
5. if using private\_key\_jwt, shall use the Authorization Server's issuer identifier value (as defined in [[RFC8414](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC8414)]) in the aud claim sent in client authentication assertions. The issuer identifier value shall be sent as a string not as an item in an array.
6. shall support refresh tokens and their rotation
7. if using MTLS client authentication or MTLS sender-constrained access tokens, shall support the mtls\_endpoint\_aliases metadata defined in [[RFC8705](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC8705)]
8. if using DPoP, shall support the server provided nonce mechanism (as defined in section 8 of [[I-D.ietf-oauth-dpop](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#I-D.ietf-oauth-dpop)])
9. shall only use authorization server metadata (such as the authorization endpoint) retrieved from the metadata document as specified in [[OIDD](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#OIDD)] and [[RFC8414](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC8414)]
10. shall ensure that the issuer URL used as the basis for retrieving the authorization server metadata is obtained from an authoritative source and using a secure channel, such that it cannot be modified by an attacker
11. shall ensure that this issuer URL and the issuer value in the obtained metadata match

**NOTE:**

This profile may be used by Confidential Clients on a user-controlled device where the system clock may not be accurate, this may cause private\_key\_jwt client authentication to fail. In such circumstances a Client should consider using the HTTP Date header returned from the server to synchronise it's own clock when generating client assertions.

**NOTE:**

Although Authorization Servers are required to support "Authorization Code Binding to DPoP Key" (as defined by section 10.1 of [[I-D.ietf-oauth-dpop](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#I-D.ietf-oauth-dpop)]), clients are not required to use it.

### 5.3.2.2. Authorization Code Flow

For the Authorization Code flow, Clients

1. shall use the authorization code grant described in [[RFC6749](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC6749)]
2. shall use pushed authorization requests according to [[RFC9126](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC9126)]
3. shall use PKCE [[RFC7636](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC7636)] with S256 as the code challenge method
4. shall check the iss parameter in the authorization response according to [[RFC9207](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC9207)] to prevent Mix-Up attacks

### 5.3.3. Requirements for Resource Servers

The FAPI 2.0 endpoints are OAuth 2.0 protected resource endpoints that return protected information for the resource owner associated with the submitted access token.

Resource servers with the FAPI endpoints

1. shall accept access tokens in the HTTP header as in Section 2.1 of OAuth 2.0 Bearer Token Usage [[RFC6750](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC6750)]
2. shall not accept access tokens in the query parameters stated in Section 2.3 of OAuth 2.0 Bearer Token Usage [[RFC6750](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC6750)]
3. shall verify the validity, integrity, expiration and revocation status of access tokens
4. shall verify that the authorization represented by the access token is sufficient for the requested resource access and otherwise return errors as in section 3.1 of [[RFC6750](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC6750)]
5. shall support and verify sender-constrained access tokens using one of the following methods:
   * MTLS as described in [[RFC8705](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC8705)]
   * DPoP as described in [[I-D.ietf-oauth-dpop](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#I-D.ietf-oauth-dpop)]

### 5.4. Cryptography and Secrets

1. Authorization Servers, Clients, and Resource Servers when creating or processing JWTs shall
   1. adhere to [[RFC8725](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC8725)]
   2. use PS256, ES256, or EdDSA (using the Ed25519 subtype) algorithms
   3. not use or accept the none algorithm
2. RSA keys shall have a minimum length of 2048 bits.
3. Elliptic curve keys shall have a minimum length of 160 bits.
4. Credentials not intended for handling by end-users (e.g., access tokens, refresh tokens, authorization codes, etc.) shall be created with at least 128 bits of entropy such that an attacker correctly guessing the value is computationally infeasible. Cf. Section 10.10 of [[RFC6749](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC6749)].

### 5.5. Main Differences to FAPI 1.0

| *Table 1* | | |
| --- | --- | --- |
| **FAPI 1.0 Read/Write** | **FAPI 2.0** | **Reasons** |
| JAR | PAR | integrity protection and compatibility improvements for authorization requests |
| JARM | only code in response | the authorization response is reduced to only contain the authorization code, obsoleting the need for integrity protection |
| BCM principles, defenses based on particular threats | attacker model, security goals, best practices from the OAuth Security BCP | clearer design guideline, suitability for formal analysis |
| s\_hash | PKCE | protection provided by state (in particular against CSRF) is now provided by PKCE; state integrity is partially protected by PAR |
| pre-registered redirect URIs | redirect URIs in PAR | pre-registration is not required with client authentication and PAR |
| response types code id\_token or code | response type code | no ID token in front-channel (privacy improvement); nonce/signature check can be skipped by clients, PKCE cannot (security improvement) |
| ID Token as detached signature | PKCE | ID token does not need to serve as a detached signature |
| potentially encrypted ID Tokens in the front channel | No encryption and no ID Tokens in the front channel | ID Tokens only exchanged in back channel |
| nbf & exp claims in request object | request\_uri has limited lifetime | Prevents pre-generation of requests |
| x-fapi-\* headers | Moved to Implementation and Deployment Advice document | Not relevant to the core of the security profile |
| MTLS for sender-constrained access tokens | MTLS or DPoP | Due to the lack of the tight integration with the TLS layer, DPoP can be easier to deploy in some scenarios |

### 5.6. Security Considerations

### 5.6.1. Access token lifetimes

1. The use of short lived access tokens (combined with refresh tokens) potentially reduces the time window for some attacks.
2. The use of refresh tokens also allows clients to rotate their sender-constraining keys without loss of grants, either because of compromise of the key or as part of good security hygiene.
3. If issuing long-lived grants (e.g. days/weeks), the use of short lived (e.g. minutes/hours) access tokens combined with refresh tokens should be considered.
4. There is a performance and resiliency trade off, setting the access token life time too short can increase the load on and dependency on the authorization server.

### 5.6.2. DPoP Proof Replay

An attacker of type A7 (see [[attackermodel](https://openid.net/specs/fapi-2_0-security-profile-ID2.html" \l "attackermodel)]) may be able to obtain DPoP proofs that they can then replay.

This may also allow reuse of the DPoP proof with an altered request, as DPoP does not sign the body of HTTP requests nor most headers. For example, for a payment request the attacker might be able to specify a different amount or destination account.

Possible mitigations for this are:

1. Resource servers uses short-lived DPoP nonces to reduce the time window where a request can be replayed
2. Resource servers implement replay prevention using the jti header as explained in [[I-D.ietf-oauth-dpop](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#I-D.ietf-oauth-dpop)]
3. Replay of an altered request can be prevented by using signed resource requests as per FAPI Message Signing
4. Consider MTLS sender-constraining instead of DPoP
5. These mitigations may have potential complexity, performance or scalability tradeoffs. Attacker type A7 represents a powerful attacker and mitigations may not be necessary for many ecosystems.

### 5.6.3. JWKS URIs

This profile supports the use of private\_key\_jwt and in addition allows the use of OpenID Connect. When these are used Clients and Authorization Servers need to verify payloads with keys from another party. For Authorization Servers this profile strongly recommends the use of JWKS URI endpoints to distribute public keys. For Client's key management this profile recommends either the use of JWKS URI endpoints or the use of the jwks parameter in combination with [[RFC7591](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC7591)] and [[RFC7592](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC7592)].

The definition of the Authorization Server jwks\_uri can be found in [[RFC8414](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC8414)], while the definition of the Client jwks\_uri can be found in [[RFC7591](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC7591)].

In addition, this profile

1. requires that jwks\_uri endpoints shall be served over TLS;
2. recommends that JOSE headers for x5u and jku should not be used; and
3. recommends that the JWK set does not contain multiple keys with the same kid.

### 5.6.4. Duplicate Key Identifiers

JWK sets should not contain multiple keys with the same kid. However, to increase interoperability when there are multiple keys with the same kid, the verifier shall consider other JWK attributes, such as kty, use, alg, etc., when selecting the verification key for the particular JWS message. For example, the following algorithm could be used in selecting which key to use to verify a message signature:

1. find keys with a kid that matches the kid in the JOSE header;
2. if a single key is found, use that key;
3. if multiple keys are found, then the verifier should iterate through the keys until a key is found that has a matching alg, use, kty, or crv that corresponds to the message being verified.

### 5.6.5. Injection of stolen access tokens

There are potential situations where the attacker may be able to inject stolen access tokens into a client to bypass [[RFC8705](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#RFC8705)] or [[I-D.ietf-oauth-dpop](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#I-D.ietf-oauth-dpop)] sender-constraining of the access token, as described in "Cuckoo's Token Attack" in [[FAPI1SEC](https://openid.net/specs/fapi-2_0-security-profile-ID2.html#FAPI1SEC)].

A pre-condition for this attack is that the attacker has control of an authorization server that is trusted by the client to issue access tokens for the target resource server. An attacker may obtain control of an authorization server by:

1. Compromising the security of a different authorization server that the client trusts, or
2. Acting as an authorization server and establishing a trust relationship with a client using social engineering, or by
3. compromising the client

The attack may be easier if a centralized directory or other resource server discovery mechanism allows the attacker to cause the client to send the stolen access token received from the attacker controlled Authorization Server to an honest Resource Server.

The pre-conditions for this attack do not apply to many ecosystems and require a powerful attacker. In situations where the pre-conditions may be met, the possible mitigations include:

1. Clients using different DPoP keys or MTLS certificates at each authorization server
2. Clients sending the issuer identifier the access token was obtained from to the resource server, and requiring resource servers to verify the issuer matches the authorization server that originally issued the token (though there is no standardized method for clients to send the issuer to the resource server)
3. Reducing the time window for the attack by using short lived access tokens alongside refresh tokens

### 5.6.6. Authorization Request Leaks lead to CSRF

An attacker of type A3a (see [[attackermodel](https://openid.net/specs/fapi-2_0-security-profile-ID2.html" \l "attackermodel)]) can intercept an authorization request, log in at the Authorization Server, receive an authorization code and redirect the honest user via a CSRF attack to the honest client but with the attacker's authorization code. This results in the user accessing the attackers resources, thus breaking session integrity.

It is important to note that all practically used redirect-based flows are susceptible to this attack, as redirection does not allow for a tight coupling of the session between the user's browser and the client on the one side and the session between the user's browser and the authorization server on the other side. This attack, however, requires a strong attacker who can read authorization requests and perform a CSRF attack in a short time window.

Possible mitigations for this are:

1. Requiring the Authorization Server to only accept a request\_uri once. This will prevent attacks where the attacker was able to read the authorization request, but not use the request\_uri before the honest user does so.
2. Requiring the Client to only make one authorization code grant call for each authorization endpoint call. This will prevent attacks where the attacker was unable to send the authorization response before the honest user does so.
3. Reducing the lifetime of the authorization code - this will reduce the window in which the CSRF attack has to be performed.

An attacker that has the option to block a user's request completely can circumvent the first and second defenses. In practice, however, attackers can often read an authorization request (e.g., from a log file or via some other side-channel), but not block the request from being sent. If the victim's internet connection is slow, this might increase the attacker's chances.

### 5.6.7. Browser-Swapping Attacks

An attacker that has access to the authorization response sent through a victim's browser can perform a browser-swapping attack as follows:

1. The attacker starts a new flow using their own browser and some client. The client sends a pushed authorization request to the authorization server and receives a request\_uri in the response. The client then redirects the attacker's browser to the authorization server.
2. The attacker intercepts this redirection and forwards the URL to a victim. For example, the attacker can embed a link to this URL in a phishing website, an email, or a QR code.
3. The victim may be tricked into believing that an authentication/authorization is legitimately required. The victim therefore authenticates at the authorization server and may grant the client access to their data.
4. The attacker can now intercept the authorization response in the victim's browser and forward it to the client using their own browser.
5. The client will recognize that the authorization response belongs to the same browser that initially started the transaction (the attacker's browser) and exchange the authorization code for an access token and/or obtain user information.
6. Via the client, the attacker now has access to the user's resources or is logged in as the user.

With currently deployed technology, there is no way to completely prevent this attack if the authorization response leaks to an attacker in any redirect-based protocol. It is therefore important to keep the authorization response confidential. The requirements in this security profile are designed to achieve that, e.g., by disallowing open redirectors and requiring that the redirect\_uri is sent via an authenticated and encrypted channel, the pushed authorization request, ensuring that the redirect\_uri cannot be manipulated by the attacker.

Implementers need to consider the confidentiality of the authorization response critical when designing their systems, in particular when this security profile is used in other contexts, e.g., mobile applications.

### 6. Privacy considerations

There are many factors to be considered in terms of privacy when implementing this specification. Since this specification is a profile of OAuth 2.0 and OpenID Connect, the privacy considerations are not specific to this document and generally apply to OAuth or OpenID Connect. Implementers are advised to perform a thorough privacy impact assessment and manage identified risks appropriately.

Note: Implementers can consult documents like [ISO29100] and [ISO29134] for this purpose.

Privacy threats to OAuth and OpenID Connect implementations include the following:

* **Inappropriate privacy notice**: A privacy notice (e.g., provided at a policy\_url) or by other means can be inappropriate or insufficient.
* **Inadequate choice**: Providing a consent screen without adequate choices does not form consent.
* **Misuse of data**: An authorization server, resource server or client can potentially use the data not according to the purpose that was agreed.
* **Collection minimization violation**: A client asking for more data than it absolutely needs to fulfill the purpose is violating the collection minimization principle.
* **Unsolicited personal data from the resource server**: Some bad resource server implementations may return more data than requested. If the data is personal data, then this would be a violation of privacy principles.
* **Data minimization violation**: Any process that is processing more data than it needs is violating the data minimization principle.
* **Authorization servers tracking end-users**: Authorization servers identifying what data is being provided to which client for which end-user.
* **End-user tracking by clients**: Two or more clients correlating access tokens or ID Tokens to track users.
* **Client misidentification by end-users**: End-user misunderstands who the client is due to a confusing representation of the client at the authorization server's authorization page.
* **Insufficient understanding of the end-user granting access to data**: To enhance the trust of the ecosystem, best practice is for the authorization server to make clear what is included in the authorization request (for example, what data will be released to the client).
* **Attacker observing personal data in authorization request/response**: The authorization request or response might contain personal data. In some jurisdictions, even security parameters can be considered personal data. This profile aims to reduce the data sent in the authorization request and response to an absolute minimum, but nonetheless, an attacker might observe some data.
* **Data leak from authorization server**: The authorization server generally stores personal data. If it becomes compromised, this data can leak or be modified.
* **Data leak from resource servers**: Some resource servers store personal data. If a resource server becomes compromised, this data can leak or be modified.
* **Data leak from clients**: Some clients store personal data. If the client becomes compromised, this data can leak or be modified.

### 7. Acknowledgements

We would like to thank Takahiko Kawasaki, Filip Skokan, Dave Tonge, Nat Sakimura, Stuart Low, Dima Postnikov, Torsten Lodderstedt, Joseph Heenan, Travis Spencer, Brian Campbell, Ralph Bragg, Lukasz Jaromin, Pedram Hosseyni, Ralf Küsters and Tim Würtele for their valuable feedback and contributions that helped to evolve this specification.

### 8. Normative References

**[RFC9126]**

Lodderstedt, T., Campbell, B., Sakimura, N., Tonge, D., and F. Skokan, "OAuth 2.0 Pushed Authorization Requests", RFC 9126, DOI 10.17487/RFC9126, September 2021, <<https://www.rfc-editor.org/info/rfc9126>>.

**[RFC9207]**

Meyer zu Selhausen, K. and D. Fett, "OAuth 2.0 Authorization Server Issuer Identification", RFC 9207, DOI 10.17487/RFC9207, March 2022, <<https://www.rfc-editor.org/info/rfc9207>>.

**[OIDD]**

Sakimura, N., Bradley, J., Jones, M., and E. Jay, "OpenID Connect Discovery 1.0 incorporating errata set 1", 8 November 2014, <<https://openid.net/specs/openid-connect-discovery-1_0.html>>.

**[OIDC]**

Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and C. Mortimore, "OpenID Connect Core 1.0 incorporating errata set 1", 8 November 2014, <<http://openid.net/specs/openid-connect-core-1_0.html>>.

**[attackermodel]**

Fett, D., "FAPI 2.0 Attacker Model", 14 November 2022, <<https://openid.net/specs/fapi-2_0-attacker-model.html>>.

**[RFC6797]**

Hodges, J., Jackson, C., and A. Barth, "HTTP Strict Transport Security (HSTS)", RFC 6797, DOI 10.17487/RFC6797, November 2012, <<https://www.rfc-editor.org/info/rfc6797>>.

**[RFC8705]**

Campbell, B., Bradley, J., Sakimura, N., and T. Lodderstedt, "OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens", RFC 8705, DOI 10.17487/RFC8705, February 2020, <<https://www.rfc-editor.org/info/rfc8705>>.

**[RFC8725]**

Sheffer, Y., Hardt, D., and M. Jones, "JSON Web Token Best Current Practices", BCP 225, RFC 8725, DOI 10.17487/RFC8725, February 2020, <<https://www.rfc-editor.org/info/rfc8725>>.

**[RFC7592]**

Richer, J., Ed., Jones, M., Bradley, J., and M. Machulak, "OAuth 2.0 Dynamic Client Registration Management Protocol", RFC 7592, DOI 10.17487/RFC7592, July 2015, <<https://www.rfc-editor.org/info/rfc7592>>.

**[RFC7636]**

Sakimura, N., Ed., Bradley, J., and N. Agarwal, "Proof Key for Code Exchange by OAuth Public Clients", RFC 7636, DOI 10.17487/RFC7636, September 2015, <<https://www.rfc-editor.org/info/rfc7636>>.

**[I-D.ietf-oauth-rar]**

Lodderstedt, T., Richer, J., and B. Campbell, "OAuth 2.0 Rich Authorization Requests", Work in Progress, Internet-Draft, draft-ietf-oauth-rar-17, 2 December 2022, <<https://www.ietf.org/archive/id/draft-ietf-oauth-rar-17.txt>>.

**[RFC6125]**

Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <<https://www.rfc-editor.org/info/rfc6125>>.

**[RFC8659]**

Hallam-Baker, P., Stradling, R., and J. Hoffman-Andrews, "DNS Certification Authority Authorization (CAA) Resource Record", RFC 8659, DOI 10.17487/RFC8659, November 2019, <<https://www.rfc-editor.org/info/rfc8659>>.

**[RFC8414]**

Jones, M., Sakimura, N., and J. Bradley, "OAuth 2.0 Authorization Server Metadata", RFC 8414, DOI 10.17487/RFC8414, June 2018, <<https://www.rfc-editor.org/info/rfc8414>>.

**[RFC6749]**

Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <<https://www.rfc-editor.org/info/rfc6749>>.

**[RFC7525]**

Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7525, DOI 10.17487/RFC7525, May 2015, <<https://www.rfc-editor.org/info/rfc7525>>.

**[RFC8252]**

Denniss, W. and J. Bradley, "OAuth 2.0 for Native Apps", BCP 212, RFC 8252, DOI 10.17487/RFC8252, October 2017, <<https://www.rfc-editor.org/info/rfc8252>>.

**[ISODIR2]**

Standardization, I. O. F., "ISO/IEC Directives Part 2 -", , <<https://www.iso.org/sites/directives/current/part2/index.xhtml>>.

**[RFC6750]**

Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, DOI 10.17487/RFC6750, October 2012, <<https://www.rfc-editor.org/info/rfc6750>>.

**[I-D.ietf-oauth-dpop]**

Fett, D., Campbell, B., Bradley, J., Lodderstedt, T., Jones, M., and D. Waite, "OAuth 2.0 Demonstrating Proof-of-Possession at the Application Layer (DPoP)", Work in Progress, Internet-Draft, draft-ietf-oauth-dpop-11, 10 August 2022, <<https://www.ietf.org/archive/id/draft-ietf-oauth-dpop-11.txt>>.

**[RFC7591]**

Richer, J., Ed., Jones, M., Bradley, J., Machulak, M., and P. Hunt, "OAuth 2.0 Dynamic Client Registration Protocol", RFC 7591, DOI 10.17487/RFC7591, July 2015, <<https://www.rfc-editor.org/info/rfc7591>>.

### 9. Informative References

**[preload]**

Anonymous, "HSTS Preload List Submission", , <<https://hstspreload.org/>>.

**[FAPI1SEC]**

Fett, D., Hosseyni, P., and R. Kuesters, "An Extensive Formal Security Analysis of the OpenID Financial-grade API", 31 January 2019, <<https://arxiv.org/abs/1901.11520>>.

**[I-D.ietf-oauth-security-topics]**

Lodderstedt, T., Bradley, J., Labunets, A., and D. Fett, "OAuth 2.0 Security Best Current Practice", Work in Progress, Internet-Draft, draft-ietf-oauth-security-topics-21, 27 September 2022, <<https://www.ietf.org/archive/id/draft-ietf-oauth-security-topics-21.txt>>.

### Appendix A. Notices

Copyright (c) 2022 The OpenID Foundation.

The OpenID Foundation (OIDF) grants to any Contributor, developer, implementer, or other interested party a non-exclusive, royalty free, worldwide copyright license to reproduce, prepare derivative works from, distribute, perform and display, this Implementers Draft or Final Specification solely for the purposes of (i) developing specifications, and (ii) implementing Implementers Drafts and Final Specifications based on such documents, provided that attribution be made to the OIDF as the source of the material, but that such attribution does not indicate an endorsement by the OIDF.

The technology described in this specification was made available from contributions from various sources, including members of the OpenID Foundation and others. Although the OpenID Foundation has taken steps to help ensure that the technology is available for distribution, it takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this specification or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any independent effort to identify any such rights. The OpenID Foundation and the contributors to this specification make no (and hereby expressly disclaim any) warranties (express, implied, or otherwise), including implied warranties of merchantability, non-infringement, fitness for a particular purpose, or title, related to this specification, and the entire risk as to implementing this specification is assumed by the implementer. The OpenID Intellectual Property Rights policy requires contributors to offer a patent promise not to assert certain patent claims against other contributors and against implementers. The OpenID Foundation invites any interested party to bring to its attention any copyrights, patents, patent applications, or other proprietary rights that may cover technology that may be required to practice this specification.

### Author's Address

**Daniel Fett**

yes.com

Email: [mail@danielfett.de](mailto:mail@danielfett.de)