

A2B攻撃に対する アイデンティティ技術の有効性と 組織への提言

2026年10月30日

一般社団法人OpenIDファウンデーション・ジャパン

代表理事 富士 榮 尚寛



エグゼクティブサマリー

A2B攻撃 (App-to-Backend Attack) の本質は「アプリの中身は攻撃者に読まれる」という前提を欠いたAPI設計にあり、OpenID Connect／OAuth関連技術はその前提に立った“認証・認可の作り直し”を可能にします。

2026年10月、NICT CYNEXの脅威情報分析チームLETTICEは、公開Androidアプリを解析してバックエンドAPIを狙う攻撃が国内の情報漏えいの一因になっていると注意喚起しました。狙われたのは、アプリ自体ではなく、API側の認証・認可や入力値検証の不備です。

本書の要点は次の4つです。

- アプリにシークレットを置かない。アプリに埋め込んだAPIキーやクライアントシークレットは、解析で必ず取り出されます。公開クライアント+PKCE、アプリ構成証明 (Attestation) を組み合わせ、静的なシークレットに依存しない構成に移行すべきです。
- 「誰のデータか」はトークンで決める。IDOR¹／BOLA²の根本対策は、リクエスト中のIDではなく、検証済みのアクセストークンの主体 (iss + sub) とスコープに基づいてサーバー側で認可判定を行うことです。
- 盗まれても使えないトークンにする。DPoPやmTLSによる送信者制約トークン、短い有効期限、細粒度の権限付与 (RAR) により、漏えい時の被害範囲を限定します。
- **OAuth**は万能ではない。SQLインジェクションやSSRF³などの実装脆弱性は、認可基盤とは別にセキュア開発と診断で潰す必要があります。

企業・組織には、まず短期的に侵害の確認、APIの把握、「他人のデータが見えないか」の点検に取り組み、中長期的にはFAPI 2.0等の標準に沿った認可基盤と、アイデンティティを経営のテーマとして扱う体制を目指すことをご提案します。

¹ Insecure Direct Object Reference: WebサイトやAPIでURLやパラメータのID/ファイル名を書き換えることで本来アクセス権のない他人のデータを参照できてしまう脆弱性

² Broken Object Level Authorization: WebサイトやAPIにおいて、オブジェクトレベルの認可機能が適切に実装されていない状態

³ Server-Side Request Forgery: Webアプリケーションの機能を利用して、攻撃者が意図した宛先へサーバーから不正なリクエストを送信させてしまう脆弱性

1. はじめに

本書は、A2B攻撃による情報漏えいの連鎖を受け、アイデンティティ技術の観点から「何が有効で、何が有効でないか」を整理し、企業等の組織が取るべき行動を示すものです。

2026年10月7日から9日にかけて、個人情報保護委員会、JPCERT/CC、国家サイバー統括室、総務省などが相次いで関連する注意喚起を行いました。10月13日にはNICT CYNEX(LETTICE)が攻撃手法と対策をまとめています。各注意喚起は「侵害を確認し、脆弱性を塗りつぶす」ためのものです。本書はその先、つまり「同じ種類の攻撃に構造的に強いAPIをどう作るか」に焦点を当てています。

想定読者は、モバイルアプリやWeb APIを提供する事業者の経営層・CISO、サービス企画者、アーキテクト、開発責任者です。第2・3章と第6章は経営層向け、第4・5章は技術者向けの内容となっています。

2. A2B攻撃とは何か

A2B攻撃(App-to-Backend Attack)は、誰でも入手できるスマートフォンアプリを手がかりに、その裏側にある企業のサーバーを攻撃する手口です。2026年10月13日にNICTが名前を付けて公表したばかりの用語のため、まず仕組みを平易に説明します。

2.1 概要

多くのスマートフォンアプリは、画面の裏側で企業や組織のサーバー(バックエンド)と通信し、会員情報や注文履歴を取得しています。その窓口となるのがAPIです。スマートフォンにダウンロードされたアプリの中身を解析されることを防ぐのは非常に困難です。

攻撃者はアプリを解析して、「どのサーバーの、どのAPIに、どんな方法で通信すればよいか」を調べ上げます。そのうえでアプリを使わずにAPIへ直接アクセスし、例えば会員番号を一つずつずらしながら、本来は見られない他人の情報を引き出します。

このとき、利用者が何かを間違えたわけでも、アプリが壊れているわけでもありません。問題は、サーバーが「自社アプリから来た要求だから大丈夫」と信じて、相手が誰で、そのデータを見てよいかを確かめていない点にあります。

2.2 攻撃の流れ

NICT CYNEXによれば、今回確認された攻撃は次の流れで進行します。

1. 入手と解析: Google Play(日本地域)で公開されているアプリを入手し、プログラムを解析する。
2. 情報収集: 通信先のホスト、API構成、パラメータ、応答データの構造、そして埋め込まれた認証情報や暗号鍵を集める。
3. 個別最適化: 不特定多数への一律攻撃ではなく、対象ごとのAPI仕様に合わせて攻撃を組み立てる。

4. 悪用: 認証不要のAPIだけでなく、ログイン後のAPIも狙う。IDOR／BOLA、認証・セッション管理の不備、SQL／NoSQLインジェクション、SSRF、RCE⁴、クラウドストレージの設定不備が主な標的となる。

NICTは、AIによる解析や脆弱性探索の効率化で、この攻撃が多くのシステムに広がる懸念を示しています。「解析に手間がかかるから安全」という前提は崩れたと考えるべきです。

NICTはまとめとして、アプリの解析を完全に防ぐことは困難であり、内部構造やAPI仕様が知られても安全が保たれる設計・運用が重要だとしています。また、iOSアプリやWebサービスのAPIにも同様のリスクがあると指摘しています。特定のOSやストアの問題ではなく、「クライアントを信頼したAPI」全般の問題です。

2.3 2026年秋の相次ぐ漏えいとA2B攻撃の関係

A2B攻撃は、2026年秋に相次いだ個人情報漏えいの「唯一の原因」ではありませんが、複数の公的機関がその重要な一因として同じ構図を指摘しています。

2026年9月から10月にかけて、国内の多くの事業者が不正アクセスによる個人情報の漏えい（またはそのおそれ）を公表しました。これを受けて、政府と関係機関は約1週間のうちに一斉に注意喚起を行いました。主なものは次のとおりです。（出典: 参考文献[1]～[3]ほか各機関の公表）

日付	機関	内容
2026年 10月13日	NICT CYNEX(LETTICE)	公開アプリの解析結果を悪用する手口を「A2B攻撃」と名付け、相次ぐ漏えいの一因として公表
2026年 10月9日	国家サイバー統括室、総務省、経済産業省、IPA、金融庁	大量の個人情報を扱う事業者等に、外部公開システムの点検や本人確認の徹底を要請
2026年 10月8日	JPCERT/CC	手法を4つの類型に整理。うち「ケースB」は、アプリを解析してAPIの接続先やキーを特定し、内部APIを不正操作する手口
2026年 10月7日	個人情報保護委員会	注意喚起(WARNING)を改訂し、APIの設計不備によりパラメータの書き換えだけで他人の情報を取得された事例を追加

⁴ Remote Code Execution: ネットワークを経由した外部から遠隔で任意のプログラムコードやコマンドを不正に実行される脆弱性

これらの資料を並べると、次の共通点が見えてきます。

- 個人情報保護委員会の事例では、管理から漏れたAPIが存在し、会員IDが連番で、レート制限もなかったため、IDを書き換えるだけで大量の他人の情報を取得できました。
- JPCERT/CCのケースBでは、アプリ解析で得たキーを使い、通常の画面からは利用できない内部APIで権限の変更や不正なアカウントの作成が行われました。
- NICTのA2B攻撃は、こうした手口を「アプリ解析を起点に、対象ごとに最適化して繰り返す攻撃」として体系的に整理したものです。

つまり、個別の漏えいは別々の企業で起きたように見えても、その一部は「アプリを信頼したサーバー」という同じ弱点を、同じ手順で突かれたものである可能性があります。AIによる解析の自動化が、この手順を多数の企業へ短期間で広げるおそれもあります。

なお、JPCERT/CCはこのほかにも、既知の脆弱性の悪用やWebシェルの設置など別の手口を挙げています。個々の事案がA2B攻撃によるものかどうかは公表されておらず、本書も個別事案の原因を特定する立場は取りません。

3. 根本原因: クライアントを信頼した設計

A2B攻撃が成立する理由は、OWASP Mobile Application Security Cheat Sheetが冒頭で掲げる原則「Don't trust the client」に反した設計にあります。同シートは、認証・認可をサーバー側で行うこと、クライアント側の制御は回避される前提でサーバー側でも同じ制御を行うこと、認証情報をハードコードしないこと、偽装可能なデバイス識別子を認証に使わないことを求めています。

A2B攻撃で悪用される設計上の罫は、次の5つに整理できます。括弧内は、該当する主なCWE（共通脆弱性タイプ一覧）です。

- 静的なシークレットへの依存(**CWE-798**): APIキー、クライアントシークレット、署名鍵、クラウドのアクセスキーをアプリに埋め込み、「正規アプリからの通信であること」の証拠としている。配布物に含まれる値はシークレットではありません。
- クライアント申告の識別子の信用と関係性検証の欠如(**CWE-284**、**CWE-639**、**CWE-340**): `/users/{id}` のようにリクエストに含まれるIDをそのまま使用し、「その利用者とそのデータの間に、アクセスしてよい関係があるか」を検証していない。会員IDが連番など推測しやすい値であれば、攻撃者は他人のIDを順に試すだけで済みます。
- 無制限・長寿命のトークン: 独自形式のセッショントークンやベアラートークンが、どのAPIにも、誰が提示しても、長期間使える。一度入手されると被害が拡大します。
- 認可ロジックの分散(**CWE-284**): 各APIが個別に認証・認可を実装しており、旧APIや検証環境ではチェックが抜けている。NICTが公開ホストの洗い出しを最初に求めているのはこのためです。
- 必要以上のデータを返すAPI(データ最小化の欠如、**CWE-200**): 画面に表示しない項目までAPIが返し、表示の絞り込みをアプリ側に任せている。アプリを介さずAPIを直接呼べば、画面では隠していた情報もすべて読み取れます。OWASP API Security Top 10 2023の「API3:2023 Broken Object Property Level Authorization」に当たり、個人情報保護委員会の事例でも応答に不要な情報が含まれていたことが指摘されています。

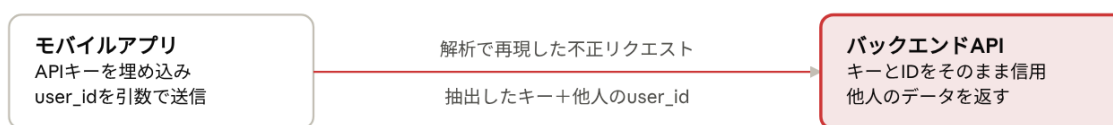
これらはいずれも「誰が、どのアプリで、何に対して、何をしたいか」をサーバーが自ら検証できないという、アイデンティティとアクセス管理の問題です。OpenID／OAuth関連技術は、まさにこの問いに標準的な答えを与えるために作られてきました。

4. OpenID／OAuth関連技術はどこで有効か

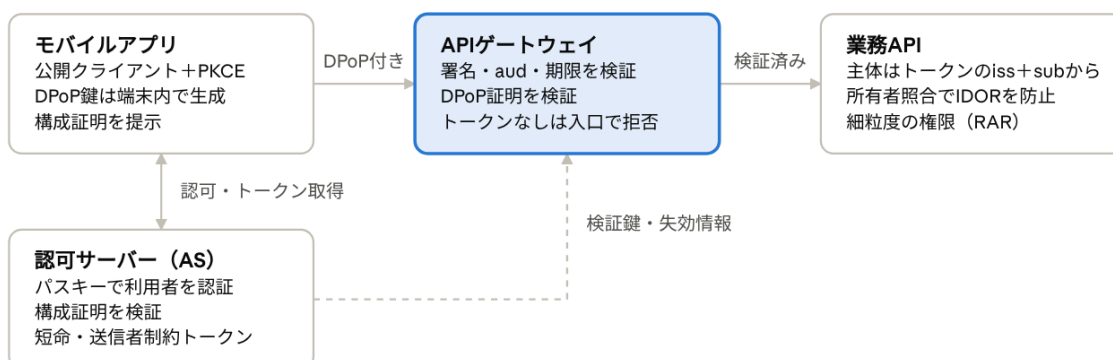
OpenID／OAuth関連技術がA2B攻撃に有効なのは、「アプリを信頼する」設計を「検証可能なトークンを信頼する」設計に置き換える点にあるからです。以下、防御の目的ごとに整理します。

信頼の根拠を「アプリ」から「検証できるトークン」へ移す

A2B攻撃で狙われた設計：アプリを信頼していた



OpenID／OAuth：トークンを検証する設計



A2B攻撃で狙われたのは、すべてのアプリではなく、アプリを信頼する設計になっていたAPIです。そうした設計では、解析で抽出されたキーと書き換えたIDだけでAPIが応答します。OpenID／OAuth型では、認可サーバーが発行し端末の鍵に紐付いたトークンを入口で検証し、業務APIはトークンの主体でデータへのアクセス可否を決めます。

なお、この図は自社アプリと自社の認可サーバーの組み合わせ（ファーストパーティ）を想定しています。この場合、認可サーバーがパスキーで利用者を直接認証すればよく、OpenID Connect（OIDC）は必須ではありません。外部のIDプロバイダーと連携する場合や、複数のサービスでログインを共有する場合には、OIDCを組み合わせます。

4.1 アプリにシークレットを置かない

- **公開クライアント+PKCE**: モバイルアプリはシークレットを保持できない「公開クライアント」として扱われます。RFC 8252 (OAuth 2.0 for Native Apps) に従い、システムブラウザ

経由の認可コードフローとPKCE(RFC 7636)を使えば、アプリにクライアントシークレットを埋め込む必要がなくなります。解析されても、取り出せる「合鍵」はありません。

- 非推奨フローの廃止: OAuth 2.0 Security BCP(RFC 9700)は、Implicitフローを非推奨(SHOULD NOT)とし、Resource Owner Password Credentialsフローの使用を禁止(MUST NOT)としています。アプリがパスワードを直接ログインAPIに送信する独自実装は、攻撃者にとってリスト型攻撃の入口になります。
- アプリ構成証明(Attestation)との組み合わせ: Play Integrity APIやApp Attestの結果をサーバー側で検証し、正規アプリ以外からの自動化された呼び出しのコストを引き上げます。IETF OAuth WGでは、これをOAuthのクライアント認証に組み込む「Attestation-Based Client Authentication」の標準化が進んでいます(2026年10月時点でWGラストコール中)。ただしこれはセキュリティ強化策であり、認可の代わりにはなりません。

4.2 「誰のデータか」をトークンで決める

- IDトークンとアクセストークンを使い分ける: OpenID ConnectのIDトークンは、OpenID Providerが「この利用者が、いつ、どのように認証されたか」をクライアント(アプリ)に伝えるためのもので、宛先(aud)はクライアント自身です。APIを呼び出すためのものではなく、APIがIDトークンを受け付けてはいけません。APIへのアクセスには、OAuthの認可サーバーがそのAPI宛てに発行したアクセストークンを使います。
- APIはアクセストークンから主体を取得する: APIに提示されるのはOAuthのアクセストークンです。OAuth 2.0自体はアクセストークンの形式を定めておらず、subやaudを持たないランダムな文字列形式の場合もあります。またClient Credentialsグラントのように利用者が関わらないトークンには、そもそも利用者としての主体がありません。そのため、個人データを扱うAPIでは、認可サーバーとリソースサーバーの間で「主体と宛先を確実に取得できる」形を設計として決めておく必要があります。具体的には、sub・client_id・audを必須とするRFC 9068のJWT形式を採用するか、ランダム文字列型トークンの場合はRFC 7662のイントロスペクション応答にsubやaudを含めるよう運用します(同仕様ではこれらは任意項目です)。APIは、トークンが自分宛てに発行されたものであることを確認したうえで、「このトークンの主体が、要求されたオブジェクトの所有者か」を必ず照合します。リクエスト中の利用者IDを信用しなければ、IDを連番で変える攻撃は成立しません。
- subが示すのは「いま操作している人」ではない: アクセストークンやイントロスペクション応答のsubが示すのは、「このトークンがどの利用者の権限で発行されたか」です。いまAPIを呼び出しているのがその利用者本人であることまでは保証しません。たとえば盗まれたベアラートークンを攻撃者が提示すれば、APIには被害者のsubが届きます。また、subは発行元(iss)の中でしか一意ではなく、クライアントごとに異なる値(OpenID Connectのペアワイズ識別子など)が使われることもあります。そのため、次の3点を組み合わせます。①DPoPやmTLSでトークンを正規クライアントの鍵に紐付け、提示者が発行先と同じであることを確かめる(4.3参照)。②利用者の照合はissとsubの組で行い、社内の会員IDとの対応付けはサーバー側で管理する。③決済や登録情報の変更など重要な操作では、トークンの認証時刻(auth_time)や認証レベル(acr)を確認し、不十分であれば再認証を求める(RFC 9470 OAuth 2.0 Step Up Authentication Challenge Protocol)。
- 宛先と権限の限定: Resource Indicators(RFC 8707)でトークンの宛先APIを限定し、Rich Authorization Requests(RFC 9396)で「この口座の、この操作だけ」といった細粒度の権限を表現します。管理機能のスコープは一般利用者向けアプリには決して発行しません。

- 認可判定の外部化:オブジェクト単位の判定を各APIに散らばせず、ポリシー判定点(PDP)に集約すると、古いAPIや検証環境だけチェックが抜ける事態を防げます。OpenID FoundationのAuthZEN WGは、PEPとPDPの間の標準APIを「Authorization API 1.0」として策定しています(2026年1月に最終仕様として承認)。

4.3 盗まれても使えないトークンにする

- 送信者制約トークン:DPoP(RFC 9449)やOAuth mTLS(RFC 8705)は、トークンを端末内の鍵に紐付けます。ログや通信からトークンだけが漏れても、鍵を持たない攻撃者は再利用できません。DPoPの鍵は、Android Keystore(StrongBox/TEE)やiOS Secure Enclaveで生成すると、効果が高まります。
- 短い寿命とローテーション:アクセストークンは数分~数十分の有効期限とし、リフレッシュトークンには送信者制約またはローテーション(RFC 9700)を適用します。漏えいが疑われる場合は、RFC 7009の失効により、直ちに無効化します。

4.4 「正しく使う」ためのプロファイル

- **FAPI 2.0 Security Profile**: OAuthは選択肢が多く、組み合わせを誤るとそれ自体が脆弱性になります。OpenID Foundationが策定したFAPI 2.0は、PAR(RFC 9126)、PKCE、送信者制約トークンなどを必須とし、攻撃者モデルに基づく形式分析を経た「安全な組み合わせ」を定めています。OpenIDファウンデーション・ジャパンは、金融に限らず、個人情報扱うAPIの基準線としてFAPI 2.0の採用を強く推奨します。
- 適合性テストと認定: OpenID Foundationは、OpenID ConnectやFAPIの適合性テストスイートおよび認定プログラムを提供しています。自社実装や調達製品が仕様どおりに動作するかを客観的に確認できます。

4.5 利用者認証と継続的なリスク対応

- パスキーによる利用者認証: APIを直接呼び出せる攻撃者は、ログインAPIに対するリスト型攻撃も容易に行うことができます。認可サーバー(OpenID Connectを使う場合はOpenID Provider)側でパスキー(FIDO)を採用すれば、パスワードの使い回しを前提とした攻撃を防ぐことができます。
- **Shared Signals Framework(CAEP/RISC)**: OpenID FoundationのShared Signals仕様は、セッション失効やアカウント侵害などのイベントをサービス間で即時に共有します。異常を検知した瞬間に、関連する全サービスでトークンやセッションを停止できます。

4.6 限界: アイデンティティ技術では防げないもの

SQL/NoSQLインジェクション、SSRF、RCE、危険なデシリアライズ、クラウドストレージの公開設定は、認可基盤を整えてもそれだけでは解消しません。ただし、「認証不要のAPI」を極小化し、正当なトークンを持たないリクエストを入口で拒否すれば、これらの脆弱性に到達できる攻撃者を大きく絞れます。また、OAuthの実装自体の不備(audや署名の未検証など)も攻撃対象になるため、トークン検証はAPIゲートウェイ等で一元化し、全ホストに漏れなく適用することが重要です。

5. 攻撃手法と対策の対応表

NICTや個人情報保護委員会が示した点検項目のうち、アイデンティティ技術が根本対策になるのは上の4行です。下の4行は「到達できる攻撃者を絞る」「漏れる範囲を減らす」という補助的な効果にとどまります。

攻撃手法・不備	有効な技術	効果	残る課題
埋め込まれたAPIキー・シークレットの抽出	公開クライアント+PKCE(RFC 7636, 8252)、Attestation-Based Client Authentication	根本対策:抽出できるシークレットがなくなる	既存アプリのシークレットの失効と再発行が必要
IDOR/BOLA(他人のIDを指定。CWE-284、CWE-639)	OAuthアクセストークンの主体(iss + sub)の照合、送信者制約トークン(RFC 9449、8705)、RAR(RFC 9396)、AuthZENによるPDP集約	根本対策:主体をサーバーが検証可能な形で得られる	利用者とデータの関係の照合は各APIの実装責任。推測困難な識別子(CWE-340対策)も併用
トークン偽造・認証回避・セッション不備	標準トークン検証(RFC 9068, 7662)、FAPI 2.0、適合性テスト	根本対策:独自方式を検証済みの方式に置き換える	全ホストでの検証の徹底
入手したトークンの再利用	DPoP(RFC 9449)、mTLS(RFC 8705)、短寿命トークン、Shared Signals	根本対策:鍵なしでは使えない	端末自体が乗っ取られた場合は別対策
ログインAPIへのリスト型攻撃	認可サーバー(またはOpenID Provider)でのパスキー認証、ROPC廃止(RFC 9700)	補助～根本:パスワード依存を減らす	移行期間中のパスワード併用
SQL/NoSQLインジェクション、SSRF、RCE	認証必須化と入口でのトークン拒否	補助:未認証の攻撃者を遠ざける	入力値検証、セキュア開発、診断が必須

クラウドストレージの設定不備	Token Exchange(RFC 8693)による短命・限定的なクレデンシャル	補助: 静的アクセスキーをなくす	公開設定の棚卸しと修正
過剰なデータ返却(CWE-200)	スコープとRAR(RFC 9396)による返却範囲の限定	補助: 返す項目を権限と結びつけられる	応答項目の最小化はAPI設計の責任

6. 企業・組織への提言

本書では、取り組みを「緊急度が高く、短期的に取り組むこと」と「中長期的に目指す姿」の2段階で提言します。すべてを一度に行う必要はなく、サービスの規模や扱うデータの重要度に応じて、優先順位を付けて進めていただければと考えています。

6.1 緊急度が高く、短期的に取り組むこと

まずは、すでに被害が出ていないかを確認し、攻撃の侵入経路を塞ぐことが大切です。多くは既存の仕組みを大きく変えずに着手できます。

1. 公開している**API**を把握する: アプリが通信するすべてのサーバーとAPIを、旧バージョンや検証環境も含めて洗い出しましょう。把握できていないAPIは守れません。
2. 侵害の有無を確認する: NICTの指針に沿って、2026年8月以降のアクセスログや認証ログを確認し、連番IDへのアクセスや大量取得の痕跡がないか調べましょう。
3. アプリに含まれるシークレットを無効にする: 自社アプリを攻撃者の視点から解析し、埋め込まれたAPIキーや鍵があれば、失効と再発行を検討してください。
4. 「他人のデータが見えないか」を点検する: 個人情報を出すAPIで、ログイン中の利用者と要求されたデータの持ち主が一致しているかを確認しましょう。A2B攻撃に対する最も効果的な対策の一つです。あわせて、応答に画面で使わない項目が含まれていないか、会員IDが連番など推測しやすい値になっていないかも確認しましょう。
5. 当面の防御策として、レート制限やWAFで大量アクセスを抑え、不要な公開APIや管理機能は停止することを検討してください。
6. 利用者に認証の強化を呼びかける: 多要素認証やパスキーが利用できる場合は、その利用を案内しましょう。

6.2 中長期的に目指す姿

次の段階では、「アプリの中身が知られても安全」な状態を、標準技術に基づいて組織全体で実現することを目指します。場合によって既存システムへの大きな変更を伴うこともあるため、システム更新の機会に合わせて段階的に進めることを推奨します。

- 標準に基づく認証・認可への移行:独自のログインやセッションの仕組みを、OAuth(認可コードフロー+PKCE)に置き換え、アプリにシークレットを置かない構成にします。外部のIDプロバイダーとの連携や、複数サービス間でのログイン共有が必要な場合は、OpenID Connectを組み合わせます。
- トークン検証の一元化:APIゲートウェイなどでトークンをまとめて検証し、正当なトークンがない要求は入口で拒否する形にします。
- **API**の応答を必要最小限にする:画面で使う項目だけを返すようAPIを設計し、表示の絞り込みをアプリ側に任せないようにします。外部に出る識別子には、連番ではなく推測困難なランダム値を使用します。
- **FAPI 2.0**の採用:4.4で述べたとおり、個人情報や決済を扱うAPIにはFAPI 2.0 Security Profileの採用を強く推奨します。DPoPなどの送信者制約トークンにより、トークンが漏えいしても悪用されにくくなります。
- アプリと利用者の確からしさを高める:Play Integrity APIやApp Attestによるアプリ構成証明、パスキーによる利用者認証を組み合わせます。
- 認可ルールの集約と連携:「誰が何をしてよいか」のルールを一か所で管理し、Shared Signals(CAEP/RISC)で異常検知時にサービスを横断してセッションを止められるようにします。
- 適合性テストと調達要件の活用:OpenID Foundationの適合性テストで実装を確認し、開発委託や製品調達の際には認証・認可の要件として盛り込みます。
- アイデンティティを経営のテーマにする:認証・認可を各開発チームの実装の細部としてではなく、共通基盤として投資し、経営層とCISO、事業部門が一緒に責任を持つ体制を整えます。

個別実装のばらつきは、A2B攻撃が狙う「一か所だけの抜け」を生み出しやすくします。共通の仕組みを少しずつ整えることが、結果として最も確実な備えになります。

7. おわりに

A2B攻撃は、「アプリは解析される」という当然の前提を、AIによる効率化が現実の脅威に変えたことを示しています。対策の中心は、アプリを隠すことではなく、サーバーが「誰が、何に、何を、何を」を自ら検証できる状態を作ることです。

OpenIDファウンデーション・ジャパンは、OpenID Connect、FAPI、Shared Signals、AuthZENなどの仕様の日本語での普及、実装者向けの情報共有、適合性テストと認定の活用促進を通じて、国内の事業者が標準に基づく安全な認可基盤へ移行できるよう支援していきます。会員企業、関係機関、開発者コミュニティとの連携を広く呼びかけます。

参考文献

■ 公的機関による注意喚起

[1] [個人情報保護委員会「不正アクセスによる個人データ漏えい防止のための注意喚起\(WARNING\)」\(令和6年12月11日、令和8年10月7日改訂\)](#)

[2] [JPCERT/CC「直近で相次いでいる国内組織における不正アクセスに関する注意喚起」
\(JPCERT-AT-2026-0030、2026年10月8日公開、10月23日更新\)](#)

[3] [NICT CYNEX 脅威情報分析チーム LETTICE「【注意喚起】相次ぐ情報漏洩、その原因の一つは「アプリ解析からのサーバ攻撃」— AIで高度化する脆弱性探索、Androidアプリを手がかりにしたA2B攻撃に注意 —」\(2026年10月13日、Rev 1.1\)](#)

■ セキュリティ基準・脆弱性分類

[4] [OWASP Cheat Sheet Series「Mobile Application Security Cheat Sheet」\(1. Don't Trust the Client\)](#)

[5] [OWASP API Security Top 10 2023「API3:2023 Broken Object Property Level Authorization」](#)

[6] [MITRE「CWE-200: Exposure of Sensitive Information to an Unauthorized Actor」](#)

[7] [MITRE「CWE-284: Improper Access Control」](#)

[8] [MITRE「CWE-340: Generation of Predictable Numbers or Identifiers」](#)

[9] [MITRE「CWE-639: Authorization Bypass Through User-Controlled Key」](#)

[10] [MITRE「CWE-798: Use of Hard-coded Credentials」](#)

■ IETF 仕様

[11] [RFC 7009「OAuth 2.0 Token Revocation」](#)

[12] [RFC 7636「Proof Key for Code Exchange by OAuth Public Clients」\(PKCE\)](#)

[13] [RFC 7662「OAuth 2.0 Token Introspection」](#)

[14] [RFC 8252「OAuth 2.0 for Native Apps」](#)

[15] [RFC 8693「OAuth 2.0 Token Exchange」](#)

[16] [RFC 8705「OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens」](#)

[17] [RFC 8707「Resource Indicators for OAuth 2.0」](#)

[18] [RFC 9068「JSON Web Token \(JWT\) Profile for OAuth 2.0 Access Tokens」](#)

[19] [RFC 9126「OAuth 2.0 Pushed Authorization Requests」\(PAR\)](#)

[20] [RFC 9396「OAuth 2.0 Rich Authorization Requests」\(RAR\)](#)

[21] [RFC 9449「OAuth 2.0 Demonstrating Proof of Possession \(DPoP\)」](#)

[22] [RFC 9470「OAuth 2.0 Step Up Authentication Challenge Protocol」](#)

[23] [RFC 9700「Best Current Practice for OAuth 2.0 Security」](#)

[24] [IETF Internet-Draft「OAuth 2.0 Attestation-Based Client Authentication」
\(draft-ietf-oauth-attestation-based-client-auth、2026年10月時点で第12版\)](#)

■ OpenID Foundation 仕様・活動

[25] [OpenID Foundation「OpenID Connect Core 1.0 incorporating errata set 2」](#)

[26] [OpenID Foundation「FAPI 2.0 Security Profile」\(Final、2025年2月22日\)](#)

[27] [OpenID Foundation「Shared Signals Working Group」](#)

[28] [OpenID Foundation「AuthZEN Working Group」](#)

[29] [OpenID Foundation「OpenID Certification」](#)