FAPI 2.0 Profile Definitions

Author: Daniel Fett, yes.com, danielf@yes.com
Last updated: 2020-02-20

This document aims to define two profiles of the Financial-grade API 2.0. The profiles are defined by their security requirements, expressed through attacker models they defend against and additional non-repudiation requirements. From these requirements, the utilized security mechanisms are derived. Previous FAPI profiles were defined by the security mechanisms without stating an explicit attacker/threat model or security requirements.

See also: OAuth Attacker Model Cheat Sheet

Security Goals

FAPI aims to achieve the following security goals (with respect to the attacker models defined below):

Authorization

No attacker can access resources belonging to a user.
The access token is the ultimate credential for access to resources in OAuth. Therefore, this security goal is fulfilled if no attacker can successfully obtain and use an access token belonging to a user.

Authentication

No attacker is able to log in at a client under the identity of a user.
The ID token is the credential for authentication in OpenID Connect. This security goal therefore is fulfilled if no attacker can obtain and use an ID token carrying the identity of a user for login.

Session Integrity

Session Integrity is concerned with attacks where a user is tricked into logging in under the attacker's identity or inadvertently using the resources of the attacker instead of the user's own resources. Attacks in this field include CSRF attacks (traditionally defended against by using "state" in OAuth) and session swapping attacks.
In detail:

- **For authentication: No attacker is able to force a user to be logged in under the identity of the attacker.**
- **For authorization: No attacker is able to force a user to use resources of the attacker**

Attacker Model

Assumptions (What *is* Working?)

This attacker model assumes that certain parts of the infrastructure are working correctly. In other words, failures in these parts likely lead to attacks that are out of the scope of this attacker model and must be analyzed separately. For example, if a major flaw in TLS was found that undermines data integrity in TLS connections, a network attacker (A2, below) would be able to compromise practically all OAuth and OpenID Connect sessions in various ways. This would be fatal, as even application-level signing and encryption is based on key distribution via TLS connections.

The following parts of the infrastructure are out of the scope of this attacker model:

- **TLS:** It is assumed that TLS connections are not broken, i.e., data integrity and confidentiality are ensured. The correct public keys are used to establish connections and private keys are not known to attackers (except for explicitly compromised parties). Exceptions are A6 and A9, where an attacker compromises a TLS terminating endpoint.
- **JWKS:** Where applicable, key distribution mechanisms work as intended, i.e., encryption and signature verification keys of uncompromised parties are retrieved from the correct endpoints.
- **Browsers and Endpoints:** Except for A4a and A4b, devices and browsers used by resource owners are not compromised. Other endpoints not controlled by an attacker behave according to the protocol.

Attackers

FAPI aims to ensure the security goals listed above for arbitrary combinations of the following attackers:

Web Attacker

A1 - Web Attacker

Standard web attacker model. Can send and receive messages just like any other party on the internet. Can participate in protocols flows as normal user. Can send links to honest users that are then visited by these users. This means that the web attacker has the ability to cause arbitrary authorization requests from user's browsers, as long as the contents are known to the attacker.
Cannot intercept messages sent to other parties, and cannot break cryptography. Here: Cannot play the role of a legitimate AS in the ecosystem (see A1a).

A1a - Web Attacker (participating as AS)

Like the web attacker, but can also participate as an AS in the ecosystem. Note that this AS can use/replay messages by honest ASs, and can send users to endpoints of honest ASs. (In other publications, this capability was subsumed in the web attacker model.)

Network Attacker

A2 - Network attacker

Controls the whole network (like a rogue WiFi access point or a nation-state sponsored hacker). Can intercept, block, and tamper with messages intended for other people, but cannot break cryptography (unless the attacker has learned the respective decryption keys). Most attacks by this kind of attacker can be defended against by using transport layer protection like TLS.

Attackers at the Authorization Endpoint

Note: The attackers for the authorization request are more fine-grained than those for the token endpoint and resource endpoint, since these messages pass through the complex environment of the user's browser/app/OS with more things that can go wrong. This demands for a more fine-grained analysis.

A3a - Read Authorization Request

The capabilities of the web attacker, but can also read the authorization request. This might happen on mobile operating systems (where apps can register for URLs), on all operating systems through the browser history, or due to Cross-Site Scripting on the AS. There have been cases where anti-virus software intercepts TLS connections and stores/analyzes URLs.

A3b - Read Authorization Response

The capabilities of the web attacker, but can also read the authorization response. This can happen e.g., due to the URL leaking in proxy logs, web browser logs, web browser history, or on mobile operating systems.

Token Endpoint

A5 - Read Token Requests and Responses

This attacker makes the client use a token endpoint that is not the one of the honest AS. This attacker can read and tamper with messages sent to and from this token endpoint that the client thinks as of an honest AS.

A7 - Read Resource Requests and Responses

The capabilities of the web attacker, but this attacker can also read requests sent to and from the resource server, for example because the attacker can read TLS intercepting proxy logs on the RS's side.

A8 - Tamper with Resource Responses

The capabilities of A7, but this attacker can also tamper with responses from the resource servers (e.g., a fully compromised resource server).

Non-Repudiation Requirements

Beyond what is captured by the attacker model, parties could try to deny having sent a particular message, for example, a payment request. For this purpose, non-repudiation is needed. This is achieved by providing application-level signatures that can be stored together with the payload and meaningful metadata of a request or response.
This is relevant for authorization requests (JAR), authorization responses (JARM), ID tokens, introspection and userinfo responses, and resource requests and responses.

Affected messages

NR1: Authorization Requests
NR2: Authorization Responses
NR3: ID Token Contents
NR4: Introspection Responses
NR5: Userinfo Responses
NR6: Resource Requests
NR7: Resource Responses

Attacks and Mitigations

The following attacks and mitigations must be considered beyond those described in the Security BCP.

(Please refer to the OAuth Attacker Model Cheat Sheet for a non-exhaustive list of attacks that are applicable under each attacker model including references to descriptions of the attacks.)

Code Injection and PKCE Chosen Challenge attack

Attackers A3b and A5 can read the code from the authorization response or token request. To protect against misuse of such a code, client authentication MUST be used. This leaves the attacker the only option to use code injection to misuse the stolen code.

To prevent code injection attacks, PKCE or nonce MUST be used, as recommended in the Security BCP. These are only effective if **client authentication** is used. They can prevent code injection if the integrity and authenticity of the authorization request can be ensured. This is not the case, however, if an attacker is able to mount a PKCE Chosen Challenge Attack with Code Injection attack (requires A1 and either A3b or A5). **There is currently no established mitigation for this attack.**

Cuckoo's Token Attack

A web attacker (A1a) that has obtained an access token can circumvent the sender-constraining of that token using the Cuckoo's Token attack in which the attacker delivers the access token from a token endpoint that he controls to the client, which then uses the token at a resource server (see https://arxiv.org/abs/1901.11520). To prevent this, **the client could communicate the authorization server that was used or the token endpoint to the resource server.** The resource server can then verify that the authorization server is legitimate.

Access Token Injection with ID Token Replay

An attacker A6 that has obtained an access token and ID token from a user can replay both at a token endpoint (see https://arxiv.org/abs/1901.11520). **This could be prevented if, again, the token endpoint would be communicated to the resource server (see above).**

Cross-Browser Payment Initiation

(Might be out of the scope of FAPI, but related.)
In the cross-browser payment initiation attack, an attacker wants to make a user pay for the goods he ordered at the web site of some merchant. To avoid this, payment (or more generally, any actions taken on behalf of the user) **MUST only be executed after the access token was presented at a resource server.**

FAPI Profiles

FAPI Baseline

- Objectives
    - Fulfill the security goals under the FAPI attacker model

    - If possible, avoid usage of application-level signatures (except for the ID token), i.e., no JAR or JARM
- Use PAR and code flow with PKCE
- Use mTLS for sender-constraining of access tokens
- Use client authentication (no public clients)
- (Discuss!) Develop mitigations against Cuckoo's Token Attack and Access Token Injection with ID Token Replay
- Use RAR to represent fine grained authorization requests
- Consent management
    - Guidance on authorization grant (aka consent) lifecycle management, including re-authorization, updated authorizations, separate authorization and revocation
    - refresh tokens and authorizations
    - OAuth Token Revocation (RFC 7009)
- CIBA (POS, call center use cases) - TL: I would strongly advocate to generalize CIBA to support generic OAuth
- app 2 app redirect (UX & preventing the SCA dead end)

FAPI Baseline plus Non-Repudiation (Advanced)

Same as Baseline, plus:

- Fulfill the non-repudiation goals
- Non-repudiation via JAR and JARM (NR1 and NR2)
- Signed introspection/userinfo responses (NR4 and NR5)
- Signed requests to and responses from the resource endpoints (NR6 and NR7)
- Include ID Tokens as detached signatures for backward compatibility
- Message signing

Changelog

2019-12-20

Removed attackers A4a and A4b

We must assume that the integrity of honest AS's authorization endpoints is given at all times. Otherwise, attackers could trivially read and misuse authentication credentials. Same for a compromised browser or operating system. The capabilities that are realistic, reading authorization requests or responses and injecting authorization requests, are captured by A3a, A3b, and A1, respectively.

Conflate attackers A5 and A6

As above, integrity of the AS must be given at all times, else much easier attacks that are out of the scope of what OAuth can protect against are possible. Therefore, the integrity of

the token endpoint must be given at all times. A5 (former A5 and A6) is now described more precisely: An attacker can make a client developer use a different token endpoint (different URI), but not read/tamper with requests to/from the original token endpoint.

Acknowledgements

These attacker models draw heavily from the formal analysis of the first FAPI versions conducted by Pedram Hosseyni, Ralf Kuesters (both University of Stuttgart), and Daniel Fett (yes.com).