オープン API のセキュリティ：認可処理における脆弱性と対策の高度化

宇根正志（うねまさし）

要　旨

オープン API は、API の提供者が外部の組織への使用に供するために公開した API を指す。金融分野では、金融機関が電子決済等代行業者などのフィンテック事業者に対して顧客の口座情報を提供するケースなどで活用されている。金融機関は、顧客がフィンテック事業者に対して口座情報の提供を認可したことを確認し、フィンテック事業者と連携して口座情報を安全に提供することが求められる。

こうした認可の処理に関するプロトコルの標準仕様として OAuth 2.0 が広く採用されている。もっとも、OAuth 2.0 の要求事項を十分に満たしていない実装例が少なからず存在していることが、学術研究の成果として最近発表されている。このようなサービスには脆弱性が存在し、それが悪用されると情報漏洩などのリスクにつながる可能性があるため、OAuth 2.0 を実装する際に脆弱性を極力排除する必要がある。

脆弱性を排除するうえで、OAuth 2.0 のセキュリティ・プロファイルである FAPI 2.0 を活用することが有効である。ただし、実装環境や想定される攻撃方法によっては、FAPI 2.0 の要求事項を満たすだけではリスクを十分に軽減できない場合がある。そのため、自社のサービスで想定される実装環境や攻撃方法を考慮したうえで、FAPI 2.0 の要求事項を適用しつつ、追加的な対応の必要性を検討することが重要である。

キーワード：　オープン API、脆弱性、セキュリティ、フィンテック、リスク、FAPI 2.0、OAuth 2.0

本稿は 2024 年 5 月 31 日時点の情報に基づいて作成した。本稿の作成に当たっては、小岩井航介氏（KDDI 株式会社）から有益なコメントを頂いた。ここに記して感謝したい。ただし、本稿に示されている意見は、筆者個人に属し、日本銀行の公式見解を示すものではない。また、ありうべき誤りはすべて筆者個人に属する。

宇根正志　日本銀行金融研究所参事役（E-mail: masashi.une@boj.or.jp）

1. はじめに

オープン API（application programming interface）は、API の提供者が外部の組織に使用させるために公開した API を指す。オープン API を活用した金融サービスの代表的な例としては、金融機関が電子決済等代行業者などのフィンテック事業者に対して顧客の口座情報や金融取引にかかる処理を API によって提供するケースが挙げられる。個人の顧客が複数の金融機関に開設している預金口座の情報をフィンテック事業者のアプリによって一元的に収集・管理するなど、使い勝手のよいサービスが既に提供されている。フィンテック事業者がこうしたサービスを複数の金融機関と連携して円滑に実現するうえで、オープン API は不可欠な技術である（Alam, Azad, and Ali [2022]、Basel Committee on Banking Supervision [2024]）。

オープン API を用いた安全な金融サービスを実現するうえで、認可に関する処理（認可処理）が重要である。金融機関は、顧客がフィンテック事業者に対して何らかの処理の実行を認可したことを適切に確認するなど、認可処理を適切に実行する必要がある。金融サービスにおける認可処理の実装では、OAuth 2.0 に準拠するケースが多い（中村［2018］）。OAuth 2.0 は、認可処理を実現するプロトコル（認可プロトコル）のフレームワークを定める標準仕様 RFC 6749（Hardt [2012]）とそれに付随する各種機能を実現するための多数の標準仕様から構成されている。適用するサービスの機能や特性に応じて採用する標準仕様を選択し、フレームワークと組み合わせて実装するというアプローチとなっている。そのため、セキュリティの観点からは、オープン API を用いるサービスのリスク[1] を評価したうえで、リスク軽減策として適切なセキュリティ機能を選択し、それを実現するための標準仕様の要求事項を選択して実装することが求められる。

もっとも、OAuth 2.0 に関連する標準仕様にはさまざまな要求事項が存在することから、認可プロトコルの処理フローをなるべく効率化しつつ、選択したセキュリティ機能に必要な要求事項を適切に抽出して認可プロトコルの設計に反映することは必ずしも容易でない。OAuth 2.0 に準拠したオープン API の実装において、必須とされる要求事項が満たされていない事例が少なからず存在することが学術研究の成果として最近発表されている（Philippaerts, Preuveneers, and Joosen [2022, 2023]）。この研究では、OAuth 2.0 に準拠したサービス（約 100 件）を対象に、標準仕様やベスト・プラクティスにおいて必須とされる要求事項が満たされているか否か、満

1 オープン API を用いるサービスのリスクを評価する際には、本稿で焦点を当てる認可に関連するリスクに加えて、API に関連する業務フローやシステム・リソースのリスク、API の接続先のリスクなど、他のリスクについても評価し対応する必要がある。API に関連する各種リスクについては補論 1. を参照されたい。

たされていない場合にはどのような脆弱性が存在しうるかが調査・分析された。その結果、OAuth 2.0 のフレームワークの技術仕様において必須とされる要求事項の約 2 割が満たされていなかったことが判明した。また、要求事項が満たされていないことによって脆弱性が存在し、それが悪用された場合、顧客に関する情報が漏洩するなどの問題が生じうることが示された。

OAuth 2.0 の標準仕様における要求事項を適切に選択するうえで、セキュリティの観点から選択すべき要求事項を定めている FAPI 2.0 を活用することが有用である（Fett [2022b]）。FAPI 2.0 は、比較的価値が高い情報を取得したり取引を実行したりするケース（high-value scenario）を想定しており、FAPI 2.0 の要求事項に沿って認可プロトコルを設計・実装することによって比較的高いセキュリティを達成することが期待できる。また、一定の条件のもとでは特定のセキュリティ目標の達成を数学的に証明することができるという望ましい面もある（Hosseyni, Küsters, and Würtele [2024]）。

ただし、証明の前提となる条件が実装環境と合致しないケースでは、セキュリティ目標の達成可否が定かでないため、相応のリスクが存在する可能性がある。FAPI 2.0 の要求事項を採用する際には、適用対象のサービスや実装環境が FAPI 2.0 における想定と整合的か否かを確認し、整合的でない部分がある場合、それに関連するリスクを軽減するための対策を追加するなどの対応が求められる。こうした留意点を考慮しつつ FAPI 2.0 を適切に活用することが重要である。

2 節では、オープン API における認可処理の全体像と、それを具体化した OAuth 2.0 のフレームワークを説明する。3 節では、OAuth 2.0 の認可プロトコルに対する主な脅威や攻撃方法を中村［2018］をベースに説明するほか、Philippaerts, Preuveneers, and Joosen [2022, 2023] を引用し、標準仕様などの要求事項の充足度合いや脆弱性に関する研究結果を紹介する。4 節では、FAPI 2.0 の主な要求事項を説明するほか、3 節の攻撃方法への有効性を考察する。

2. オープン API における認可処理

本節では、認可処理の全体像と OAuth 2.0 の認可プロトコルのフレームワークを説明する。

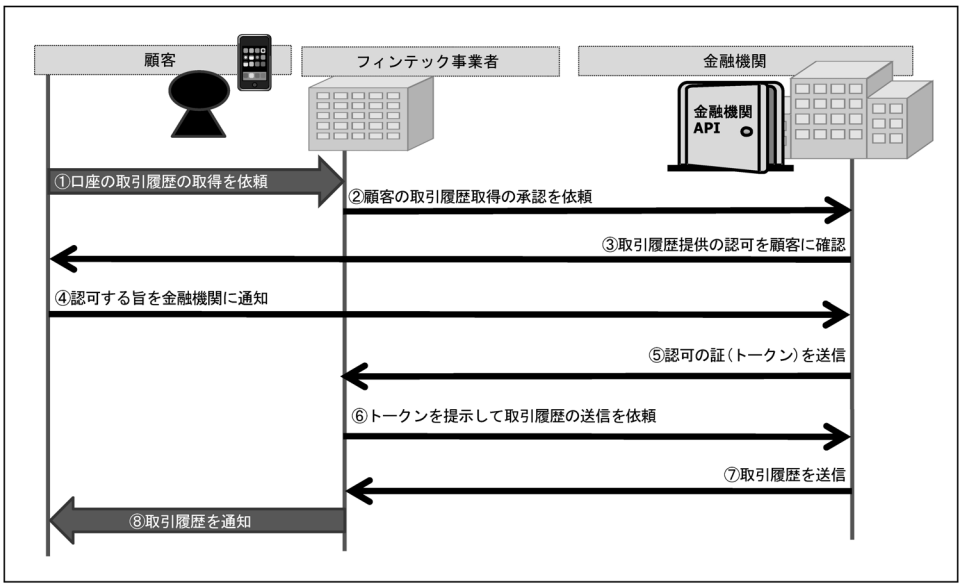
(1) 認可処理の全体像

理解しやすいように、ここでは、フィンテック事業者が金融機関にオープン API

を介してアクセスし、顧客の口座の取引履歴を取得するケースを考える。顧客の依頼を受けたフィンテック事業者は、取引履歴を顧客に代わって取得することを金融機関に認めてもらう必要がある。金融機関は、フィンテック事業者に取引履歴を提供することを顧客に確認する必要がある。こうした処理の流れは次のように整理することができる（図 1 を参照）[2]。

① 顧客は、自分の口座の取引履歴の取得をフィンテック事業者に依頼する。
② フィンテック事業者は、金融機関のオープン API を用いて、顧客の口座の取引履歴の取得に対する承認を金融機関に依頼する。
③ 金融機関は、フィンテック事業者に取引履歴を提供することを顧客に確認する。
④ 顧客は提供を認可する旨を返信する。
⑤ 金融機関は、認可の証となるデータ（トークン）をフィンテック事業者に送信する。
⑥ フィンテック事業者はトークンを金融機関に提示して取引履歴の送信を依頼する。
⑦ 金融機関は取引履歴をフィンテック事業者に送信する。
⑧ フィンテック事業者は顧客に取引履歴を通知する。

図 1　オープン API における認可処理の全体像（イメージ）



2 ここでは、認可処理に注目するために、通信相手の確認や通信メッセージの一貫性の確認の処理を明記していないが、通常、これらの処理（認証）も実行される。

(2) OAuth 2.0 の認可プロトコルのフレームワーク

イ. エンティティ

OAuth 2.0 の認可プロトコルのフレームワークでは、リソース・オーナー、ユーザ・エージェント、クライアント、リソース・サーバ、認可サーバが登場する。

●リソース・オーナー

リソース・オーナーは、金融機関の顧客であり、フィンテック事業者の顧客でもある。ここでのリソースとは、フィンテック事業者が顧客の認可によって取得する情報（金融機関に顧客が開設している口座の取引履歴など）や実行する取引（送金など）を意味している。以下では、理解しやすくするために、リソース・オーナーを顧客と呼ぶ。

●ユーザ・エージェント

ユーザ・エージェントは、顧客がフィンテック事業者や金融機関とやり取りするためのソフトウェアである。主に、顧客の端末のブラウザを想定している。以下ではユーザ・エージェントをブラウザと呼ぶ。

●クライアント

クライアントは、フィンテック事業者が顧客にサービスを提供するためのアプリケーション・ソフトウェアである。

クライアントは、顧客の端末内で動作するもの（ネイティブ・アプリ）[3] と、フィンテック事業者のサーバで動作するもの（サーバ・アプリ）がある。OAuth 2.0 では、ネイティブ・アプリに関して、各アプリが固有の秘密情報（暗号鍵など）[4] を安全に使用することができないケース（パブリック・クライアント）を対象としている。サーバ・アプリに関しては、固有の秘密情報を安全に使用することができるケース（コンフィデンシャル・クライアント[5]）を想定している。

●リソース・サーバ

リソース・サーバは、金融機関のサーバであり、顧客の口座を管理して口座の取引履歴を提供したり送金を実行したりする。

●認可サーバ

認可サーバは、金融機関のサーバであり、顧客の認可を確認したうえでフィン

3 ネイティブ・アプリは、インターネット上のアプリ配信サイトなどから端末にダウンロード・インストールされて動作するケースや、顧客の端末のブラウザ上でウェブ・アプリケーションとして動作するケースがある。

4 秘密情報は、クライアントが他のエンティティに対して認証を実施する際などに用いられる。

5 コンフィデンシャル・クライアントには、各クライアントに固有の秘密情報がそのクライアント自身によって安全に管理されるケースに加えて、端末や OS の機能によって安全に管理されるケースも含まれる。

テック事業者に顧客のリソースへのアクセスを許可する。

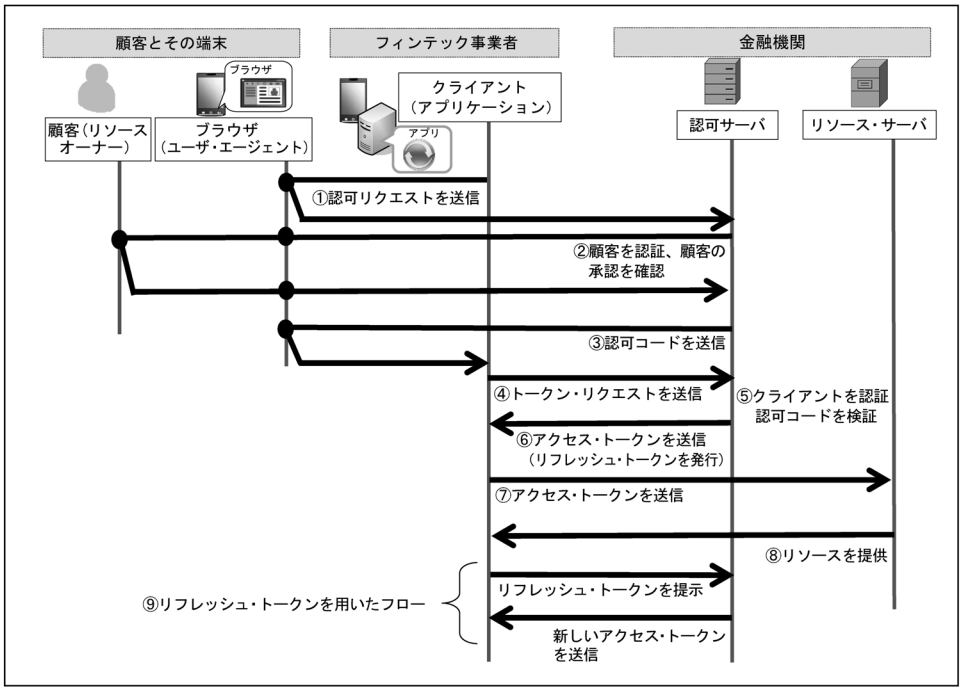
ロ．認可プロトコルの処理フロー

OAuth 2.0 では、認可プロトコルの処理フローとして、認可コード・フロー[6]、インプリシット・フロー、リソース・オーナー・パスワード・クレデンシャル・フロー、クライアント・クレデンシャル・フローの 4 種類[7] が想定されている[8]。以下では、代表的なフローである認可コード・フローを取り上げて説明する。概要は以下のとおりである（図 2 を参照）。

① クライアントは、顧客の口座の取引履歴へのアクセスを求めるメッセージ（認可リクエスト）を認可サーバに送信する。認可リクエストは顧客のブラウザを経由して認可サーバに送られる。
② 認可サーバは、顧客を認証し、クライアントへの認可を確認する。顧客の認証では、例えば、認可サーバはインターネット・バンキングのログイン ID・パスワードの入力を顧客に求める。
③ 顧客から認可を確認した後、認可サーバは、その証として認可コードを発行し、それを含むメッセージ（認可レスポンス）をクライアントに送信する。認可レスポンスは、ブラウザを経由してクライアントに送信される。
④ クライアントは、認可サーバを認証した後、認可コードなどを認可サーバに送信し、アクセス・トークンの送信を依頼する（トークン・リクエスト）。
⑤ 認可サーバは、クライアントを認証した後、認可コードを検証する。具体的には、認可コードが有効であること、アクセス・トークンの送信先が認可コードの送信先と一致していることなどを検証する。
⑥ 認可サーバは、認可コードの検証が成功した場合、アクセス・トークンを発

6 認可コードは、認可サーバが顧客による認可を確認した証としてクライアントに対して発行するトークンである。
7 インプリシット・フローは、認可サーバがクライアントに認可コードを発行せずに直接アクセス・トークン（クライアントがリソースにアクセスする際に用いられるトークン）を発行する方式である。リソース・オーナー・パスワード・クレデンシャル・フローは、クライアントが認可サーバから認可コードを受け取る代わりに顧客からログイン ID とパスワードを受け取り、それを認可サーバに送信することによってアクセス・トークンを得る方式である。クライアント・クレデンシャル・フローは、クライアントが認可コードの代わりに自分のクレデンシャルを認可サーバに送信してアクセス・トークンを得る方式である。
8 これらに加えて、OAuth 2.0 の拡張として、デバイス認可フロー（Denniss *et al.* [2019]）とクライアント・イニシエイティッド・バックチャネル認証フロー（Fernández *et al.* [2021]）が存在する。デバイス認可フローは、顧客の端末がブラウザを搭載していない、または、顧客とのインタフェースに制限があるなどの場合を想定した方式である。クライアント・イニシエイティッド・バックチャネル認証フローは、クライアントと（顧客の）ブラウザ間の通信が行われることなく、認可サーバが顧客の認証や認可の確認を（顧客の）端末との間で直接行う場合を想定した方式である。

図 2　OAuth 2.0 における認可コード・フロー



資料：中村［2018］図表 2

行してクライアントに送信する。認可サーバは、アクセス・トークンが有効期限切れとなった際に新しいアクセス・トークンを得るためのリフレッシュ・トークンを発行してクライアントに送信する場合がある。

⑦ クライアントは、アクセス・トークンをリソース・サーバに送信する。

⑧ リソース・サーバはアクセス・トークンを検証し、それが成功すればクライアントにリソース（ここでは口座の取引履歴）を提供する。

⑨ クライアントは、アクセス・トークンの有効期限が切れた際に、リフレッシュ・トークンを認可サーバに提示して新しいアクセス・トークンを取得する場合がある。リフレッシュ・トークンを取得しておくことによって、クライアントは、認可処理を再度実施することなく、新しいアクセス・トークンを得ることができる。

3. 認可処理における主な脅威

(1) 攻撃者の想定と主な脅威

攻撃者が、2 節で紹介した一連の処理における脆弱性を悪用して被攻撃者（顧客）の口座の取引履歴の入手を試みるケースを考える。金融機関、フィンテック事業者、攻撃者に関して以下の想定を置く。これは中村［2018］の想定と同一である。

・ 金融機関が管理するサーバの特権は攻撃者に奪取されない。
・ 金融機関とフィンテック事業者の内部者は不正行為を行わない。
・ 攻撃者は、クライアント、それが稼働する端末・サーバ、顧客の端末、ブラウザに関して、管理者権限を奪取することがある。
・ 攻撃者は、暗号化・署名に関する処理、および、認証に関する処理を不正に実行することができない。

(2) 主な脅威の類型

本節（1）の想定のもとで主な脅威を整理すると以下の 4 つが挙げられる（表 1 を参照）[9]。これらのうち、I〜III の脅威に対応する攻撃方法は中村［2018］を引用し、IV の脅威に対応する攻撃方法は OAuth 2.0 のセキュリティ・ベスト・カレント・プラクティス（Lodderstedt *et al.* [2024]）[10] において紹介されているものを引用した。各攻撃方法の概要は補論 2. を参照されたい。

I. アクセス・トークンを奪取する。
II. クライアントとリソース・サーバとの間のセッションを奪取する。
III. 自分の口座の取引履歴を攻撃者の口座の取引履歴として入力するように顧客を誘導する。
IV. ブラウザを不正なサイトにアクセスさせ、クレデンシャルを入力するように顧客を誘導して奪取する。その後、顧客のクレデンシャルを用いて顧客になりすまし、認可プロセスを実行する。

9 脅威や攻撃方法の名称については、内容を理解しやすくするために、中村［2018］のものを一部変更している。
10 このベスト・プラクティスは OAuth 2.0 において想定すべき脅威とそれへの対策例を記載している。

表 1　主な脅威と攻撃方法

主な脅威	攻撃方法
Ⅰ．アクセス・トークンの奪取	① アクセス・トークンの推測
	② 通信経路上での盗聴
	③ クライアントからの奪取
	④ 転送機能の悪用
	⑤ 中継サーバの自動接続機能の悪用
	⑥ 認可リクエストの改変
Ⅱ．セッションの奪取	⑦ 不正なアプリの使用
	⑧ セッション管理用パラメータの奪取
Ⅲ．顧客の誘導	⑨ 攻撃者のリソースの偽装
	⑩ セッション管理用パラメータの悪用
Ⅳ．顧客のクレデンシャルの奪取	⑪ 不正なサイトへのリダイレクト

備考：中村［2018］と Lodderstedt *et al.* [2024] をベースに作成した。

(3)　認可プロトコルの実装における脆弱性と脅威の調査・分析の事例

本節（2）で示した脅威や攻撃方法は広く知られている。したがって、OAuth 2.0 の各種標準仕様の要求事項などを参照しつつ、各脅威によるリスクを軽減するための対応が相応に実施されているとみることができる。この点について、標準仕様などの要求事項が実装においてどの程度満たされているか、また、どのような脆弱性が残存しているかを調査・分析した研究事例（Philippaerts, Preuveneers, and Joosen [2022, 2023]）が最近発表されている。

この研究は個人の ID 情報を第三者に提供するサービスを対象としている。このサービスは、ある組織が保持している ID 情報を、それに紐づく個人による認可に基づいて他の組織に提供するというものであり、広く提供されている。この研究は、一般的な認可プロトコルの実装におけるセキュリティの現状を把握して金融分野での検討に活用するという観点で有益である。

イ．調査の概要

調査の対象や実施時期など、概要は以下のとおりである。

- 調査対象：オープン API による個人 ID 提供事業者（individual identity

provider）。全体で 100 件（初回の調査）[11]。このうち 20 件は OpenID Connect もサポートしている[12]。また、海外の金融機関 2 件が含まれている。

- 調査実施時期（2 回）：2020 年 11 月（初回）と 2023 年 3 月（フォローアップ）であり、フォローアップでの調査対象は 100 件（初回）のうち 92 件。
- 調査の内容：主な標準仕様やベスト・プラクティスにおいて必須（must）や推奨（should）とされている要求事項が満たされているか否かを明らかにする。ただし、調査者は、クライアントおよび顧客として調査対象先にアクセスすることから、要求事項のうち、認可サーバやリソース・サーバに関するもののみを調査する。対象とする主な標準仕様やベスト・プラクティスは以下のとおりである[13]。
  - RFC 6749: The OAuth 2.0 Authorization Framework
  - RFC 6750: The OAuth 2.0 Authorization Framework: Bearer Token Usage[14]
  - RFC 6819: OAuth 2.0 Threat Model and Security Considerations[15]
  - RFC 7009: OAuth 2.0 Token Revocation[16]
  - RFC 7523: JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants[17]
  - RFC 7636: Proof Key for Code Exchange by OAuth Public Clients (PKCE)[18]

11 調査対象先は Philippaerts, Preuveneers, and Joosen [2022] に明記されている。この論文の筆者らは論文発表前に調査対象先に対して内容を連絡し発表の許諾を得ている（coordinated disclosure）。

12 OpenID Connect は、OAuth 2.0 において、認可サーバの認証結果に基づいてクライアントがリソース・オーナーの属性を検証するとともに、クライアントがリソース・オーナーの属性情報を認可サーバなどから取得するプロトコルである（Sakimura *et al.* [2014]）。

13 いずれも 2021 年 5 月 20 日時点のものが参照されている。

14 この標準仕様は、保持者と紐づけされていないアクセス・トークン（bearer token）を使用するケースにおいて、クライアントがアクセス・トークンをリソース・サーバに送信する手順やメッセージ形式を定めている（Jones and Hardt [2012]）。

15 この標準文書は、OAuth 2.0 におけるセキュリティの特性、プロトコルを実装・運用する際に想定すべき脅威、各脅威への対策方針などを記載している（Lodderstedt, McGloin, and Hunt [2013]）。

16 この標準仕様は、アクセス・トークンなどを失効させる際に、クライアントと認可サーバの通信されるメッセージの形式、セキュリティ上の留意点などを定めている（Lodderstedt, Dronia, and Scurtescu [2013]）。

17 この標準仕様は、JavaScript ベースのトークン JWT（JSON〈JavaScript Object Notation〉Web Token）によってアクセス・トークンを実装するケースにおいて、トークンの形式や処理方法、使用上の留意点などを定めている（Jones, Campbell, and Mortimore [2015]）。

18 この標準仕様は、認可サーバが認可コードの正当な保持者を確認する手法 PKCE を定めている（Sakimura, Bradley, and Agarwal [2015]）。クライアントは、乱数 R とそのハッシュ値（コード・チャレンジ）を生成したうえで、認可リクエストにコード・チャレンジと（それを生成した）ハッシュ関数の情報を加えて認可サーバに送信する。認可サーバは、認可コード発行時に、それと紐づけてコード・チャレンジとハッシュ関数を登録する。トークン・リクエストにおいて、クライアントは R を認可コードとともに認可サーバに送信し、認可サーバは R からコード・チャレンジを生成して認可コードとの対応関係を確認する。確認できた場合、認可サーバは通信相手のクライアントが認可

  - OAuth 2.0 Security Best Current Practice
  - OpenID Connect
- 調査の手法：以下の手順で調査が実施された。
  ① 標準仕様などからセキュリティに関する要求事項を抽出する。
  ② 抽出した各要求事項が調査対象先において満たされているか否かを明確にするためのテスト・ケース[19]をそれぞれ作成する。
  ③ 調査者の端末に（各調査対象先の）クライアントをインストールし、調査対象先にアクセスしてテスト・ケースを実行する。
  ④ テスト・ケースの結果から、調査対象先における各要求事項の達成状況を明らかにするほか、要求事項が満たされていないことによって生じうる脆弱性やそれを悪用する脅威を分析する。

ロ. 主な調査・分析結果

上記の調査・分析が 2020 年 11 月と 2023 年 3 月にそれぞれ実施された。主な結果を紹介すると以下のとおりである。

・調査対象先がサポートしていた認可プロトコルの処理フローに関しては、2020 年 11 月の調査において、認可コード・フローをサポートしていた調査対象先が全体の約 94%であった。インプリシット・フロー、リソース・オーナー・パスワード・クレデンシャル・フローをサポートしていた先はそれぞれ全体の約 37%、約 3%であった。
・標準仕様などにおける要求事項の充足度合いを表す指標として、失敗したテスト・ケースがテスト・ケース全体に占める割合（失敗率〈failure rate〉）[20]を算出した。失敗率が低いほど要求事項の充足度合いが高いと判断できる。2023 年 3 月の調査では、OAuth 2.0 のフレームワークの標準仕様 RFC 6749 において、必須および推奨の要求事項に対応するテスト・ケースの失敗率がそれぞれ約 19%、約 37%であった（表 2 を参照）。セキュリティ・ベスト・カレント・プラクティスでは、必須および推奨の要求事項に対応するテスト・ケースの失敗率がそれぞ

コードの正当な保持者と判断する。

19 テスト・ケースは、それが失敗した場合、対応する要求事項が満たされていないと判断できるように作成される。例えば、要求事項が「コンフィデンシャル・クライアントは認可サーバに対して認証を実施しなければならない（必須である）」というものの場合、これに対応するテスト・ケースとして、「クライアントは、クライアント認証を実施することなく、認可サーバからアクセス・トークンを取得することができない」と設定する。仮に、ある個人 ID 提供事業者においてこのテスト・ケースが失敗した場合（クライアント認証なしでアクセス・トークンの取得が可能な場合）、対応する要求事項が満たされていないと判断することができる。

20 調査対象先（全体で 92 件）に対してそれぞれテスト・ケースを実施し、失敗したケースがテスト・ケース全体に占める割合を調査対象先ごとに算出する。そのうえで、失敗したケースの割合を調査対象先に関して平均し、それをテスト・ケースの失敗率とする。

表２　各標準仕様などにおけるテスト・ケースの失敗率（2023 年 3 月時点）

標準仕様やベスト･プラクティス	テスト・ケースの失敗率	
	必須の要求事項	推奨の要求事項
RFC 6749（Authorization Framework）	19.1	36.9
RFC 6750（Bearer Token Usage）	0.4	62.2
RFC 6819（Threat Model and Security Considerations）	1.8	21.6
RFC 7009（Token Revocation）	4.3	12.5
RFC 7523（JWT Profile）	12.5	---
RFC 7636（PKCE）	8.4	---
Security Best Current Practice	31.7	65.6
OpenID Connect	16.2	5.3

備考：表中の数字の単位はパーセント。値が小さいほど要求事項の充足度合いが高い。

れ約 32％、約 66％であった。

・テスト・ケースの結果から判明した主な脆弱性・脅威は以下のとおりである。

① リフレッシュ・トークンの奪取・悪用：約 44％の調査対象先（2020 年 11 月時点）において、認可サーバによるリフレッシュ・トークンの取扱いが不適切な場合[21]があった。リフレッシュ・トークンを奪取した攻撃者は、この取扱いによってアクセス・トークンを入手する。この脆弱性は、表 1 におけるクライアントからの奪取に対応する攻撃で悪用される可能性がある。

② 認可コードの悪用：約 43％の調査対象先において、認可サーバが PKCE を適切に実施していない場合[22]があったほか、約 14％の調査先において、使用済みの認可コードが再度提示された際にアクセス・トークンを誤って発行する場合があった。このような場合、認可コードを奪取した攻撃者はアクセス・トークンを入手することができる。この脆弱性は、表 1 における中継サーバの自動接続機能の悪用やセッション管理用パラメータの奪取に対応する攻撃で悪用される可能性がある。

21 例えば、クライアント認証によってリフレッシュ・トークンの使用を制限していないケースや、認可サーバが使用済みのリフレッシュ・トークンを無効化していないケースである。

22 例えば、認可サーバが、コード・チャレンジの値を含まない認可リクエストを受信したにもかかわらず認可プロトコルをそのまま継続する（エラー・メッセージを返信したり認可プロトコルを中止したりしない）ケースが紹介されている。このようなケースにおいて、攻撃者は、認可リクエストのコード・チャレンジのみを削除することによって PKCE を無効化することができる場合がある（PKCE ダウングレード攻撃）。

③ クライアントへのなりすまし：約 24％の調査対象先において、クライアントが認証用に秘密情報を保持しているにもかかわらず、認可サーバがトークン・リクエストの際などにクライアント認証を適切に実行しない場合があった。このような場合、攻撃者は認可サーバに対してクライアントになりすますことができる。この脆弱性は、表 1 におけるクライアントからの奪取、中継サーバの自動接続機能の悪用、認可リクエストの改変、不正なアプリの使用、セッション管理用パラメータの奪取に対応する攻撃で悪用される可能性がある。

④ ログ・ファイルなどからのアクセス・トークンの奪取：約 40％の調査対象先において、リソース・サーバがクライアントからリソース・リクエストを受信した際に、それに含まれるアクセス・トークンをログ・ファイルなどに意図せず格納しうる場合があった[23]。このような場合、ログ・ファイルへのアクセスが可能な攻撃者はアクセス・トークンを奪取することができる。この脆弱性は、表 1 におけるクライアントからの奪取に対応する攻撃で悪用される可能性がある。

ハ．小括と留意点

標準仕様などにおける必須の要求事項が少なからず満たされていないケースがあることが示された。また、リフレッシュ・トークンやアクセス・トークンの奪取などに悪用されうる脆弱性が相応の割合の調査対象先で存在することも明らかとなった。

ただし、この調査は認可プロトコルにのみ焦点を当てたものであり、調査対象先のサービスのリスクを検討しているわけではない。例えば、認可プロトコルに脆弱性があったとしても、保護対象となる情報（この場合は個人 ID）が漏洩するなどのリスクを運用によって軽減しているケース（例えば、認可コードやアクセス・トークンの有効期間を短く設定する）が考えられる。今回の脆弱性がサービスのリスクの増加につながるとは必ずしもいえない点に留意が必要である。

今回の調査は個人 ID を提供するサービスを対象としたものであり、金融分野の状況を示すものではないが、金融機関やフィンテック事業者は、認可プロトコルの実装における脆弱性の実例として捉え、自社のサービスにおける脆弱性の洗出しやリスク評価を実施する際に参考にすることができる。

23 クライアントがリソース・リクエストにおいてアクセス・トークンを URL の一部（クエリ・パラメータ）として埋め込んでリソース・サーバに送信する場合、これを受信したリソース・サーバがログ・ファイルにアクセス・トークンを格納するほか、過去の通信相手の URL を外部に送信する際にアクセス・トークンも一緒に送信する可能性がある。リソース・サーバには、アクセス・トークンをログ・ファイルに格納しないようにする、または、ログ・ファイルからの情報漏洩を防止することが求められる。

4. 認可プロトコルにおける FAPI 2.0 の適用

3 節（3）の研究成果は、OAuth 2.0 の認可プロトコルの実装にさまざまな脆弱性が潜んでいる可能性を示唆している。こうした脆弱性を作り込まないように認可プロトコルを設計・実装することが望ましい。その方法の 1 つとして、FAPI 2.0 を活用することが考えられる。

(1) FAPI 2.0 の概要

FAPI 2.0 の旧バージョンである FAPI セキュリティ・プロファイル 1.0（FAPI 1.0）は、2016 年 6 月に OpenID Foundation の FAPI ワーキング・グループによって策定が開始され、2021 年 3 月に公開された。その後、FAPI 1.0 の改訂が始まり、2022 年 12 月に FAPI 2.0 セキュリティ・プロファイルのインプリメンテーション・ドラフト（Fett [2022b]）が公開された[24]。

FAPI という名称は、金融サービスを対象として策定された経緯から、当初は「Financial-grade API」の略称として呼ばれていた。その後、金融分野に限らず医療分野や公的分野などでも幅広く活用されるようになったことから、現在では「FAPI」が（略称ではなく）正式名称となっている。

FAPI 2.0 のセキュリティ目標は、以下のそれぞれの状態を達成することである（Fett [2022a]）。

・攻撃者が他者の（保護された）リソースにアクセスできない。
・攻撃者が他者の ID のもとでクライアントにログインできない。
・攻撃者が自分の ID のもとで他者をクライアントにログインさせることができない。
・攻撃者が自分のリソースに他者を誘導し使用させることができない。

FAPI 2.0 の要求事項を満たす認可プロトコルが一定の条件のもとでセキュリティ目標を達成することは、形式検証の手法（formal verification）によって証明されている（Hosseyni, Küsters, and Würtele [2024]）。また、FAPI 2.0 を適用した認可プロトコルが実際に要求事項を満たしていることを確認するためのツールが提供されてい

24 以下の FAPI 2.0 に関する説明は 2022 年 12 月のインプリメンテーション・ドラフトに基づく。ただし、本稿の執筆時点において FAPI 2.0 の標準化が継続しており、本節の説明と異なる内容で今後標準化が進められる可能性がある点に留意されたい。

るほか、ツールによる確認が実施された事例も知られている[25]。

(2) FAPI 2.0 における想定環境や攻撃者

イ. 想定環境とスコープ

主な想定環境とスコープは以下のとおりである。

・暗号化やデジタル署名に用いられる鍵の配送は、期待通り安全に行われる。
・顧客の端末やブラウザは安全に動作する。
・攻撃者に悪用されていないエンティティは期待通りに動作する。
・暗号鍵などの秘密情報はランダムで安全に生成される。攻撃者が秘密情報を効率的に推定することは困難である。
・クレデンシャルの漏洩につながるデータベースの設定ミス、OS やブラウザの設定ミスは、スコープ外とする。
・認可サーバ上でマルウェアなどを実行させる攻撃（remote code execution）はスコープ外とする。
・フィッシングへの対策はスコープ外とする。
・クライアントや認可サーバによる顧客の登録・本人確認、当人確認、ID・アクセス管理は、スコープ外とする。

ロ. 攻撃者の類型

攻撃者の類型は主に表 3 の 5 つである。特に、表 3 の A3a と A7 の攻撃者は、認可サーバとリソース・サーバにそれぞれアクセス可能であり、相当高度なスキルを有していることを想定している。もっとも、アクセス対象は認可リクエストとリソース・サーバへのリクエストであり、攻撃者が入手できる情報が限定されている。

(3) FAPI 2.0 の要求事項を適用した認可プロトコル

FAPI 2.0 の主な要求事項は以下のとおりである（図 3 参照）[26]。

25 イギリスやブラジルのオープン・バンキング（Open Banking）においては、FAPI 1.0 に基づくものではあるが、こうしたツールによる認証の取得が必須とされている（OpenID Foundation [2022]）。適合性を確認するツールに関する情報は https://openid.net/certification/に掲載されている（アクセス日：2024 年 5 月 27 日）。確認済みの事例は https://openid.net/developers/certified-openid-connect-implementations/ に掲載されている（アクセス日：2024 年 5 月 27 日）。

26 FAPI 2.0 では、認可サーバがインプリシット・フローとリソース・オーナー・パスワード・クレデンシャル・フローによるリクエストを拒否することが定められている。したがって、FAPI 2.0 を適用した認可プロトコルでは、これら以外のフローが使用されることとなる。

表 3　攻撃者の類型

類型	攻撃者の行動の概要
A1	・顧客を不正なサイトに誘導し、任意の認可リクエストを開始させる。 ・他のエンティティ間のメッセージを横取りしたりブロックしたりすることは不可能。認可サーバとして振る舞うこともできない。
A1a	・A1 の行動に加えて、認可サーバとして振る舞う。 ・他の認可サーバから取得したメッセージを悪用したり、顧客を不正なサイトに誘導したりすることができる。
A2	・無線アクセス・ポイントを悪用したり、脆弱なネットワーク・ノードを操作したりして、顧客が使用するネットワークを悪用可能。 ・メッセージの横取り・ブロック・改変が可能。
A3a	・A1 の行動に加えて、認可サーバにアクセス可能。 ・認可リクエストを入手可能。
A7	・A1 の行動に加えて、リソース・サーバにアクセス可能。 ・リソース・サーバへのリクエストを入手可能。

・エンティティ間における暗号通信や認証は、暗号通信プロトコル TLS（Transport Layer Security）のバージョン 1.3、または、一定の条件を満たすバージョン 1.2 を使用して行う[27]。
・クライアントはコンフィデンシャル・クライアントのみとする。
・クライアントへのメタデータの提供は、RFC 8414（OAuth 2.0 Authorization Server Metadata）や OpenID Connect Discovery によって適切に行う[28]。
・認可サーバによるクライアント認証の方式は、MTLS（Mutual TLS）[29] または private_key_jwt[30] とする。

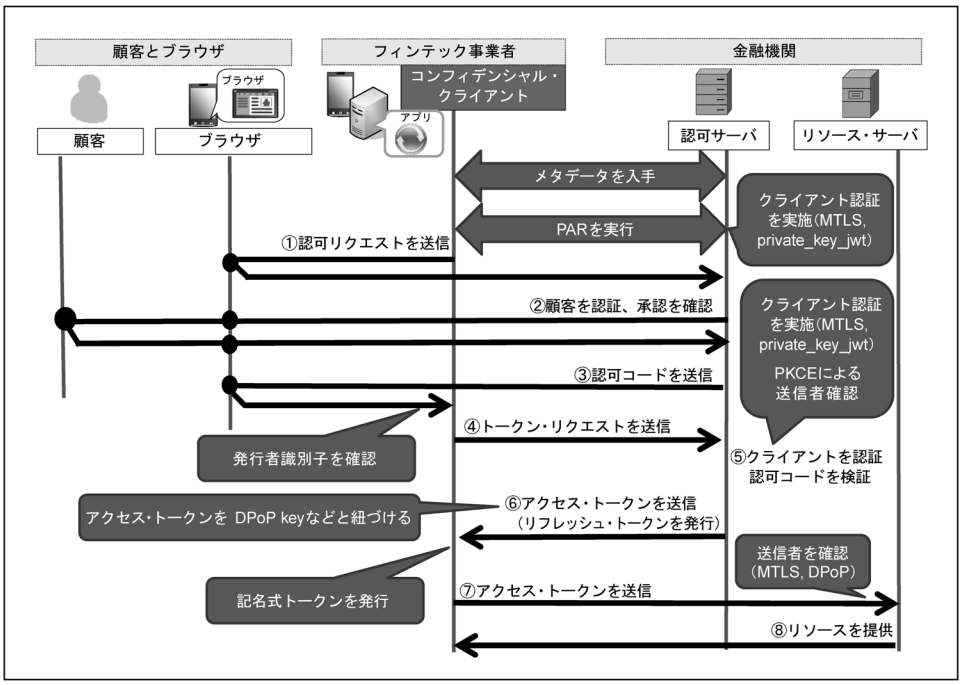
27 TLS 1.2 によって接続する場合には、条件として、TLS の安全な使用のための推奨事項（Sheffer, Holz, and Saint-Andre [2015]）に従うこと、特定の暗号アルゴリズムの組合せを使用することなどが定められている。

28 RFC 8414 は、認可サーバに関するメタデータ（認可サーバの識別子、認可リクエストやトークン・リクエストの送信先アドレス、認可プロトコルの処理フロー、サービス内容など）、メタデータのリクエストやレスポンスのメッセージ形式などを定めている（Jones, Sakimura, and Bradley [2018]）。OpenID Connect Discovery は、クライアントが OpenID Connect に基づく処理を実行する際に、認可サーバのメタデータを取得する方法を定めている（Sakimura *et al.* [2023]）。

29 MTLS（RFC 8705）は、サーバ認証に加えてクライアント認証も実行する TLS を指す（Campbell *et al.* [2020]）。クライアントが自分の電子証明書とデジタル署名を認可サーバに送信し、認可サーバが署名検証や電子証明書の有効性検証などを実施する。

30 private_key_jwt は、クライアントがデジタル署名を JWT に埋め込んで認可サーバに送信し、認可サーバは署名検証によってクライアントを認証する手法である（Sakimura *et al.* [2014]）。

図 3　FAPI 2.0 の主な要求事項を適用した認可コード・フロー



資料：Hosseyni, Küsters, and Würtele [2024] Figure 1

- クライアントが認可リクエストの内容を事前に認可サーバに登録する仕組み PAR（Pushed Authorization Request）[31] を用いる。認可サーバは登録時にクライアント認証を実施する。
- クライアントは認可レスポンス内の発行者識別子を用いて認可サーバを特定・検証する。
- 認可サーバは認可コードの送信者確認を PKCE（ハッシュ関数 SHA256 によるコード・チャレンジを使用）によって行う。
- 認可サーバは記名式トークン（持ち主が特定されているトークン）のみ発行する。
- リソース・サーバはアクセス・トークンの送信者確認を行う。その方法として、MTLS または DPoP（Demonstrating Proof of Possession）[32] を用いる。DPoP を使

31 PAR（RFC 9126）は、クライアントが認可リクエスト送信前にリクエスト内容など（PKCE のコード・チャレンジなども含まれる）を認可サーバに直接送信・登録する手法である（Lodderstedt *et al.* [2021]）。認可リクエスト時には、リクエスト内容などの情報の格納先を示す情報（Uniform Resource Identifier）とクライアントの識別子のみを送信する。これによって、リクエスト内容がブラウザなどから漏洩したり改変されたりするリスクを低くすることができる。

32 DPoP（RFC 9449）は、クライアントの署名検証鍵（DPoP key）を用いてアクセス・トークンの送信

用する場合、認可サーバは認可コードも DPoP key と紐づけする。
・認可サーバは顧客が認可の内容を正しく認識して認可処理を開始するように情報を適切に提供する。

(4) 各攻撃方法によるリスク軽減に寄与する主な要求事項

FAPI 2.0 の要求事項を満たすように OAuth 2.0 の認可コード・フローを実装したとする。その場合、3 節（2）の表 1 に示した各攻撃方法によるリスクの軽減に寄与するか否かを検討すると、表 4 に示すように、大半の攻撃方法によるリスクの軽減に寄与する。表 4 の「リスク軽減に寄与する FAPI 2.0 の主な要求事項」の列に記載されている要求事項（群）は、それぞれ対応する攻撃方法への対策となりうる要求事項を示しており、攻撃方法を防止するための十分条件ではない点に留意されたい。また、これらの要求事項を適用した際のリスク軽減の効果が十分か否かは、各サービスのビジネス要件に依存するため、一概にはいえない。

一部の攻撃方法については、リスクを軽減できない場合やその効果が状況に依存する場合がある。具体的には、不正なアプリの使用、セッション管理用パラメータの奪取、攻撃者のリソースの偽装、セッション管理用パラメータの悪用が挙げられる。これらについてそれぞれ以下で説明する。

イ. 不正なアプリの使用

これは、攻撃者が不正なアプリを顧客の端末にインストールさせ、そのアプリを用いて認可コードや顧客のリソースを奪取する攻撃である。FAPI 2.0 には不正なアプリを排除する機構が準備されておらず、FAPI 2.0 の要求事項を適用するだけではリスクを十分に軽減することが困難である。

対策としては、フィンテック事業者がアプリ配信サイトを限定する、アプリ配信サイトにおける不審なアプリを探索・排除する、正しいアプリの入手方法を顧客に説明して理解を得るといった対応が挙げられる[33]。

ロ. セッション管理用パラメータの奪取

これは、攻撃者がクライアントやそれが動作する端末の管理者権限を用いてセッ

者を確認する手法である（Fett *et al.* [2023]）。アクセス・トークンは DPoP key と紐づけされて発行される。リソース・サーバは、クライアントから、アクセス・トークンとともにクライアントのデジタル署名（DPoP Proof）と署名検証鍵を受信した後、署名検証、および、署名検証鍵とアクセス・トークンの対応関係を認可サーバへの問合せなどにより検証する。

33 これらに加えて、例えば、顧客の端末の OS による不正なアプリの使用を防止する機構（アプリのインストールの状態、通信内容、既知の不正なコードの有無などから不正なアプリか否かを分析・判断するなど）を活用するといった対応も考えられる。

表 4　各攻撃方法によるリスク軽減に寄与する FAPI 2.0 の主な要求事項

攻撃方法	リスク軽減に寄与する FAPI 2.0 の主な要求事項 （【 】：要求事項の効果）
① アクセス・トークンの推測	128 ビット以上のエントロピーをもつクレデンシャルの使用
② 通信経路上での盗聴	TLS（バージョンが 1.3 または条件付きで 1.2）の採用
③ クライアントからの奪取	・トークン送信者の確認 【アクセス・トークンの不正使用の検知】 ・クライアント認証 【リフレッシュ・トークンの不正使用の検知】
④ 転送機能の悪用	・オープン・リダイレクト（自動転送）の禁止 ・PAR【リダイレクト先の情報の事前登録と改変の検知】 ・トークン送信者の確認 【アクセス・トークンの不正使用の検知】
⑤ 中継サーバの自動接続機能の悪用	・PKCE【認可コードの不正使用の検知】 ・PAR【PKCE のコード・チャレンジの事前登録】 ・クライアント認証【認可コードの送信者の検証】 ・トークン送信者の確認 【アクセス・トークンの不正使用の検知】
⑥ 認可リクエストの改変	・メタデータの安全な提供 【正しいリソース・サーバなどの特定】 ・PAR【リダイレクト先の情報の事前登録と改変の検知】 ・発行者識別子の検証【認可コードの発行者の検証】 ・クライアント認証【認可コードの送信者の検証】 ・トークン送信者の確認 【アクセス・トークンの不正使用の検知】
⑦ 不正なアプリの使用	（不正なアプリの排除機構が準備されていない）
⑧ セッション管理用パラメータの奪取 （奪取した認可コードなどを用いてクライアントになりすますケース）	・PKCE【認可コードの不正使用の検知】 ・PAR【PKCE のコード・チャレンジの事前登録】 ・クライアント認証【認可コードの送信者の検証】 ・トークン送信者の確認 【アクセス・トークンの不正使用の検知】
⑨ 攻撃者のリソースの偽装 ⑩ セッション管理用パラメータの悪用	認可内容の情報の顧客への十分な提供 （効果は顧客による認可内容の理解度合いに依存）
⑪ 不正なサイトへのリダイレクト	・オープン・リダイレクト（自動転送）の禁止 ・PAR【リダイレクト先の情報の事前登録と改変の検知】

ション管理用のパラメータを奪取し、それを悪用して正規のセッションを乗っ取るというものである。最終的には、攻撃者は顧客のリソースを不正に入手する。

このタイプの攻撃のうち、攻撃者がセッションの乗っ取りによって認可コードやアクセス・トークンを入手したうえで認可サーバやリソース・サーバにそれらを提示するケースに関しては、FAPI 2.0 の要求事項（クライアント認証、PKCE、PAR、トークン送信者の確認）によって、攻撃者によるクライアントへのなりすましを検知することができる。ただし、攻撃者が、クライアントやそれが動作する端末の管理者権限を悪用し、セッション管理用パラメータの奪取とともに、他の処理（例えば、攻撃者への通信の実施）を実行するケースも想定される。このようなケースに対しては、上記の要求事項による対応だけではリスクを軽減できない可能性がある[34]。

対策としては、FAPI 2.0 の要求事項を満たすことに加えて、クライアントやそれが動作する端末において権限昇格につながる脆弱性を極力排除することが挙げられる。

ハ．攻撃者のリソースの偽装

これは、攻撃者が、攻撃者の口座の取引履歴（金融機関が保持しているもの）のアクセス先に顧客をフィッシングなどによって誘導し、顧客に自分の口座の取引履歴を入力させるなどして奪取するものである。複数の口座の取引履歴を同期させることができるケースでは、攻撃者は、自分の口座の取引履歴と顧客の口座の取引履歴を同期させるように顧客を誘導し、同期させた後で顧客の口座の取引履歴を入手することも想定される。

この攻撃では、正規の認可プロトコルが実行されるため、顧客が攻撃を検知しづらい。FAPI 2.0 では、認可の内容を顧客が正しく理解したうえで認可処理を実施するように、認可サーバが認可内容に関する情報を顧客に十分に提供することとしている。これに基づく対応によって、顧客が認可内容を正確に理解し、攻撃者による誘導に気づくことができたならば、リスク軽減に寄与する。

また、金融機関が複数の口座の取引履歴の同期を禁止する、フィンテック事業者が口座の取引履歴へのアクセス先に関する情報を顧客に対して適切に提供するといった対応もリスク軽減につながる。この攻撃方法によるリスクが高いと判断した場合には、上記の対応を組み合わせて実施することが有用である。

ニ．セッション管理用パラメータの悪用

これは、攻撃者が、クライアントやそれが動作する端末の管理者権限を用いて顧客のセッション管理用のパラメータを攻撃者のものに改変し、顧客を攻撃者の口座の取引履歴のアクセス先に誘導するというものである。この攻撃に関する FAPI

34 例えば、攻撃者が、リソース・サーバから顧客の口座の取引履歴を入手した際にそれを攻撃者のサーバに送信するようにクライアントの機能を改変するというシナリオが考えられる。この場合、クライアント認証やトークン送信者の確認などではリスク軽減が難しい。

2.0 の要求事項の効果や対応に関しては、攻撃者のリソースの偽装の場合と同様である。

5. おわりに

本稿では、オープン API における認可プロトコル OAuth 2.0 について、脅威や攻撃方法を既存研究に基づいて整理したほか、標準仕様などの要求事項が満たされておらず脆弱な実装が少なからず存在している状況を最近の研究成果を引用しつつ紹介した。また、脆弱性を排除する方法として、FAPI 2.0 の要求事項を認可プロトコルに適用した場合、大半の攻撃方法によるリスクの軽減に寄与することを示した。ただし、一部の攻撃方法に関しては、リスクを軽減できない場合やその効果が状況に依存する場合があり、追加的な対応が必要になることも説明した。

サイバーセキュリティの脅威は日々高度化しており、オープン API を標的とした攻撃も今後増える可能性がある[35]。そうしたなか、OAuth 2.0 に関連する標準仕様などの改訂が随時行われているほか、脆弱性や脅威に関する学術研究も進められている。オープン API のシステムのセキュリティを維持・向上させるためには、脅威の状況や標準仕様の動向を常に把握し、脆弱性を生じさせないように認可プロトコルを実装することが必要である。また、リスク評価を適宜実施し、許容できないレベルのリスクが生じている場合にはリスク軽減策を検討することも求められる。こうした対応を実施するうえで、FAPI 2.0 を活用することが有効である。今後、オープン API による安全なサービスが継続的に提供されることを期待したい。

35 Akamai Technologies, Inc. は、2023 年中に発生したインターネット上での攻撃全体の約 3 割が API を標的にしていた旨の調査結果を発表するとともに、攻撃が今後増加する可能性があるとの見方を示している（https://www.akamai.com/lp/soti/lurking-in-the-shadows、アクセス日：2024 年 5 月 27 日）。

参考文献

中村啓佑、「OAuth 2.0 に対する脅威と対策：金融オープン API の一段の有効活用に向けて」、『金融研究』第 37 巻第 3 号、日本銀行金融研究所、2018 年、111～141 頁

Alam, Mahbub Ul, Muhammad Abul Kalam Azad, and Md. Showkat Ali, "Best Practices to Secure API Implementations in Core Banking System (CBS) in Banks," *Proceedings of 2022 IEEE 12th Annual Computing and Communication Workshop and Conference*, IEEE, 2022, pp. 730–735.

Basel Committee on Banking Supervision, "Digitalisation of Finance," Bank for International Settlement, 2024 (available at https://www.bis.org/bcbs/publ/d575.pdf、2024 年 5 月 27 日).

Campbell, Brian, John Bradley, Nat Sakimura, and Torsten Lodderstedt, "OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens," RFC 8705, IETF, 2020 (available at https://www.rfc-editor.org/rfc/rfc8705.pdf、2024 年 5 月 27 日).

Denniss, William, John Bradley, Michael B. Jones, and Hannes Tschofenig, "OAuth 2.0 Device Authorization Grant," RFC 8628, IETF, 2019 (available at https://www.rfc-editor.org/rfc/pdfrfc/rfc8628.txt.pdf、2024 年 5 月 27 日).

Fernández, Gonzalo, Florian Walter, Axel Nennker, Dave Tonge, and Brian Campbell, "OpenID Connect Client-Initiated Backchannel Authentication Flow—Core 1.0," OpenID Foundation, 2021 (available at https://openid.net/specs/openid-client-initiated-backchannel-authentication-core-1_0.html、2024 年 5 月 27 日).

Fett, Daniel, "FAPI 2.0 Attacker Model," fapi-2_0-attacker-model-03, OpenID Foundation, 2022a (available at https://openid.net/specs/fapi-2_0-attacker-model.html、2024 年 5 月 27 日).

———, "FAPI 2.0 Security Profile," fapi-2_0-security-profile-03, OpenID Foundation, 2022b (available at https://openid.net/specs/fapi-2_0-security-profile.html、2024 年 5 月 27 日).

———, Brian Campbell, John Bradley, Torsten Lodderstedt, Michael B. Jones, and David Waite, "OAuth 2.0 Demonstrating Proof of Possession (DPoP)," RFC 9449, IETF, 2023 (available at https://www.rfc-editor.org/rfc/rfc9449.pdf、2024 年 5 月 27 日).

Hardt, Dick, "The OAuth 2.0 Authorization Framework," RFC 6749, IETF, 2012 (available at https://www.rfc-editor.org/rfc/pdfrfc/rfc6749.txt.pdf、2024 年 5 月 27 日).

Hosseyni, Pedram, Ralf Küsters, and Tim Würtele, "Formal Security Analysis of the OpenID FAPI 2.0: Accompanying a Standardization Process," Cryptology ePrint Archive, Paper 2024/078, International Association for Cryptologic Research, 2024 (available at https://eprint.iacr.org/2024/078.pdf、2024 年 5 月 27 日).

Jones, Michael B., Brian Campbell, and Chuck Mortimore, “JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants,” RFC 7523, IETF, 2015 (available at https://www.rfc-editor.org/rfc/pdfrfc/rfc7523.txt.pdf、2024 年 5 月 27 日).

———, and Dick Hardt, “The OAuth 2.0 Authorization Framework: Bearer Token Usage,” RFC 6750, IETF, 2012 (available at https://www.rfc-editor.org/rfc/pdfrfc/rfc6750.txt.pdf、2024 年 5 月 27 日).

———, Nat Sakimura, and John Bradley, “OAuth 2.0 Authorization Server Metadata,” RFC 8414, IETF, 2018 (available at https://www.rfc-editor.org/rfc/pdfrfc/rfc8414.txt.pdf、2024 年 5 月 27 日).

Lodderstedt, Torsten, John Bradley, Andrey Labunets, and Daniel Fett, “OAuth 2.0 Security Best Current Practice,” Internet Draft, draft-ietf-oauth-security-topics-27, IETF, 2024 (available at https://www.ietf.org/archive/id/draft-ietf-oauth-security-topics-27.html、2024 年 5 月 27 日).

———, Brian Campbell, Nat Sakimura, Dave Tonge, and Filip Skokan, “OAuth 2.0 Pushed Authorization Requests,” RFC 9126, IETF, 2021 (available at https://www.rfc-editor.org/rfc/rfc9126.pdf、2024 年 5 月 27 日).

———, Stefanie Dronia, and Marius Scurtescu, “OAuth 2.0 Token Revocation,” RFC 7009, IETF, 2013 (available at https://www.rfc-editor.org/rfc/pdfrfc/rfc7009.txt.pdf、2024 年 5 月 27 日).

———, Mark McGloin, and Phil Hunt, “OAuth 2.0 Threat Model and Security Considerations,” RFC 6819, IETF, 2013 (available at https://www.rfc-editor.org/rfc/pdfrfc/rfc6819.txt.pdf、2024 年 5 月 27 日).

OpenID Foundation, “Open Banking, Open Data and Financial-Grade APIs: A Whitepaper for Open Banking and Open Data Ecosystem Participants Globally,” Version: Final 1.0, OpenID Foundation, 2022 (available at https://openid.net/wordpress-content/uploads/2022/03/OIDF-Whitepaper_Open-Banking-Open-Data-and-Financial-Grade-APIs_2022-03-16.pdf、2024 年 5 月 27 日).

Philippaerts, Pieter, Davy Preuveneers, and Wouter Joosen, “OAuch: Exploring Security Compliance in the OAuth 2.0 Ecosystem,” *Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses*, Association for Computing Machinery, 2022, pp. 460–481.

———, ———, and ———, “Revisiting OAuth 2.0 Compliance: A Two-Year Follow-Up Study,” *Proceedings of 2023 IEEE European Symposium on Security and Privacy Workshops*, IEEE, 2023, pp. 521–525.

Sakimura, Nat, John Bradley, and Naveen Agarwal, “Proof Key for Code Exchange by

OAuth Public Clients," RFC 7636, IETF, 2015 (available at https://www.rfc-editor.org/rfc/pdfrfc/rfc7636.txt.pdf、2024 年 5 月 27 日).

———, ———, Michael B. Jones, and Edmund Jay, "OpenID Connect Discovery 1.0 Incorporating Errata Set 2," OpenID Foundation, 2023 (available at https://openid.net/specs/openid-connect-discovery-1_0.html、2024 年 5 月 27 日).

———, ———, ———, Breno de Medeiros, and Chuck Mortimore, "OpenID Connect Core 1.0 Incorporating Errata Set 1," OpenID Foundation, 2014 (available at https://openid.net/specs/openid-connect-core-1_0-errata1.html、2024 年 5 月 27 日).

Sheffer, Yaron, Ralph Holz, and Peter Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)," RFC 7525, IETF, 2015 (available at https://www.rfc-editor.org/rfc/pdfrfc/rfc7525.txt.pdf、2024 年 5 月 27 日).

補論 1. API に関するセキュリティ・リスク

The OWASP Foundation Inc.[36] は、API の開発主体やサービス提供主体が認識しておくべき主なセキュリティ・リスク 10 件を「Top 10 API Security Risks」として 2023 年に公表している[37]。リスクをその所在（リスクにつながる脆弱性が存在しうる箇所）に基づいて類型化すると表 A-1 のとおりである。認可プロトコルに関するリスクは、オブジェクトそのもの、オブジェクトのプロパティ、機能の 3 つのレベルに分けられている。

認証プロトコルのリスクについては、従来から金融機関が各種サービスのリスク

表 A-1　OWASP Top 10 API Security Risk (2023) で挙げられているリスク

リスクの所在		主なリスク
認可プロトコル	オブジェクト・レベル	オブジェクトそのものやその識別子の漏洩などによって、リソースへの不正アクセスが発生する。
	オブジェクト・プロパティ・レベル	API の応答からオブジェクトの属性情報が漏洩し、その悪用によってリソースへの不正アクセスが発生する。
	機能レベル	アクセス制御ポリシーの脆弱性（権限を確認しないなど）が悪用され、他のユーザによる API リクエストが横取りされたり改変されたりする。
認証プロトコル		認証プロトコルの脆弱性が悪用され、他のユーザになりすましされる。
API リソース管理		API へのリクエストの制御に不備があり、大量のデータを短時間で送信するなどの攻撃によって、API の処理能力が枯渇する。
業務フロー管理		API へのリクエストの制御に不備があり、短時間でサービス要求を繰り返し行うなどの攻撃によって、業務遂行に問題が発生する。
API のアクセス先管理		API の処理において不適切なサイトへのアクセスが実行され、アクセス先のサイトに関する情報が漏洩する。
API 構成管理		API の設定が不適切であり、他のユーザのリソースやシステムの重要な情報が漏洩する。
API インベントリ管理		API のインベントリ管理が不適切（API の属性情報が更新されないなど）であり、API 構成管理を適切に実施できない。
外部の API の管理		セキュリティ・レベルが低い外部の API にアクセスすることにより、API に関する重要な情報の漏洩、不正な処理、サービス妨害が発生する。

36 The OWASP Foundation Inc. は、ソフトウェアのセキュリティ向上を目的とするグローバルな非営利団体であり、セキュリティ向上に資する各種プロジェクトの支援、イベント開催などを通じた人材交流・コミュニティ形成の支援、セキュリティに関する啓発資料や情報の提供などを行っている。OWASP は Open Worldwide Application Security Project の略称であり、The OWASP Foundation Inc. の登録商標である。

37 これらのリスクは、https://owasp.org/API-Security/editions/2023/en/0x11-t10/（アクセス日：2024 年 5 月 27 日）において公表されている。

のレベルに応じた適切な認証手段を選択・実装してきた。API における処理フローにおいて、顧客やフィンテック事業者をどのように認証するかについても、これまでの延長線上で検討することができる。

API リソース管理や業務フロー管理のリスクについても、従来の業務継続に関する取組みを、API の計算資源の確保や関連業務の継続に拡張してリスク軽減策を検討することになる。

API 構成管理のリスクに関しては、API 開発プロセスにおけるコード・レビューや各種テストの実施を通じて、不適切な設定を検知・修正する対応が考えられる。ソフトウェア開発一般で実施されているものであり、API 開発においても同様に対応することとなる。また、API を FAPI 2.0 ベースで実装する場合、OpenID Foundation による適合性テストや認証スキームを活用することもできる。

API インベントリ管理のリスクについては、金融機関が API のインベントリを適切に構築・管理していない場合、それらのシステムの構成管理に支障が生じ脆弱性への対応などが適切に行われない可能性がある。システムの構成要素やそのバージョン、API を通じて提供される情報とその重要度、リスク軽減策の実施状況など、通常のシステムにおけるインベントリと同様の管理が必要となる。

外部の API の管理のリスクについては、金融機関による API 接続先管理の一環として対応することが想定される。

補論 2. 主な脅威の概要

(1) アクセス・トークンの奪取

① アクセス・トークンの推測

攻撃者が、自分のリソースにアクセスするためのアクセス・トークンを自分の端末などから入手したうえで、それを分析して顧客のリソースにアクセスするためのアクセス・トークンを推定する。サーバ・アプリの場合には、クライアントのサーバ（アクセス・トークンを格納）などから不正に入手する。

② 通信経路上での盗聴

攻撃者が、クライアントと認可サーバとの間の通信、または、クライアントとリソース・サーバとの間の通信を盗聴・解析してアクセス・トークンを入手する。

③ クライアントからの奪取

攻撃者が、クライアントや端末・サーバの脆弱性を悪用してそれらの管理者権限を入手し、それを用いてアクセス・トークンやリフレッシュ・トークンをクライアントから奪取する。

④ 転送機能の悪用

クライアントが特定の URL にアクセス・トークンを自動的に転送する機能（オープン・リダイレクト）を有効化しているケースにおいて、攻撃者が、リダイレクト先の URL を自分のサーバのものに改変し、アクセス・トークンを自分のサーバに転送させる。

⑤ 中継サーバの自動接続機能の悪用

顧客の端末が認可サーバなどとの通信に中継サーバ（プロキシ）を使用しているケースにおいて、攻撃者が、プロキシの自動設定ファイルを悪用して認可コードやアクセス・トークンをプロキシから自分のサーバに送信させる。

⑥ 認可リクエストの改変

攻撃者が、認可リクエストにおいて、認可サーバやリソース・サーバの IP アドレスを自分のサーバのものに変更し、認可コードやアクセス・トークンをクライアントから攻撃者のサーバに送信させる。

(2) セッションの奪取

⑦ 不正なアプリの使用

攻撃者が、不正なネイティブ・アプリを顧客の端末にインストールさせ、ブラウザとクライアントの間のセッションに中間者として侵入し、認可コードや顧客の口座の取引履歴を奪取する。

⑧ セッション管理用パラメータの奪取

攻撃者が、クライアントやそれが動作する端末の管理者権限を奪取し、それを用いてセッション管理用のパラメータを入手する。入手したパラメータによって正規のセッションを引き継ぎ、顧客の口座の取引履歴を奪取する。

(3) 顧客の誘導

⑨ 攻撃者のリソースの偽装

攻撃者が、フィッシングなどによって、攻撃者の口座の取引履歴へのアクセス先に顧客を誘導し、顧客の口座の取引履歴を入力させる。また、複数の口座の取引履歴を同期させることができる場合には、顧客の口座の取引履歴と攻撃者の口座の取引履歴を同期させるように顧客を誘導し、攻撃者は同期後に顧客の口座の取引履歴を入手する。

⑩ セッション管理用パラメータの悪用

攻撃者が、クライアントやそれが動作する端末の管理者権限を奪取してセッション管理用のパラメータを改変し、攻撃者の口座の取引履歴のアクセス先に顧客を誘導する。そのうえで、顧客に自分の口座の取引履歴を入力させる。顧客が口座の取引履歴を入力したら、攻撃者はそれにアクセスする。また、複数の口座の取引履歴を同期させることができる場合には、顧客の口座の取引履歴と攻撃者の口座の取引履歴を同期させるように顧客を誘導し、同期後に顧客の口座の取引履歴を入手する。

(4) 顧客のクレデンシャルの奪取

⑪ 不正なサイトへのリダイレクト

攻撃者が、フィッシング・サイトの URL をクライアントのアクセス先として認可サーバに登録し、フィッシング・メールなどによって不正な認可リクエストを顧客に開始させる。それを受信した認可サーバは、エラー・

メッセージなどをブラウザに返信するとともに、ブラウザを登録済の URL（フィッシング・サイト）にリダイレクトさせる。顧客がそのサイト上で ID やパスワードを入力すると、攻撃者がそれらを用いて顧客になりすまし、正規の認可プロトコルを経て顧客の口座の取引履歴を入手する。