OAuth2.0 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

なかむらけいすけ
中村啓佑

要　旨

金融関連分野でオープン API（Application Programming Interface）の利用促進に向けた動きが加速するなか、認可プロトコルの一種である OAuth2.0 に熱い視線が向けられている。しかし、OAuth2.0 は、大枠としてのフレームワークと組み合わされるオプション群を規定したものに過ぎず、実装を誤ると、さまざまな脅威に対して脆弱になることが知られている。本稿では、先行研究を参照しつつ、OAuth2.0 に起因する典型的な脆弱性を紹介するとともに、現時点でとりうるセキュリティ対策についても解説する。

キーワード：　オープン API、セキュリティ、認可プロトコル、FinTech、OAuth2.0

本稿の作成に当たっては、OpenID Foundation 理事長の崎村夏彦氏および富士通研究所の野田敏達氏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。

中村啓佑　日本銀行金融研究所企画役補佐
（現システム情報局企画役補佐、E-mail: keisuke.nakamura@boj.or.jp）

1. はじめに

近年、情報技術を活用した新しい金融サービス（FinTech）の開発・普及の起爆剤として、オープン API（Application Programming Interface）の整備に向けた動きが、官民を挙げて加速している[1]。銀行法等の一部を改正する法律（2017 年 5 月 26 日成立）には、銀行および口座情報サービスや決済指図伝達サービスを提供する電子決済等代行業者に対して、オープン・イノベーションを進めていくために必要な項目が新たに規定された[2]。銀行に対しては、例えば、オープン API にかかる体制整備を行うなどの努力義務が課された。また、電子決済等代行業者は、登録が必須となった[3]。内閣府［2017］も、オープン API の活用による FinTech の促進に向け、2020 年 6 月までに 80 以上の銀行がオープン API を導入することを目標に据えた。

こうしたなか、オープン API の具体的な活用に向けた検討会等が次々と立ち上げられ、その成果が公表されている。例えば、オープン API のあり方に関する検討会における検討結果（全国銀行協会［2017］）、金融機関における FinTech に関する有識者検討会における報告書（金融情報システムセンター［2017a］）や、同検討会の下部組織として設置された API 接続先チェックリストワーキンググループによる API 接続チェックリスト（試行版、金融情報システムセンター［2017b］）がそれである。特に、API 接続チェックリストでは、金融機関や電子決済等代行業者に求められるセキュリティ対策案が示されている。具体的には、金融機関に対し、API を介して提供されるデータや機能の範囲を制御するための代表的なプロトコル OAuth2.0 の仕組み等を理解し、それらが正しく実装および運用されていることを確認することなどが求められている。

OAuth2.0 を用いれば、利用者、FinTech 企業等の中間業者、金融機関のそれぞれが、セキュリティ面やシステムの維持管理面において、多くのメリットを享受できる。例えば、中間業者は、利用者の ID やパスワードを管理する必要がなくなる。

1 API とは、銀行以外の者が銀行のシステムに接続し、その機能を利用できるようにするためのプログラムを指し、このうち、銀行が FinTech 企業等の中間業者に API を提供し、利用者の同意に基づいて、銀行システムへのアクセスを許諾する形態をオープン API という（金融審議会［2016］）。オープン API の形態については、中村［2017］に詳しい。

2 口座情報サービスは、利用者が、（複数の）金融機関における自分の口座残高等のデータを集計し、確認できるサービスを指す。また、決済指図伝達サービスは、決済指図を金融機関に伝達し、その結果を確認できるサービスをいう。

3 電子決済等代行業者を登録制とした点を踏まえると、日本におけるオープン API は、欧州と同様のメンバー API を想定したものになると考えられる。メンバー API とは、規範性を有する一定の取決めを遵守することを条件にコミュニティへの加入を認め、コミュニティ・メンバーのみに API の利用を認めるものである（European Banking Association Working Group on Electronic Alternative Payments [2016]、中村［2017］）。

利用者や金融機関にとっても、必要なデータのみを開示するように制御できるようになり、現状よりもセキュリティが強化されることとなる（中村［2017］）。

OAuth2.0 は、2012 年に IETF（Internet Engineering Task Force）で標準化された RFC 6749（Hardt [2012]）、RFC 6750（Jones and Hardt [2012]）、RFC 6755（Campbell and Tschofenig [2012]）を含む複数の技術仕様からなるプロトコルであり、既にさまざまなサービスで活用されている[4]。しかし、そうしたサービスのなかには、OAuth2.0 に起因するセキュリティ上の脆弱性が指摘されているものも多い。例えば、Google Play や Apple マーケット等の公式のアプリケーション配信サイトで提供されていた 149 のモバイルアプリ（OAuth2.0 が使われていたもの）を調査したところ、脆弱性を有するものが約 6 割（89 モバイルアプリ）に及ぶとの結果が報告されている（Chen *et al.* [2014]）。また、Facebook や Google 等からダウンロードされた 182 のモバイルアプリを調査したところ、約 4 割（85 モバイルアプリ）が脆弱性を有していたとの報告もある（Yang, Lau and Liu [2016]）。OAuth2.0 にはさまざまなメリットがあるが、金融分野で使用する際には、事前にセキュリティに関する検討を詳細に行い、安全に利用できることを確認しておく必要があるといえよう[5]。

以下、本稿では、2 節で OAuth2.0 のプロトコルの仕組みを概説する。そのうえで、3 節では OAuth2.0 にかかる脅威について説明する。さらに、4 節ではそれらの脅威へのセキュリティ対策について説明し、5 節では、金融機関や中間業者における今後の対応や課題について考察する。

2. OAuth2.0 の概要

(1) 認可とそのプロトコル

認可（authorization）とは、エンティティやプログラムにアクセス権を与えることを指す（電子情報通信学会［2004］）。オープン API の文脈でいえば、例えば、中間業者 A が利用者 B の銀行 C にある口座情報にアクセスする場合、利用者 B が銀行 C にある自分の口座情報へのアクセス権を中間業者 A に事前に与えること（認可）が必要となる。認可は、通信相手やメッセージの正当性（本来の状態であるこ

4 IETF は、インターネット技術の標準化を推進する任意団体であり、各種の技術仕様を RFC（Request for Comments）として標準化している。

5 金融機関のシステムで適切に OAuth2.0 を実装するためには、実務者向けの詳細なガイドラインが重要となるが、こうしたガイドラインの整備については、金融 ISAC（Information Sharing and Analysis Center）が設置した FinTech セキュリティ WG（Working Group）において検討が進められている。

と）を確認する認証（authentication）とは異なる[6]。

オープン API における認可プロトコルとして、金融業界では、OAuth2.0 が推奨されることが多い。例えば、英国の Open Data Institute は、OAuth2.0 または OpenID Connect を推奨している（Open Data Institute [2016]）。米国の FS-ISAC （Financial Services-Information Sharing and Analysis Center）も OAuth2.0 を推奨している（FS-ISAC [2015]）[7]。また、わが国でも、全国銀行協会［2017］が OAuth2.0 を推奨しているほか、金融情報システムセンター［2017b］においても、OAuth2.0 を念頭に置いた記載がなされている。

OAuth2.0 は、あらゆる環境に適用できるように、大枠としてのフレームワークとそこで組み合わされるオプション群のみを規定している。このため、実際のシステムで使用する際に必要な具体的な設定内容に関する記述は非常に少ない。したがって、OAuth2.0 をサービス内容や環境に応じてどう実装するかは、金融機関や中間業者の判断に委ねられている[8]。適切に実装するためには、OAuth2.0 を正確に理解するとともに、個々のアプリケーションが晒される可能性のある脅威を適切に想定する必要がある。

（2） OAuth2.0 のフロー

RFC 6749 には、OAuth2.0 の中心的な役割を果たすエンティティとして、次の 5 つが記載されている（図表 1 参照）。すなわち、①利用者（リソース・オーナー）、②リソース・オーナーが用いる端末のブラウザ等（ユーザ・エージェント）、③中間業者が提供するアプリケーション（クライアント）、④認可を行うサーバ（認可サーバ）、⑤保護コンテンツ（口座情報や振込機能等）を格納するサーバ（リソース・サーバ）である。

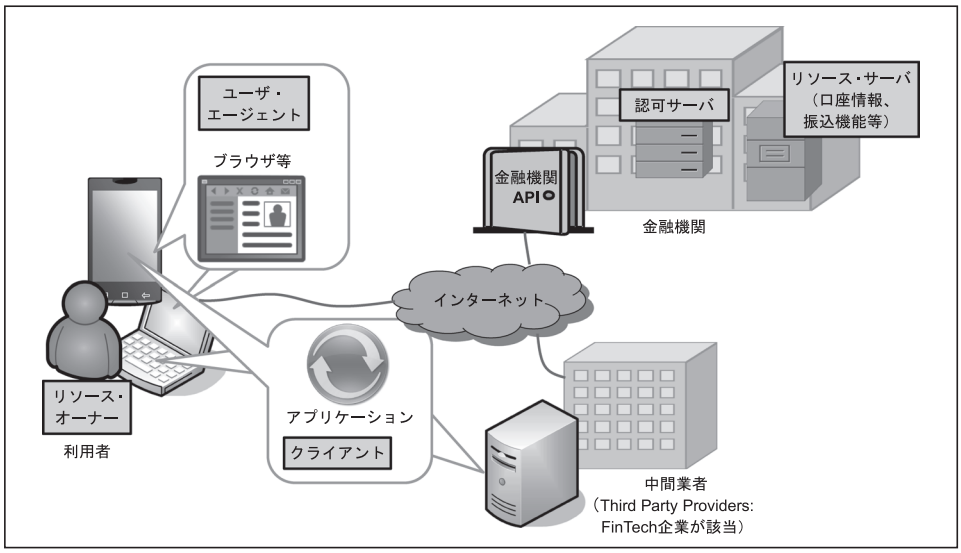
OAuth2.0 のプロトコルとしては、①認可コード方式、②インプリシット方式、③リソース・オーナー・パスワード・クレデンシャル方式、④クライアント・クレデンシャル方式の 4 種類が規定されている。これらは、クライアントが認可の内容を示すデータ（アクセス・トークンと呼ばれる）を用いてリソース・サーバにアクセスする点で共通しているが、アクセス・トークンの入手方法が異なっている。本稿では、①の認可コード方式を取り上げる。同方式は、リソース・オーナーが自分

6 認証は、確認する内容によって、相手認証やメッセージ認証に分類される（電子情報通信学会［2004］）。

7 FS-ISAC は、世界的な金融業界のサイバー犯罪および物理的脅威情報の分析と共有を目的として、1999 年に米国に設立された非営利団体である。

8 OAuth2.0 を実装する際には、さまざまなパラメータや設定値を具体的に設定する必要があり、それらの一覧をプロファイルと呼ぶ。金融機関や中間業者は、サービス内容等に応じた脅威や対策方針を決定したうえで、具体的な対策やそのために必要なプロファイルを決定していくことになる。

図表 1　RFC 6749（OAuth2.0 のフレームワーク）のエンティティ



の ID やパスワードを中間業者に提供する必要がないため、安全性が高いと考えられる[9]。

認可コード方式における処理のフロー（認可フロー）は、以下のとおりである（図表 2 参照）。

①クライアントが、ユーザ・エージェントを介して、リソース・サーバへのアクセスを要求するリクエストを認可サーバに送信する[10]。

②認可サーバは、ユーザ・エージェントの画面を介して、リソース・サーバへのアクセスをクライアントに許可すること（認可を与えること）をリソース・オーナーに確認する。リソース・オーナーは、それに同意する場合には、承認する旨を送信する。

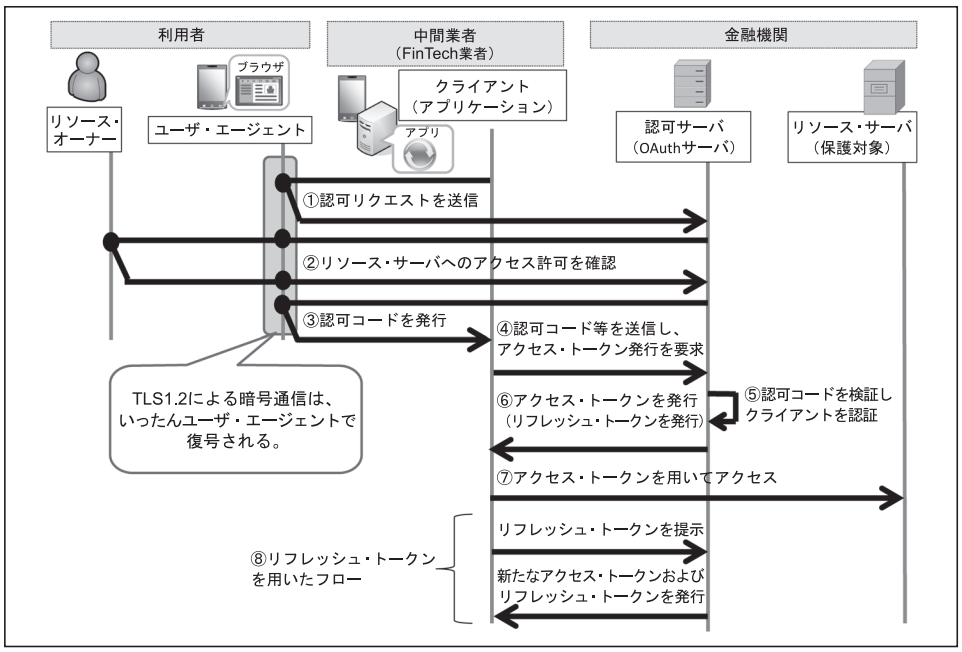
③認可サーバは、ユーザ・エージェントを経由して、アクセス・トークンの発行に必要なデータを格納した認可コードをクライアントに発行する[11]。

9 リソース・オーナー・パスワード・クレデンシャル方式とクライアント・クレデンシャル方式は、アクセス・トークン発行時に、リソース・オーナーの ID やパスワード（クレデンシャルと呼ばれる）を中間業者に提供する方式である。また、インプリシット方式は、アクセス・トークンが漏洩する可能性が相対的に高い方式である。RFC 6749 は、認可コード方式が利用できる場合には、これらの方式を推奨していない。このため、保護コンテンツについて機密性や完全性が要求される金融分野のサービスでは、認可コード方式以外の方式が推奨される状況は考えにくい。

10 リソース・オーナーがクライアントを起動する処理は、上記①の処理の前に行われる。この処理については、RFC 6749 に記載がない。

11 実際のシステムでは、上記②と③の間に、リソース・オーナーに対する相手認証が金融機関によって行われるが、これについては RFC 6749 に記載がない。

図表２　認可コード方式における処理フローの概要（概念図）



④クライアントは、認可コードに加えて、クライアントが保有している秘密情報（クライアント・シークレット）を認可サーバに送信し、アクセス・トークンの発行を要求する。

⑤認可サーバは、認可コードを検証するとともに、クライアント・シークレットを用いてクライアント認証を行う[12]。

⑥認可サーバは、上記⑤が成功した場合、アクセス・トークンを発行する。

⑦クライアントはアクセス・トークンを用いてリソース・サーバにアクセスする。

⑧リフレッシュ・トークンを用いる場合、クライアントは、それを認可サーバに提示し、新しいアクセス・トークンおよび新しいリフレッシュ・トークンの発行を受ける。

認可サーバは、リフレッシュ・トークンをアクセス・トークンとあわせて発行できる。リフレッシュ・トークンは、アクセス・トークンの有効期限が切れた際、認可サーバに送信して新しい有効期限のアクセス・トークンや、予め与えられた範囲よりも少ない範囲で有効なアクセス・トークンの発行を要求する際に用いられる。これにより、アクセス・トークンの再発行の際に、上記①～⑤のフローを再度行う

12　RFC 6749 は、アクセス・トークン発行時のクライアント認証を必須のプロセスと位置付けていない。しかし、セキュリティの観点から考えると、クライアント認証は、必須とせざるを得ない。このため、本稿では、クライアント認証を実施することを前提とする。

必要がなくなり利便性が向上するほか、不必要に大きな権限のアクセス・トークンの使用を回避し、データ利用の範囲の最小化に寄与する。

上記のフローにおいて留意事項が 2 つ存在する。第 1 に、上記③の認可サーバとクライアントの間の通信は、基本的に TLS（Transport Layer Security）1.2 によって暗号化されているものの、唯一、中継するユーザ・エージェントにおいていったん復号されるという問題がある。これによって、認可コードがユーザ・エージェントにおいて奪取されるなどの脅威が生じるので、このための対策が求められる（詳細は 4 節）。第 2 に、アクセス・トークンが持参人払式トークンであることに起因するリスクに留意する必要がある（Jones and Hardt [2012]）。持参人払式トークンは、認可の対象となるクライアントを特定する情報を含まず、トークンを提示するすべてのエンティティにサービスが提供される。このため、奪取されると容易に悪用されてしまう。こうした脅威への対策として、クライアントを特定する情報を含む記名式トークンを利用する方法が検討されている（Jones *et al.* [2017]、Campbell *et al.* [2017]）。

3. OAuth2.0 に対する脅威

本節では、OAuth2.0 に対する脅威をテーマとした RFC や関連する学術論文（Lodderstedt, McGloin, and Hunt [2013]、Lodderstedt, Bradley, and Labunets [2017]、Fett, Küsters, and Schmitz [2016]）を参照し、それらに記載されている主な脅威のなかから、OAuth2.0（認可コード方式）を用いたシステムに当てはまるものを抽出し、整理する。

(1) OAuth2.0 を用いたシステムと攻撃者等の前提

OAuth2.0 を用いたシステムと攻撃者等に関して、以下の前提を置く[13]。

- 情報資産は、リソース・サーバに格納されている保護コンテンツおよびリソース・サーバ上で実行できる機能とする。
- クライアントは、中間業者において動作するケース（サーバアプリ）とリソース・オーナーの端末で動作するケース（ネイティブアプリ）を想定する[14]。

13 一般的なマルウェアによる攻撃等、OAuth2.0 の認可フローに固有とはいえない攻撃への対策については、ネットワーク経由でのサービス提供において当然求められるべきものであり、ここではそうした対策が実施されていることを前提として議論を進める。

14 ネイティブアプリでは、通常、クライアント・シークレットが固定値となる。このため、攻撃者はダ

- 攻撃者は、リソース・オーナーにかかる保護コンテンツ（例えば、口座情報等の参照系と呼ばれる情報）を不正に閲覧したり、同サーバで実行できる機能（例えば、口座振込機能等の更新系と呼ばれる機能）を悪用したりして、リソース・オーナーの口座から不正送金等を試みるものとする。
- 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こととする。また、内部者による攻撃は想定しない。
- 攻撃者は、クライアントおよびそれが稼動する端末・サーバの特権の奪取を試みることとする。ただし、暗号化や相手認証、署名に用いる秘密鍵や署名の検証時に用いる公開鍵等のセキュリティ機能に関するものを除く[15]。また、中間業者の内部者による攻撃は想定しない。
- 攻撃者は、リソース・オーナーの端末およびユーザ・エージェントの特権（暗号化や相手認証等のセキュリティ機能に関するものを除く）の奪取を試みることとする。

(2) 想定される脅威

攻撃者が保護コンテンツにアクセスする方法として、大きく次の 3 つが想定される。すなわち、アクセス・トークンの奪取、保護コンテンツにアクセスするセッション等の奪取、攻撃者の保護コンテンツを介したアクセスである。以下では、これらをさらに細分化して説明する（図表 3 参照）[16]。

ウンロードしたアプリに対してリバース・エンジニアリングを行うなどして、クライアント・シークレットを把握し、これを用いた認証を無効化できる。ここでは、認可サーバへのアクセスの都度、乱数等を生成して認可サーバとの間で共有する方式で、クライアント認証と類似した仕組みとして標準化された PKCE（Proof Key for Code Exchange）を採用することとする（Sakimura, Bradley, and Agarwal [2015]）。PKCE については補論 1 を参照されたい。サーバアプリの場合、端末から直接サーバアプリを実行する構成に加えて、サーバアプリへのアクセスのみを行うアプリケーション（アクセスアプリ）を用いてサーバアプリを実行する構成もある。

15 暗号化や相手認証等のセキュリティ機能を無効化するマルウェアへの対策については、5 節（2）を参照されたい。

16 本稿では、OAuth2.0 で規定されていない、リソース・オーナーに対する相手認証を検討対象外としているが、実装が不適切な場合、認証で用いられるデータ（ID、パスワード等）が奪取される可能性がある。具体的には、直前のセッションで送信されたデータを後続のセッションに引き継ぐ設定となっているケース（リダイレクトのステータス・コードを 307 とする）で発生する（307 リダイレクト攻撃と呼ばれる）。対策として、中間業者は、直前のセッションで送信されたデータを後続のセッションに引き継がない設定とする（リダイレクト・ステータス・コードを 303 とする）ことが考えられる。そのほか、正規の認証画面のボタンの上に（不正な処理を実行させるための）透明なボタンを被せる攻撃も考えられる（クリックジャッキング攻撃と呼ばれる）。対策として、金融機関、中間業者は、ユーザ・エージェントにおいて、HTML（HyperText Markup Language）文書の中に別の HTML 文書を埋め込む機能（iframe という）を許可しない設定にすることが考えられる。

図表 3　想定される主な脅威

主な脅威	具体的な攻撃方法
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クライアント等からの奪取
	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ざん
	⑦認可コード等の奪取
	A. 認可コード等の推測
	B. 通信経路上での盗聴
	C. クライアントからの認可コード等の奪取
	D. 転送機能の悪用
	E. 中継サーバの自動接続機能の悪用
	F. リソース・サーバやクライアントへのなりすまし
保護コンテンツにアクセスするセッション等の奪取	⑧クライアント等へのなりすまし
	⑨セッション管理用パラメータの改ざん
	⑩取引内容の改ざん
攻撃者の保護コンテンツを介したアクセス	⑪攻撃者の保護コンテンツの偽装
	⑫セッション管理用パラメータの悪用

イ．アクセス・トークンの奪取

アクセス・トークンを奪取する攻撃として、主に 7 つの方法が考えられる（図表 3 ①〜⑦）。

①アクセス・トークンの推測

リソース・オーナー以外のアクセス・トークン（例えば、攻撃者自身のアクセス・トークン）を用いてアクセス・トークンを推測する。攻撃者自身のアクセス・トークンは、ネイティブアプリの場合、自分の端末から入手することが考えられる。サーバアプリの場合は、アクセス・トークンを保持するサーバ等の脆弱性を悪用し、何らかの特権を用いて入手することが考えられる[17]。

②通信経路上での盗聴

クライアントと認可サーバの間の通信やクライアントとリソース・サーバの間の通信を盗聴・解析してアクセス・トークンを奪取する。

③クライアント等からの奪取

クライアントやそれが動作する端末・サーバの脆弱性等を悪用し、特権を用い

17 アクセス・トークンがさまざまな領域（例えば、ログファイル）に格納される可能性がある点に留意する必要がある。

てクライアント等からアクセス・トークンを奪取する。

④転送機能の悪用

指定したURLにクライアントがアクセス・トークンを自動転送する機能（オープン・リダイレクタと呼ばれる）を有効としていた場合に、ユーザ・エージェントまたはクライアントが稼動している端末・サーバの特権を用いて上記URLを改ざんし、攻撃者のサーバのアドレスにアクセス・トークンを転送させる。

⑤中継サーバの自動接続機能の悪用

リソース・オーナーの端末が中継サーバ（Proxy）を利用している場合に、中継サーバの自動設定ファイル（PACファイル）を悪用し、アクセス・トークンを含むURL情報を攻撃者のサーバに送信させる（Kotler and Klein [2016]）[18]。

⑥認可リクエストの改ざん

ユーザ・エージェントがクライアントに送信する認可リクエストを改ざんし、リソース・サーバのIPアドレスを攻撃者のサーバのものに変更することによって、アクセス・トークンを攻撃者のサーバに送信させる[19]。

⑦認可コード等の奪取

認可コード、リフレッシュ・トークン、クライアント・シークレット、PKCEの鍵セット（認可コード等）を奪取し、それらを用いてアクセス・トークンを奪取する。その具体的な方法としては、主に次の6つ（A〜F）が考えられる。

A. 認可コード等の推測

リソース・オーナー以外の認可コード等（例えば、攻撃者が正規のリソース・オーナーとして認可フローを行って入手したもの）を用いて、リソース・オーナーの認可コード等を推測する。攻撃者自身の認可コード等は、ネイティブアプリの場合、攻撃者自身の端末から入手することが考えられる。サーバアプリの場合には、認可コードがユーザ・エージェント経由で発行される際に入手することが考えられる。

奪取した認可コード等を用いてアクセス・トークンを入手する方法として、攻撃者が正規の認可フローを実行しつつ、認可コードを（奪取したものに）置き換えることが考えられる（認可コードの置換）[20]。また、奪取したリフレッシュ・トークンとクライアント・シークレットを認可サーバに送信し、アクセス・トークンを要求することも想定される（リフレッシュ・トークンの悪用）。認可コードの置換とリフレッシュ・トークンの悪用は、以下のB〜Fにおいても適用される可能性がある。

18 URL情報を攻撃者のサーバに送信するフローは、図表2⑦で発生する。こうした攻撃はWPAD／PAC（Web Proxy Auto-Discovery Protocol/Proxy Auto-Config）攻撃と呼ばれる。

19 こうした攻撃（IdP Mix-Up Attackと呼ばれる）は、認可コード方式の範囲外で実行されると考えることもできるが、OAuth2.0のフレームワーク内に含まれるため、ここに記述することとした。

20 認可コードを置き換える攻撃はコード・インジェクション攻撃と呼ばれる。

B. 通信経路上での盗聴

クライアントと認可サーバの間の通信や、クライアントとリソース・サーバの間の通信を盗聴・解析し、認可コード等を奪取する。

C. クライアントからの認可コード等の奪取

クライアントやそれが動作する端末・サーバの脆弱性等を利用し、特権を用いて、クライアントから認可コード等を奪取する。

D. 転送機能の悪用

オープン・リダイレクタが有効となっている場合に、クライアントが動作する端末・サーバの特権を用いて、上記 URL を攻撃者のサーバに改ざんし認可コード等を奪取する。

E. 中継サーバの自動接続機能の悪用 クライアントが動作する端末・サーバが中継サーバを利用している場合に、中継サーバの PAC ファイルを改ざんし、認可コード等を含む URL 情報を攻撃者のサーバに送信するようにする。

F. リソース・サーバやクライアントへのなりすまし

サーバアプリの場合、DNS（Domain Name System）や ARP（Address Resolution Protocol）を悪用し、リソース・サーバやクライアントの IP アドレスを攻撃者のサーバのものに改ざんし、サーバアプリから認可コード等を送信させる[21]。

ロ. 保護コンテンツにアクセスするセッション等の奪取

保護コンテンツにアクセスするセッション等を奪取する主な攻撃方法として、クライアント等へのなりすましとセッション管理用パラメータの改ざんが考えられる（図表 3 ⑧〜⑩）。

⑧クライアント等へのなりすまし

ネイティブアプリの場合は、偽物のネイティブアプリをリソース・オーナーの端末にインストールさせておき、それをリソース・オーナーに実行させ、攻撃者のサーバに接続させる[22]。サーバアプリの場合は、端末からクライアントにアクセスする際に用いられる URL を改ざんしたり、偽物のアクセスアプリを端末にインストールさせておき、それをリソース・オーナーに実行させたりすることによって、攻撃者のサーバに接続させる。こうした方法により、リソース・オーナーが正規のアクセス・トークンを用いて行うセッションが、攻撃者を経由して

21 DNS は、インターネット上のホスト名等に使われるドメイン名と IP アドレスの紐付けを管理するシステムである。ARP は、IP アドレスから MAC（Media Access Control）アドレスを得るプロトコルである。

22 PKCE は、クライアントが正規のネイティブアプリである場合にのみ機能するものであり、偽のネイティブアプリの場合は別の対策が必要になる。

行われることとなる。その結果、攻撃者はリソース・オーナーにかかる保護コンテンツにアクセスできる。

⑨セッション管理用パラメータの改ざん

クライアントまたはそれが動作する端末・サーバの特権を用いて、リソース・オーナーのセッション管理用のパラメータ（ステート・パラメータ）を奪取し、リソース・オーナーのセッションを引き継ぐことによって、保護コンテンツにアクセスする。

⑩取引内容の改ざん

リソース・オーナーの端末の特権を用いて、ユーザ・エージェント上で、認可リクエストの通信内容もしくは認可コード発行の通信内容（例えば、振込先や振込金額等）を改ざんし、不正な内容の取引を実行させる[23]。

ハ．攻撃者の保護コンテンツを介したアクセス

攻撃者の保護コンテンツを介したアクセスによる主な攻撃方法としては、攻撃者の保護コンテンツの偽装とセッション管理用パラメータの悪用が挙げられる（図表3 ⑪、⑫）。

⑪攻撃者の保護コンテンツの偽装

リソース・オーナーが攻撃者の保護コンテンツにアクセスするように誘導し、保護コンテンツとして格納すべきリソース・オーナーの情報を、攻撃者がアクセスできる領域に入力させ、当該情報を奪取する。

異なるリソース・オーナーの保護コンテンツ間の同期が認められている場合には、次の攻撃も考えられる[24]。つまり、フィッシング・メール等によってリソース・オーナーを攻撃者の保護コンテンツにアクセスさせた後、それをリソース・オーナーの保護コンテンツと同期するよう誘導する。リソース・オーナーが同期したとき、攻撃者は自分の保護コンテンツからリソース・オーナーの保護コンテンツにアクセスできるようになる。

⑫セッション管理用パラメータの悪用

クライアントまたはそれが稼動する端末・サーバの特権を用いて、リソース・オーナーのステート・パラメータを攻撃者のステート・パラメータに改ざんし、

23 たとえクライアントに入力した値（例えば、振込先や振込金額）の確認画面が認可サーバやリソース・サーバ等で実行される前に表示されたとしても、内容の改ざんにリソース・オーナーが気付かなかったり、オーバーレイ攻撃（正規画面に偽の画面を被せて表示させる攻撃）によって改ざんを見抜けなかったりした場合には、こうした攻撃が発生する。

24 例えば、金融機関が、夫婦間の口座情報を同期させるサービスを提供している場合が考えられる。

リソース・オーナーを攻撃者の保護コンテンツにアクセスさせる[25]。そのうえで、上記⑪と同様に、リソース・オーナーが保護コンテンツとして格納すべき情報を、攻撃者がアクセスできる領域に入力するように誘導するという攻撃や、異なるリソース・オーナーの保護コンテンツ間の同期が認められている場合には、リソース・オーナーの保護コンテンツを攻撃者の保護コンテンツと同期するように誘導するという攻撃が考えられる。

4. 各脅威への主な対策

ここでは、3 節（2）で示した脅威への対策と、各対策を有効なものとするために、リソース・オーナーに求められる対応について説明する。その際、3 節冒頭で参照した文献に加えて、Sakimura, Bradley, and Jay [2017] も参照しつつ説明する（図表 4 参照）。同文献は、認可プロトコルとして OAuth2.0 や OpenID Connect を利用する際のセキュリティ・プロファイルである Financial API について規定しているものであるが、OAuth2.0 にかかるセキュリティ対策についても記載している（Financial API の概要については補論 2 を参照）。

(1) 共通の対策

すべての脅威に共通する基本的な対策として、次の 3 点を挙げることができる。第 1 に、金融機関が、アクセス・トークンごとに、アクセスを許可するサービス項目を中間業者に提示することである（対策 1）。その際、中間業者は、リソース・オーナーがサービス項目を確認できるようにすることが求められる。

第 2 に、リソース・オーナーがアクセス・トークンを取り消すことができるようにすることである（対策 2）。リソース・オーナーがサービス項目を確認した際に、想定していたサービス項目が提示されたものと異なっていたとする。このような場合、リソース・オーナーがアクセス・トークンを取り消すことができる仕組みを中

25 ここでは、リソース・オーナーは、アクセス先が攻撃者の保護コンテンツとなっているにもかかわらず、自分の保護コンテンツにアクセスしていると誤って認識している。その際、クライアント自体はリソース・オーナーの保護コンテンツへのアクセスをリクエストする場合（Naïve RP Session Integrity Attack と呼ばれる）のほか、クライアントがリソース・オーナーとアクセス先の保護コンテンツの対応関係をチェックせず、攻撃者の保護コンテンツへのアクセスをリクエストする場合が考えられる。後者の攻撃は、CSRF 攻撃（Cross-Site Request Forgeries Attack）と呼ばれる。これらは厳密にいえば、認可コード方式の範囲外で実行されるものと考えることもできる。もっとも、OAuth2.0 のフレームワーク内に含まれるとも考えられることから、ここに記述することとした。

図表 4　3 つの脅威への主な対策

主な脅威	攻撃方法（括弧内は攻撃場所）	主な対策（【】内は実施者）
アクセス・トークンの奪取	①アクセス・トークンの推測*（クライアントの端末・サーバ）	【金融機関】暗号学的に強固な乱数等を利用（対策 4）。
	②通信経路上での盗聴（アクセス・トークンの通信路）	【金融機関・中間業者】TLS1.2 で暗号化（対策 5）。
	③クライアント等からの奪取*（クライアントとその端末・サーバ）	【中間業者】ログファイル等への認可コード等の書込み・保管を禁止（対策 6）。
	④転送機能の悪用*（クライアントとその端末・サーバ）	【金融機関・中間業者】アクセス・トークン送信先によるオープン・リダイレクタの利用を禁止（対策 7）、またはリダイレクト URI を適切に設定・検証（対策 8）。
	⑤中継サーバの自動接続機能の悪用（クライアントとその端末・サーバ）	【金融機関・中間業者】対策 5 に加え、認可コード等を URL 情報に含めない設定（対策 9）とともに、送信先サーバを認証（対策 10）。
	⑥認可リクエストの改ざん（ユーザ・エージェントと認可サーバのセッション）	【金融機関・中間業者】対策 8 に加え、アクセス・トークンに認可サーバの情報を埋込み・検証（対策 11）。
	⑦認可コード等の奪取および特権利用によるリフレッシュ・トークンの悪用（クライアントとその端末・サーバ）	【金融機関】認可コードの有効期間や利用回数の適切な設定（対策 12）、リフレッシュ・トークンの有効期間や利用回数の適切な設定（対策 13）とともに、アクセス・トークン送信時のクライアントを認証（対策 14）、または PKCE を実行（対策 15）。 【金融機関・中間業者】対策 8、15 を実施。さらに、特権利用の場合、リフレッシュ・トークンを発行しない（対策 16）、またはサーバアプリを採用（対策 17）。
	A.　認可コード等の推測*（クライアントの端末・サーバ）	【金融機関】対策 4 を実施。
	B.　通信経路上での盗聴*（認可コード等の通信路）	【金融機関・中間業者】対策 5 を実施。
	C.　クライアントからの認可コード等の奪取*（クライアント）	【金融機関・中間業者】対策 6 を実施。
	D.　転送機能の悪用*（クライアント）	【金融機関・中間業者】対策 7 または 8 を実施。
	E.　中継サーバの自動接続機能の悪用（クライアントとその端末・サーバ）	【金融機関・中間業者】対策 5、9、10 を実施。
	F.　リソース・サーバやクライアントへのなりすまし（クライアントの端末・サーバ、DNS や ARP）	【金融機関・中間業者】対策 8 を実施。
保護コンテンツにアクセスするセッション等の奪取	⑧クライアント等へのなりすまし（端末）	【中間業者・リソース・オーナー】サーバアプリはクライアント等に適切に配信・インストール（対策 18）、ネイティブアプリは対策 18 のほか、対策 17 も検討。
	⑨セッション管理用パラメータの改ざん*（クライアントとその端末・サーバ）	【金融機関・中間業者】ステート・パラメータの検証（対策 19）に加え、その使用を 1 回のみとし、ステート・パラメータの判別を制御（対策 20）。
	⑩取引内容の改ざん*（端末）	【金融機関・中間業者】認可リクエスト等に署名を付け、受信側で署名を検証（対策 21）
攻撃者の保護コンテンツを介したアクセス	⑪攻撃者の保護コンテンツの偽装（端末）	【金融機関・中間業者】対策 19 に加え、リソース・サーバでの保護コンテンツ間の同期を禁止（対策 22）。 【中間業者】リソース・オーナーに通信内容を確認（対策 23）。
	⑫セッション管理用パラメータの悪用*（クライアントとその端末・サーバ）	【金融機関・中間業者】対策 19、22 を実施。 【中間業者】対策 20 を実施。

備考：1.「*」の攻撃には特権が利用される場合がある。
　　2. 全脅威に対する共通の対策 1～3 は記載していない。

間業者が提供することが考えられる[26]。

第 3 に、アクセス・トークンの有効期間と利用範囲を適切に設定することである（対策 3）。有効期間は、サービスの特性に応じて、適切な値（例えば、1 回のサービス利用の時間を数分から数時間程度とする）に設定するほか、サーバにおけるセキュリティ対策のレベルが同一の保護コンテンツに利用範囲を限定することが考えられる。重要度が高い保護コンテンツを利用範囲に含める場合、最もレベルが高いセキュリティ対策を行うことが考えられる。

(2) アクセス・トークンの奪取への対策

①アクセス・トークンの推測 に対しては、金融機関は、アクセス・トークン等を生成する際に暗号学的に強固な擬似乱数を用いる（Eastlake and Schiller [2005]）ことが挙げられる（対策 4）。複数の金融機関が同一の認可サーバを使用する場合（例えば、共同センターの利用）、金融機関を識別する値（識別子）をアクセス・トークンとともにリソース・オーナーに送信することが考えられる。

②通信経路上での盗聴 に対しては、金融機関と中間業者は、TLS1.2（あるいは、標準化が完了すれば TLS1.3）を利用し、通信経路上のデータを暗号化することが挙げられる（対策 5）。Sheffer, Holz, and Saint-Andre [2015] は、TLS1.2 等で利用する暗号に関して、RSA 暗号の場合は鍵長を 2,048 ビット以上とし、楕円曲線暗号の場合は鍵長を 160 ビット以上とすることを求めている。

③クライアント等からの奪取 に対しては、中間業者は、ログファイル等への認可コード等の書込み・保管を禁止することが挙げられる（対策 6）。これを実施できない場合には、認可コード等が書き込まれる領域へのアクセスを厳格に制御するなどの対応が必要となる[27]。

④転送機能の悪用 に対しては、中間業者は、リダイレクト URI（Uniform Resource Identifier）で示されるトークンの送信先においてオープン・リダイレクタの使用を

26 中間業者のクライアントがマルウェアに感染すると、リソース・オーナーによる取消しの処理が適切に実行されないことも想定される。そのため、金融機関は、リソース・オーナーから直接取消しの申請を受け付ける仕組みについても検討しておく必要がある。

27 認可コード等、機密性が高いデータを格納する領域については、サーバアプリの場合、サーバのセキュリティ対策として、エス・キュー・エル・インジェクション攻撃への対策等を実施することが求められる。さらに、サービスの脆弱性を排除するために、定期的なペネトレーション・テストの実施、セキュリティ・パッチの適用、ファイアウォールや IPS（Instruction Prevention System）機器の導入、適切な監視を行うことにより、特権が奪取された場合にも直ちに検知し、通信を遮断するなどの対策を講じることが求められる。ネイティブアプリの場合は、リソース・オーナーの端末の特権を奪取して行われることから、リソース・オーナーにおけるセキュリティ対策が必要となる（4 節（5）参照）。ただし、適切な監視等を行うことができないため、サーバアプリに比べて、脅威が顕在化する可能性が高い。

禁止することが考えられる（対策 7）。あるいは、金融機関と中間業者は、通信にかかるリダイレクト URI（絶対パス）を事前に登録するほか、そのリダイレクト URI を、クライアントや認可サーバごとに異なるように、論理的に分離して設定し、その URI が完全に一致することを検証する（対策 8）。

⑤中継サーバの自動接続機能の悪用 に対しては、金融機関と中間業者が TLS1.2 を用いてデータの暗号化を実施する対策 5 に加えて、次の 2 つを実施することが求められる。金融機関と中間業者は、認可コードや各種トークンの送受信の際に、それらが（URL にではなく）本文に含まれるように実装する（対策 9）。これは、中継サーバになりすました攻撃者に URL を把握されるケースを想定した対策である。さらに、金融機関と中間業者は、クライアントまたは認可サーバが不正なサーバに認可コードや各種トークン等を送信しないように、送信先のサーバを認証する（対策 10）。

⑥認可リクエストの改ざん に対しては、対策 8 に加えて、金融機関と中間業者は、認可サーバにかかる情報をアクセス・トークンに埋め込み、クライアントにおいて検証することが求められる（対策 11）[28]。

⑦認可コード等の奪取 にかかる 6 つの攻撃方法（図表 4 ⑦ A～F）に対しては、これらに共通する（金融機関による）対策として、次の 4 つを実施することが必要である。

第 1 に、認可コードの有効期間や利用回数を適切に設定する（対策 12）。例えば、有効期間を数分程度に設定することや（有効期間は短ければ短い方がよい）、利用回数を 1 回のみに限定することが考えられる[29]。

第 2 に、リフレッシュ・トークンを利用する場合には、その有効期間や利用回数を適切に設定する（対策 13）。有効期間は、リソース・オーナーの利便性を低下させない範囲で短く設定するほか、発行する都度、リフレッシュ・トークンを一意に識別するための値を変化させることも必要である[30]。

28 認可サーバにかかる情報としては、認可サーバの識別子でもあるリダイレクト URI を用いることが考えられる。

29 1 つの認可コードが複数回送信されるケースについては何らかの対策が必要となる。例えば、複数回送信された認可コードを用いて生成されたアクセス・トークンやリフレッシュ・トークンを無効化することも考えられる。ただし、認可コードを複数回送信して可用性を低下させる攻撃により、利便性の低下につながるリスクに留意する必要がある。そのほか、中間業者のプログラムの不具合等の可能性も考えられることから、中間業者に対して金融機関から警告を発したり、リソース・オーナーに対してアクセス・トークンの取消し（対策 2）を検討するように促したりすることなども考えられる。

30 1 つのリフレッシュ・トークンが複数回送信されるケースについては何らかの対策が必要となる。例えば、それに基づいて発行されたアクセス・トークンやリフレッシュ・トークンをすべて無効化する仕組み（リフレッシュ・トークン・ローテーションと呼ばれる）を導入することが考えられる。ただし、リフレッシュ・トークンを複数回送信して可用性を低下させる攻撃が発生しうることに留意する必要がある。そのほか、中間業者のプログラムの不具合等の可能性も考えられることから、中間業者に対して金融機関から警告を発したり、リソース・オーナーに対してアクセス・トークンの取消し（対策 2）を検討するように促したりすることなども考えられる。

第 3 に、クライアントがアクセス・トークンを送信した際に、クライアント・シークレットを検証し、クライアントを認証する（対策 14）。こうした対策の代わりに、PKCE を用いることも考えられる（対策 15）。PKCE を用いる場合、ハッシュ関数として SHA-256（Secure Hash Algorithm 256bit）を用いるほか、解読されないように既知の攻撃手法への対策を別途講ずる必要がある。

第 4 に、奪取された認可コード等を悪用されないように、認可コードの置換やリフレッシュ・トークンの悪用への対策を実施することが考えられる。認可コードの置換に対しては PKCE を利用する対策 15、リフレッシュ・トークンの悪用に対してはリダイレクト URI を設定・検証する対策 8 を実施する。

攻撃者がクライアントおよびその端末・サーバの特権を利用する場合には、リフレッシュ・トークンとクライアント・シークレット（または PKCE の鍵セット）が認可サーバに送信され、アクセス・トークンが奪取される可能性がある。これに対しては、リフレッシュ・トークンを発行しない（対策 16）という対応や、クライアントとしてサーバアプリを採用するとともに、ネットワーク上の通信の監視を強化することが考えられる（対策 17）[31]。

以上の共通対策に加えて、A.〜F. の各攻撃に固有の対策を講ずることが求められる。すなわち、A. 認可コード等の推測に対しては、暗号論的に強固な疑似乱数を利用する対策 4 を実施する。B. 通信経路上の盗聴に対しては、金融機関と中間業者が TLS1.2 を用いてデータの暗号化を実施する対策 5 が有効である。C. クライアントからの認可コード等の奪取に対しては、金融機関と中間業者がログファイル等への認可コードの書込み・保管を禁止する対策 6 を講ずる。D. 転送機能の悪用に対しては、金融機関と中間業者がトークンの送信先（トークン・エンドポイント）にオープン・リダイレクタを利用させない対策 7、あるいは、リダイレクト URI を設定・検証する対策 8 を行う。E. 中継サーバの自動接続機能の悪用に対しては、金融機関と中間業者が TLS1.2 を用いてデータの暗号化を実施する対策 5 に加えて、金融機関と中間業者が、認可コード等を URL 情報に含めない設定を採用する対策 9 や、送信先サーバのドメインを認証する対策 10 を実施することが挙げられる。F. リソース・サーバやクライアントへのなりすましに対しては、金融機関と中間業者がリダイレクト URI を設定・検証する対策 8 を行えばよい。

31 例えば、サーバにおいて予期しない動作や不正なログインが検知された場合、それらの通信を直ちに遮断するなどの対応が考えられる。

(3) 保護コンテンツへのアクセスが可能なセッション等の奪取への対策

⑧クライアント等へのなりすまし に対しては、中間業者とリソース・オーナーは、一般的なオンラインでのサービスの利用・提供時と同様に、クライアント等への配信・インストールを適切に実施することが求められる（対策 18）。

例えば、サーバアプリの場合は、ブラウザのお気に入り等に当該サーバの URL を事前に登録し、サービスを利用する際に、必ずお気に入り等からアクセスすることが考えられる。中間業者は、サーバアプリを用いる場合、アクセスアプリを公式のアプリケーション配信サイトから提供するようにすることが望ましい。また、中間業者が提供するアプリと類似したものが公式のアプリケーション配信サイトから提供されていないかを確認することも考えられる。

ネイティブアプリの場合は、リソース・オーナーは、クライアントをダウンロードする際に、公式のアプリケーション配信サイトから行うことが考えられる。仮にリソース・オーナーが誤って偽のネイティブアプリをダウンロードし、そのアプリが PKCE を使用した場合、認可サーバではネイティブアプリが正規であるか否かを判別することができない。このため、金融機関と中間業者は、リソース・オーナーが対策 18 を十分に行うことができないと仮定する場合には、クライアントとしてサーバアプリを採用する対策 17 を行うことが考えられる。

⑨セッション管理用パラメータの改ざん に対しては、金融機関と中間業者は、セッションを管理するためのステート・パラメータを検証する必要がある（対策 19）。さらに、中間業者は、ステート・パラメータの利用を 1 回のみとしたうえで、通信データのフォーマットの一部（ステート・パラメータが含まれる部分）が、正規の受信者以外には判別できないように設定することが求められる（対策 20）[32]。

⑩取引内容の改ざん に対しては、クライアントが送信する認可リクエストおよび認可サーバが送信する認可コード等の発行の通信にそれぞれ署名を付け、受信側においてそれを検証することが有効である（対策 21）[33、34]。

32 こうした通信データのフォーマットの設定については、ワールド・ワイド・ウェブ・コンソーシアムが提案しているポリシー（W3C [2016]）を参照するとよい。

33 署名付きのトークンを利用する方式として、JSON Web Signature（JWS）が知られている（Jones, Bradley, and Sakimura [2015]）。具体的には、中間業者および金融機関が事前にそれぞれの公開鍵を共有したうえで、中間業者から金融機関への通信（図表 2 ①の通信）、および、金融機関から中間業者への通信（図表 2 ③の通信）において JWS を適用する。なお、図表 2 ③の通信に JWS を適用する際は、OAuth2.0 の拡張オプション（ID トークン）を用いる（de Medeiros *et al.* [2014]）。

34 署名を付けて送信することにより、金融機関と中間業者の責任分界点を明確にできる。具体的には、金融機関や中間業者から送信された内容が変更されていた場合（例えば、攻撃者による攻撃以外に、プログラムのバグによっても生じうる）、署名を利用して変更された箇所を検証できる。

(4) 攻撃者の保護コンテンツを介したアクセスへの対策

⑪攻撃者の保護コンテンツの偽装 に対しては、金融機関と中間業者は、ステート・パラメータを検証する対策 19 に加えて、リソース・サーバ上での保護コンテンツ間の同期を禁止することが必要である（対策 22）。さらに、中間業者は、クライアントが認可サーバやリソース・サーバにアクセスする都度、通信の内容（ログインしようとするユーザ名等）を、ユーザ・エージェントを介してリソース・オーナーに確認させるようにすることが求められる（対策 23）。

⑫セッション管理用パラメータの悪用 に対しては、金融機関と中間業者は、ステート・パラメータを検証する対策 19 に加えて、リソース・サーバ上での保護コンテンツ間の同期を禁止する対策 22 が必要である。さらに、中間業者は、ステート・パラメータの利用を 1 回のみとしたうえで、それが漏えいしないように、通信データのフォーマットの一部（ステート・パラメータが含まれる部分）が、正規の受信者以外には判別できないように設定する対策 20 が求められる。

(5) リソース・オーナーに求められる対応

ここまで主に金融機関と中間業者における対策を整理してきたが、リソース・オーナーの対応も重要である。具体的には、主に以下の 3 点が求められる。

第 1 に、端末等を第三者に盗取されることがないように適切に管理するほか、その起動時や中間業者のクライアントの使用時に求められる相手認証にかかるレガシー情報等（リソース・オーナーの ID、パスワード、生体情報等）を適切に管理することが求められる。

第 2 に、端末等の OS のパッチ適用やマルウェア対策ソフトの利用等、通常の端末等におけるセキュリティ対策や、ネイティブアプリやアクセスアプリのパッチ適用も速やかに行うことが求められる。

第 3 に、サービスにおけるリスクとそれへの対策について、金融機関や中間業者に確認し、その効果を把握しておくことが重要である。具体的には、使用するサービスの信頼性（不正なアプリでないことの確認方法）、中間業者が実施しているマルウェア対策や署名等を用いた改ざん防止策とそれらの効果について、サービス開始時および利用時に確認することが望ましい。

5. むすびにかえて：金融機関等における対応の留意点や今後の課題

本稿では、OAuth2.0 を用いたサービスを提供する際に想定される脅威を 12 種類に大別し、それらへの主な対策を示した。こうした脅威と対策を踏まえ、最後に、金融機関や中間業者が対策を検討していく際の留意点や攻撃の高度化への備えとして必要と考えられる対応について考察する。

(1) 対策を検討・実施していくうえでの留意点

本稿では、OAuth2.0 を用いたサービスを提供する際に起こりうる主な脅威や対策について、RFC 等の既存の文献を参照しつつ整理した。もっとも、多様なサービス形態や接続形態が想定されるなかで、本稿で説明されていない脅威が存在する可能性もある。したがって、金融機関や中間業者は、本稿で取り上げた脅威や対策に加えて、本稿で説明されていない脅威の有無についても検討し、必要な対策を講ずることが求められる。

また、本稿では、各脅威に対して主なセキュリティ対策を網羅的に示しており、それらを実施した際の利便性の低下等については検討の対象外としている。具体的には、セキュリティ対策の実施に伴う通信の処理能力の低下、リソース・オーナーに複雑な処理や過度の確認を強いることによる利便性の低下等が考えられる[35]。提供するサービスに求められるセキュリティを個別に判断したうえで、リソース・オーナーが許容する以上に利便性が低下しないように、対策を講ずることが重要である。

今後、金融機関によるオープン API の提供が本格化するにつれて、リソース・オーナーも増加することが予想される。金融機関や中間業者が安全性を確保するための対応を進めることに加えて、リソース・オーナーの側でも、適切な対応が一段と求められることになる。中間業者においては、リソース・オーナーによるセキュリティ対策や端末等の管理が適切に実施されるように、金融機関と密に連携し、サービスにかかるリスクの所在やそのインパクト、実際のサービスにおけるセキュリティの状況等について、リソース・オーナーの理解を得るよう丁寧に説明していくことが求められる。例えば、金融情報システムセンターや金融 ISAC（Information Sharing and Analysis Center）のオープン API を利用するためのチェックリスト等を

35 例えば、リフレッシュ・トークンを発行しないなどの対策が該当する。

活用して、十分なセキュリティが確保された状態にあることや、リスク管理上留意しなければならない事項を継続的に確認し、リソース・オーナーに情報還元する体制等を整備することが考えられる。また、こうした対応の実効性を高めるために、リスクが顕在化した際の責任を関係者間でどのように分担するかについても予め明確にしておくことが有益であろう。

(2) 今後の攻撃の高度化とそれへの対応

4 節で説明した各対策（図表 4 で示したもの）を実施することにより、図表 3 で整理した攻撃方法に対して一定の耐性を確保できると考えられる。しかし、今後、リソース・オーナーの端末等において、データの暗号化や相手認証等のセキュリティ機能を悪用するマルウェアが出現する可能性もある。また、クライアントを提供する中間業者のサーバに、標的型攻撃等によって、同様のマルウェアが混入する可能性もある。こうした点を踏まえると、先行き、4 節で説明した対策だけでは、十分な安全性を確保できなくなるおそれがある。特に、高い機密性や完全性が求められるサービス（例えば、更新系のサービスを利用するものなど）を金融機関および中間業者が提供する場合には、こうしたリスクに十分に目配りしていくことが必要である。

例えば、クライアントが動作している端末のアクセス制御が無効化され、クライアントに保存されたアクセス・トークン、認可コード、クライアント・シークレット（PKCE の情報を含む）が奪取されることが想定される。さらに、正規のクライアントが悪用され、アクセス・トークンと正規のクライアント（正規のクライアント・シークレットを含む）を用いて、リソース・オーナーの保護コンテンツにアクセスされる可能性もある。リフレッシュ・トークンを利用できる場合には、アクセス・トークンの有効期限が切れた後も、リフレッシュ・トークンと正規クライアントを用いて、新規のアクセス・トークンを発行させ、それを悪用する可能性もある。

こうした攻撃への対策として、次の 2 つの方法が考えられる。第 1 に、アクセス・トークンによってサービスの提供を開始する前に、OAuth2.0 を用いたチャネルとは別の（セキュアな）チャネルを用いて認証を実行する方法がある（セカンド・チャネル認証と呼ばれる）。例えば、スマートフォンで OAuth2.0 の認可フローの処理を実施し、セカンド・チャネル認証をパソコンによって実行することが考えられる。あるいは、1 台のスマートフォンのみで実行する場合には、OAuth2.0 の認可フローを通常の実行環境で実行するとともに、セカンド・チャネル認証を SIM（Subscriber Identity Module）等のセキュア・エレメントを用いて実行するという方法もありうる。

第 2 に、取引の処理フローにおける行為やリソース・オーナーの端末に表示され

る内容（参照系の情報や最終実行前の振込情報等）の奪取・改ざんを一段と困難にする方法がある。具体的には、OAuth2.0 で用いられる各種トークンについて、現在検討が進められている記名式トークン（Token Binding、Mutual TLS 等）を採用することなどが想定される。

今後、こうした対策についても順次検討が進むことが期待される。新しい対策やそれにかかる技術も活用しつつ、オープン API による革新的なサービスが、安心安全に広く利用されるようになることを期待したい。

参考文献

金融情報システムセンター、「金融機関における FinTech に関する有識者検討会報告書」、金融情報システムセンター、2017 年 a

――――、「API 接続チェックリスト（試行版）」、金融情報システムセンター、2017 年 b

金融審議会、「金融制度ワーキング・グループ報告書（平成 28 年 12 月 27 日公表）」、金融庁、2016 年

全国銀行協会、「オープン API のあり方に関する検討会報告書―オープン・イノベーションの活性化に向けて」、全国銀行協会、2017 年

電子情報通信学会、『情報セキュリティハンドブック』、オーム社、2004 年

内閣府、「平成 29 年第 10 回経済財政諮問会議・第 10 回未来投資会議合同会議、資料 7 未来投資戦略 2017　Society 5.0 の実現に向けた改革（平成 29 年 6 月 9 日開催）」、内閣府、2017 年

中村啓佑、「金融分野の TPPs と API のオープン化：セキュリティ上の留意点」、『金融研究』第 36 巻第 3 号、日本銀行金融研究所、2017 年、83～110 頁

Campbell, Brian, John Bradley, Nat Sakimura, and Torsten Lodderstedt, “Mutual TLS Profile for OAuth 2.0 draft-ietf-oauth-mtls-03,” Internet Engineering Task Force, 2017.

――――, and Hannes Tschofenig, “An IETF URN Sub-Namespace for OAuth,” Request for Comments 6755, Internet Engineering Task Force, 2012.

Chen, Eric, Yutong Pei, Shuo Chen, Yuan Tian, Robert Kotcher, and Patrick Tague, “OAuth Demystified for Mobile Application Developers,” *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communication Security*, ACM New York, 2014, pp. 892–903.

Denniss, William, and John Bradley, “OAuth 2.0 for Native Apps draft-ietf-oauth-native-apps-12,” Internet Engineering Task Force, 2017.

Eastlake, Donald E, and Jeffrey I. Schiller, “Randomness Requirements for Security,” Request for Comments 4086, Internet Engineering Task Force, 2005.

European Banking Association Working Group on Electronic Alternative Payments, “Understanding the Business Relevance of Open APIs and Open Banking for Banks,” European Banking Association, 2016.

Fett, Daniel, Ralf Küsters, and Guido Schmitz, “A Comprehensive Formal Security Analysis of OAuth 2.0,” *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ACM New York, 2016, pp. 1204–1215.

Financial Services-Information Sharing and Analysis Center, “Durable Data API,” Financial Services-Information Sharing and Analysis Center, 2015.

Hardt, Dick, “The OAuth 2.0 Authorization Framework,” Request for Comments 6749, Internet Engineering Task Force, 2012.

Jones, Michael B., John Bradley, Brian Campbell, and William Denniss, "OAuth 2.0 Token Binding draft-ietf-oauth-token-binding-04," Internet Engineering Task Force, 2017.

———, ———, and Nat Sakimura, "JSON Web Signature," Request for Comments 7515, Internet Engineering Task Force, 2015.

———, and Dick Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage," Request for Comments 6750, Internet Engineering Task Force, 2012.

Kotler, Itzik, and Amit Klein, "Crippling HTTPS with unholy PAC," blackhat USA 2016, 2016.

Lodderstedt, Torsten, John Bradley, and Andrey Labunets, "OAuth Security Topics draft-ietf-oauth-security-topics-02," Internet Engineering Task Force, 2017.

———, Mark McGloin, and Phil Hunt, "OAuth 2.0 Threat Model and Security Considerations," Request for Comments 6819, Internet Engineering Task Force, 2013.

de Medeiros, Breno, Marius Scurtescu, Paul Tarjan, and Michael B. Jones, "OAuth 2.0 Multiple Response Type Encoding Practices," OpenID Foundation, 2014.

Open Banking, "Open Banking forms collaboration with OpenID Foundation," Open Banking, 2017.

Open Data Institute, "The Open Banking Standard," Open Data Institute, 2016.

Sakimura, Nat, John Bradley, and Naveen Agarwal, "Proof Key for Code Exchange by OAuth Public Clients," Request for Comments 7363, Internet Engineering Task Force, 2015.

———, ———, and Edmund Jay, "Financial API," OpenID Foundation, 2017.

———, ———, Michael B. Jones, and Breno de Madeiros, "OpenID Connect Core 1.0 incorporating errata set 1," OpenID Foundation, 2014.

Sheffer, Yaron, Ralph Holz, and Peter Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)," Request for Comments 7525, Internet Engineering Task Force, 2015.

Yang, Ronghai, Wing Cheong Lau, and Tianyu Liu, "Signing into One Billion Mobile App Accounts Effortlessly with OAuth2.0," blackhat Europe 2016, 2016.

World Wide Web Consortium, "Referrer Policy," World Wide Web Consortium, 2016.

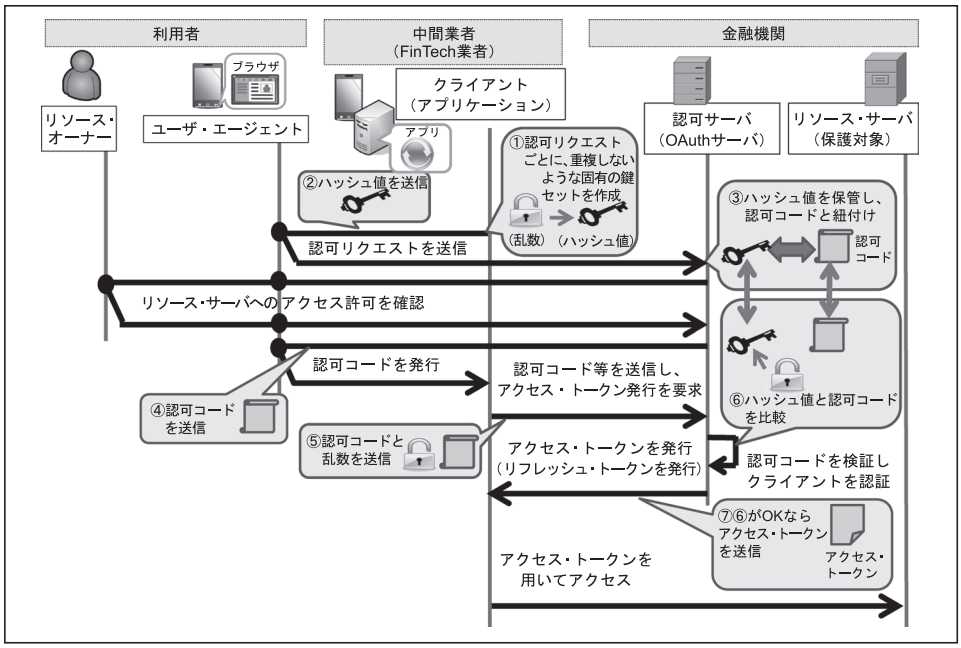
補論 1.　PKCE の概要

PKCE は、クライアント・シークレットを用いたクライアント認証を実施できない場合、それと類似した仕組みを提供するものとして標準化された。例えば、クライアントが正規のネイティブアプリであり、すべてのクライアントに対して同一のデータをクライアント・シークレットとして格納している場合に利用する。PKCE を適用することにより、クライアント・シークレットとは別の手法でクライアントを確認することが可能となる。また、現在では、コード・インジェクション攻撃への対策としても利用されている。

RFC 7636 に規定されている PKCE の処理フローは以下のとおりである（Sakimura, Bradley, and Agarwal [2015]、図表 A-1 参照）。

①クライアントは、認可リクエストを行う都度、重複しないような固有の鍵セット（乱数とそのハッシュ値）を生成する。
②クライアントは、認可リクエストを送信する際に、鍵セットのうちハッシュ値を認可サーバに送信する。
③認可サーバは、上記②で受信したハッシュ値を保管するとともに、認可コードを生成して、それらを紐付ける。

図表 A-1　PKCE のフロー（概念図）



④認可サーバは、リソース・サーバへのアクセス許可をリソース・オーナーに確認した後、上記③で生成した認可コードを、ユーザ・エージェント経由でクライアントに送信する。
⑤アクセス・トークンを入手する際、クライアントは、上記④で受信した認可コードと、上記①で生成した乱数（鍵セットの一部）を認可サーバに送信する。
⑥認可サーバは、上記⑤で受信した乱数からハッシュ値を生成する。そのうえで、上記③で受信したハッシュ値と比較するとともに、当該ハッシュ値に対応する認可コードと上記⑤で受信した認可コードを比較する。
⑦上記⑥の比較の結果、いずれも一致した場合には、認可サーバはアクセス・トークンをクライアントに送信する。

ここで用いられる鍵セットが安全に保持され、正規クライアントが利用する場合、これを用いたクライアントの確認を行うことができる。一方、鍵セットが攻撃者に奪取される場合、もしくは偽のクライアント（クライアントのなりすまし）の場合には、その他の対策を検討する必要がある。

補論 2.　Financial API に記載されているセキュリティ対策

Financial API は 5 巻から構成されている。1 巻は、Read Only API のセキュリティ対策、2 巻は Read and Write API のセキュリティ対策、3 巻から 5 巻は API 等の実装に必要なソースが記載されている[36]。以下では、1 巻と 2 巻に記載されている主なセキュリティ対策の概要を、本稿で説明した対策と同様のものとそうでないものに分けて示す[37]。

(1)　Read Only API におけるセキュリティ対策

イ．本稿で説明した対策と同等と考えられるもの

（イ）　認可サーバに対する対策

- クライアント・シークレットを用いること。
- 事前にリダイレクト URI を登録し、そのうちの 1 つと完全に一致することを確認すること。
- リソース・オーナーが事前に承認していないリソース・サーバのサービスにクライアントがアクセスする場合、リソース・オーナーの同意を必要とすること。
- 認可コードの利用回数を 1 回のみとすること。認可コードの利用回数が確認できない場合、認可コードの有効期間を適切に設定する（例えば 1 分間）。
- アクセスを長期間許可する場合、リソース・オーナーがそれを把握できるようにすること。アクセス・トークンの有効期間は、リフレッシュ・トークンの寿命よりも短くなければならない。
- アクセス・トークンは、128 ビット以上の暗号学的に安全な乱数または擬似乱数系列から生成すること。
- 公開鍵暗号に RSA 暗号を用いる場合は鍵長を 2,048 ビット以上とするほか、楕円曲線暗号を用いる場合は鍵長を 160 ビット以上とすること。
- ハッシュ関数は SHA-256 を用いること。
- アクセス・トークンごとにアクセスを許可しているサービス項目について、その一覧を照会できる仕組みを提供すること。
- リソース・オーナーがアクセス・トークンを取り消し、クライアントに付与さ

36 Financial API は、英国の Open API Banking Standard が、Financial API を策定している OpenID Foundation の Financial API Working Group と共同作業を行うと発表するなど、注目されている（Open Banking [2017]）。

37 Financial API はそれぞれの対策に対して、使用するパラメータ値を指定しているほか、Sakimura *et al*. [2014] と同等の対策を実施することとしている。これらの詳細については、Sakimura, Bradley, and Jay [2017] を参照されたい。

れたトークンを更新する仕組みを提供すること。
- TLS を用いたクライアント認証を用いること。
- ID トークンを用いた検証を行うこと。

（ロ） リソース・サーバに対する対策
- TLS1.2（あるいは、現在標準化作業中の TLS1.3 が完成すればそれも含む）を利用し、通信経路上を暗号化すること。
- HTTP 通信を行う際には、 HTTP に記載した URI のリソースを取得する設定（GET メソッドと呼ばれる）を用いること。
- 提示されたアクセス・トークンの有効性（有効期限が到来していないか、または取り消されていないか）を確認すること。
- アクセスが要求されたサービスが、提示されたアクセス・トークンの権限で利用できるものか否かを確認すること。
- アクセス・トークンを発行したリソース・オーナーを識別できるようにすること。

（ハ） コンフィデンシャル・クライアントに対する対策
- 公開鍵暗号に RSA 暗号を用いる場合は鍵長を 2,048 ビット以上とするほか、楕円曲線暗号を用いる場合は鍵長を 160 ビット以上とすること。また、共通鍵暗号を用いる場合は、クライアントの秘密鍵が 128 ビット以上であることを検証すること。
- TLS を用いたクライアント認証を用いること。
- TLS1.2（あるいは、現在標準化作業中の TLS 1.3 が完成すればそれも含む）を利用し、通信経路上を暗号化すること。
- HTTP 通信を行う際には、そのヘッダーにアクセス・トークンの情報を含めること。
- 複数の金融機関で同じ認可サーバを使用する場合（例えば、共同センター等の場合）、金融機関は、金融機関を識別する値（識別子）も送信すること。

（ニ） その他の対策
- ログファイルへの厳格なアクセス制御を実施すること。パラメータがログに記録される場合、ログを介して情報が漏洩する可能性がある。

ロ. 本稿で説明した対策とは異なるもの

（イ） 認可サーバに対する対策
- 認証リクエストにリダイレクト URI も含めること。
- クライアント認証は、クライアント・シークレットではなく、TLS 相互認証な

いしは JWS クライアント認証を利用すること。
- エラー時には、事前に取り決めた値を返答すること。

（ロ） リソース・サーバに対する対策
- HTTP のヘッダーには、使用する文字コード（UTF-8）や JSON を用いること等を記載すること。

（ハ） 認証に関する対策
- ID 管理を行う組織に求める運用管理基準（LoA と呼ばれる）は、4 段階中、下から 2 段階目の基準（LoA 2 と呼ばれる）以上で運用すること。
- リソース・オーナーと提示されたアクセス・トークンの権限を検証し、要求されたリクエストがその権限で認められていない場合は、事前に取り決めたエラー値を返答すること。
- リクエストに対する応答には、所定のフォーマットを用いること。
- ログには特定のパラメータを記述すること。

（ニ） パブリック・クライアントに対する対策
- PKCE によって鍵セットを生成する際に利用するハッシュ関数は SHA-256 を用いること。
- リダイレクト URI は、1 クライアント 1 認可サーバとなるように、論理的に分離すること。
- セッションを開始する際（認可コード発行前）に、クライアントが送信したリダイレクト URI と、認可コードを用いてアクセス・トークンを要求する通信に含まれているリダイレクト URI を比較し、それらが異なる場合は要求を却下すること。
- ネイティブアプリを実装する際の要件を記載した文献（Denniss and Bradley [2017]）において必要な対策とされているものを実装すること。
- CSRF 攻撃への対策を実施すること。
- ID トークンを発行し、リプレイ攻撃を緩和するために使用されるパラメータ値等を記載すること。
- 認証する際には特定のパラメータを用いること。

（2） Read and Write API のセキュリティ対策

イ． 本稿で説明した対策と同等と考えられるもの

（イ） 認可サーバに対する対策

- 認可コード、アクセス・トークン、リフレッシュ・トークンを発行する際は、それらを使用するクライアントと紐付け、記名式トークン（Token Binding または Mutual TLS）[38] を利用すること。
- TLS を用いたクライアント認証を行うこと。
- 認可リクエストおよび応答リクエストは JWS を用い、ID トークンを利用するほか、応答レスポンスに、認可サーバの分離署名としての ID トークンを含めること。

（ロ） その他の対策

- 認可リクエストの改ざん（IdP Mix-Up Attack）への対策として、金融機関および中間業者は、ハイブリッド・フロー（Hybrid Flow）を用いる。そして、中間業者は、クライアントとユーザ・エージェントの間のセッションに認可リクエストの送信先の情報を記録しておき、認可コードを発行する際に送信された ID トークンの送信元の情報と一致することを検証する。

ロ． 本稿で説明した対策とは異なるもの

（イ） 認可サーバに対する対策

- クライアント認証には TLS 相互認証か JWS 認証を用いること。
- 記名式トークン（Token Binding または Mutual TLS）を提供すること。
- ID 管理を行う組織に求める運用管理基準（LoA と呼ばれる）は、4 段階中、下から 3 段階目の基準（LoA 3 と呼ばれる）以上で運用すること。
- ID トークンは、署名を付与することに加えて暗号化することもサポートすること。

（ロ） コンフィデンシャル・クライアントに対する対策

- 記名式トークン（Token Binding または Mutual TLS）を提供すること。
- 署名付きのトークンと暗号化トークンの両方を用いること。

38 本項目以降の Token Binding は、TLS 通信路に由来する秘密情報から導出される暗号学的乱数（Exported Key Material）をトークンに紐付けることを指し、Mutual TLS は、TLS 通信路に由来するクライアント証明書をトークンに紐付けることを指す。

（ハ） パブリック・クライアントに対する対策

- 認証のリクエストは OpenID Connect で用いられるリクエスト・パラメータ等を用いること。
- ID 管理を行う組織に求める運用管理基準（LoA と呼ばれる）は、4 段階中、下から 3 段階目の基準（LoA 3 と呼ばれる）以上で運用すること、もしくは、ID トークンにパラメータの内容（LoA 3 で運用したこと）を記述し、そのパラメータの内容を検証すること。
- 応答レスポンスに、認可サーバの分離署名としての ID トークンを含めること。

（ニ） リソース・サーバに対する対策

- 記名式トークン（Token Binding または Mutual TLS）を用いること。