Akamai

ホワイトペーパー

API セキュリティの基礎：知識を身に付け、エンタープライズのセキュリティを確保する

はじめに

API は、「実装上の詳細部分」から「デジタルイノベーションの戦略的な実現手段」へと急速に進化してきました。顧客、パートナー、ベンダーとの間でデジタル処理が行われるたびに、各処理の背後でシームレスなデータのやり取りを行うための API が機能しています。

そして API の普及に伴い、そのリスクも増大しています。新しいアプリケーションと AI 強化型サービスの迅速な開発とリリースを競う中で、基盤となる API の設定が誤っていたり、セキュリティ制御が不十分であったり、攻撃が容易に実行される脆弱性が存在したりすることがあまりにも多くなりました。

その結果、API が攻撃ベクトルの上位に浮上し、多くのセキュリティチームが API セキュリティ戦略の対応に追われています。そのため、API セキュリティは、IT およびセキュリティ管理者にとって最優先の戦略的課題として急速に浮上しています。

API セキュリティの基礎を固めたい方、適切な質問のリストを作成したい方、どちらにも役立つこのガイドでは、次のような詳細情報を提供しています。

- **API のタイプ**
- **今日の企業にとって API セキュリティが意味するもの**
- **API セキュリティのリスクに対処するためのベストプラクティス**
- **一般的な API 攻撃と悪用手段**

API セキュリティのベストプラクティスに直接進むには、10 ページに進んでください。

目次

API の基本

Web API とは

Web アプリケーション・プログラミング・インターフェース（Web API）とは、あらかじめ決まっている要求-応答型のメッセージシステムと接続する 1 つ以上のエンドポイントで構成される、プログラム上のインターフェースのことです。この接続先のシステムは、通常、JSON 形式または XML 形式であり、公開され、Web 経由でアクセスできます（たいていは HTTP ベースの Web サーバーという形で公開）。

つまり、「API」という言葉を聞いたときに、ほとんどの人は Web API を思い浮かべることになりますが、実際には API はエンドポイントの集まりです。エンドポイントを構成する要素は、リソースパス、そのリソースに対して実行できる処理、リソースデータの定義（JSON、XML、Protobuf などの形式）です。

Web API は、オペレーティングシステムや同じマシン上で実行されているアプリケーションのライブラリによって公開される API など、他の API とは異なりますが、一般的な「API」とは通常、特にエンタープライズのデジタルトランスフォーメーションと API セキュリティの文脈においては、HTTP ベース（Web）の API を指します。

一般に普及している API のタイプ

次の表に、API 実装におけるさまざまな使用モデルと技術的アプローチを表す用語を示します。Web API は、HTTP をベースとした API であり、そのうち今日よく使用されている主な 4 つのタイプは、RESTful、SOAP、GraphQL、gRPC です。これらの一般的なタイプとその他のタイプの API の定義を以下の表に記載しています。

API 使用モデル	説明
パブリック API	すべての開発者がインターネットを介して自由に利用、共有できる API。
外部 API	インターネットに公開されている API。パブリック API と同じ意味で使用されることもあります。
プライベート API	信頼できる開発者が使用できるように、保護されたデータセンターやクラウド環境に実装される API。
内部 API	プライベート API と同じ意味で使われることもあります。
サードパーティ API	アプリケーションで使用することを目的として、サードパーティのソースから特殊な機能やデータにプログラムでアクセスできるようにする API。
パートナー API	サードパーティ API の一種。認可されたビジネスパートナーのみがアクセスできます。
認証型 API	アクセス権限を付与された開発者（または不正にアクセスする権限を得た攻撃者）のみがアクセスできる API。
非認証型 API	特定の認証情報を必要とせずにプログラムからアクセスできる API。
HTTP API	API コールの通信プロトコルとしてハイパーテキスト・トランスファー・プロトコル（HTTP）を使用する API。

RESTful API	Representational State Transfer（RESTful）は、プレーンテキスト、HTML、XML、YAML、または JSON を使用してデータを配信する最も一般的な Web API です。RESTful API は最新のフロントエンドフレームワーク（React や React Native など）で簡単に利用でき、Web およびモバイルアプリケーションの開発を容易にします。RESTful API は、B2B で使用されるものも含め、あらゆる Web API の事実上の標準となっています。
GraphQL	GraphQL API は、Facebook が開発した新しい標準であり、単一の POST エンドポイント（通常は /graphql）を介してデータベースにアクセスできるものです。RESTful API の一般的な問題、つまり単一のユーザー・インターフェース・ページにデータを入力するために複数の呼び出しが必要となる問題を解決します。
SOAP	SOAP は、リモート・プロシジャー・コール（RPC）に冗長性のある eXtensible Markup Language（XML）を使用します。古いタイプの API でまだ使用されています。
XML-RPC	XML-RPC は、インターネット経由でプロシジャーコールを実行する手段で、エンコーディングに XML を使用し、通信プロトコルに HTTP を使用します。
gRPC	gRPC API は、HTTP/2.0 を介してアクセスする、Google 開発の高性能バイナリープロトコルで、East-West（内部ネットワーク内）通信で最もよく使用されています。
OpenAPI	OpenAPI は、API の記述と文書化を定めた仕様です。Swagger という用語はオリジナルの仕様を指し、OpenAPI は OpenAPI Initiative が開発したオープンスタンダードを指していることを知っておくと便利です。

API とエンドポイントの違い

人はよく「API」という言葉を使いますが、実際には 1 つの API エンドポイントを指しています。本来、API（サービスまたは API 製品と呼ばれることもあります）とは、業務上の機能を担うエンドポイントの集まりです。一方、個々のエンドポイントは、リソース（リソースパス、URI または統一リソース識別子とも呼ばれる）と、リソースに対して実行される操作（作成、読み取り、更新、削除）です。RESTful API では通常、操作は HTTP メソッド（POST、GET、PUT、DELETE）にマッピングされます。

North-South API とは

この API は、外部からアクセスできるように組織が用意する API のことで、主にビジネスパートナーとビジネスを行う目的で使用されます。API 公開とも呼ばれます。以下に例を示します。

> **オープンバンキングを採用している銀行が API を介して他のフィンテック組織や金融サービス組織にデータへのアクセスを提供する。**

> **ヘルスケア関連の組織が API を介して保険会社や医療機関に患者の記録へのアクセスを提供する。**

> **ホスピタリティ組織が API を介して旅行代理店やアグリゲーターに予約システムへのアクセスを提供する。**

API は、異なる組織がデータを交換できるようにする結合組織です。North-South API は、アクセスが許可され認証されているため、ともすれば安全であるとみなされがちです。しかし、一般的に見て、この API は急激に成長し、量も非常に多くなっているため、ほとんどの組織にとって最大のアタックサーフェスになります。

East-West API とは

この API は、組織内部で使用する API であり、社外からのアクセスを一切受け付けません。社内のアプリケーション間の連携や、事業部門や部署との連携に使用します。開発者が誤って、偶発的に East-West API をアクセス可能にしてしまう可能性があります。これらの API は、外部のエンティティからアクセスされたり、知られたりすることを想定したものではありませんが、攻撃者がインターネット経由でアクセス可能な East-West API を見つけると、侵害が発生します。

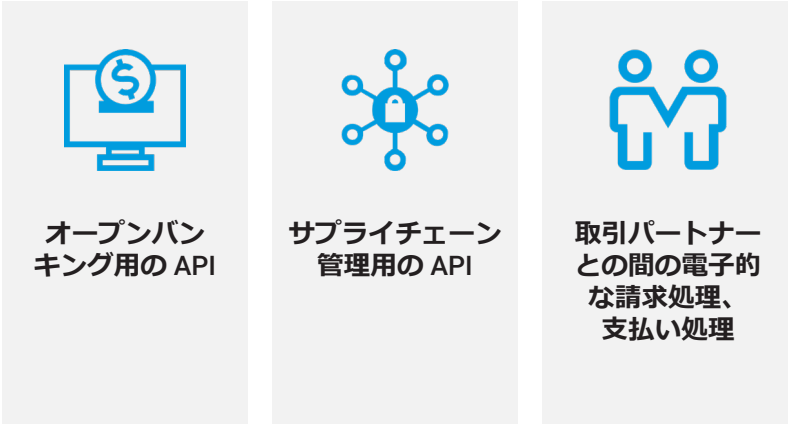
B2C API と B2B API の違い

企業対消費者（B2C）API は、Web アプリケーションやモバイルアプリケーションで活用されています。この API は一般的に、先進的なフロントエンドのクライアントが使用します。その目的は、認証済みのエンドユーザーが会社の業務機能にアクセスできるようにすることです。

企業間（B2B）API は、組織間でビジネスのやり取りをするだけでなく、場合によっては消費者を巻き込んだ形で相互にやり取りできるように組織が設置する API です。

B2B API により、エンタープライズがサプライヤー、再販業者、その他のパートナーとの連携を合理化したり、顧客により優れた体験を提供できるようになります。

B2B API の応用例を以下に示します。



API はそれぞれ利用者が大きく異なるため、API の保護に適用できるセキュリティ制御もそれぞれ異なります。業界はごく最近まで B2C のユースケースを重要視してきましたが、そのようなケースでも、B2C API のセキュリティ確保よりも、むしろ Web アプリケーションのセキュリティ確保に力を入れてきました。B2C Web アプリケーションのセキュリティ保護に一般的に採用されているセキュリティツールと制御機能は、特定の利点（Web Application Firewall（WAF）/Web Application and API Protection（WAAP）など）がありますが、B2C API を攻撃から保護するために必要な可視性、リアルタイム監視、保護のレベルを提供することはできません。

B2B API の保護はますます困難になっています。これらの API は、必要不可欠な保護メカニズムが欠けていることが多く、攻撃者の標的となりやすい傾向があります。以前の API セキュリティツールでは、B2B API の可視性が制限され、共有ユーザーの代わりに大量のデータアクセスを促進する API の保護に苦労していました（オープンバンキングで見られたように、フィンテック企業や金融機関が顧客データを合意の上で共有する場合など）。しかし、新しい API セキュリティソリューションは、ふるまい分析を提供し、異常な活動を認識できるため、これらの懸念に効果的に対処できます。

プライベート API とパブリック API の違い

プライベート API は、内部 API と呼ばれることもあり、企業の開発者や業務委託先が使用することを目的とした API です。サービス指向アーキテクチャ（SOA）の取り組みの一環として使用されることが多く、別々の部署や事業部門が互いのデータに効率的かつ効果的にアクセスできるようにして、内部的な開発を合理化することを目的としています。

一方で、パブリック API は外部 API とも呼ばれ、社外の顧客に公開する API です。これを極端に推し進めたものが、オープン API と呼ばれる、誰でも自由に利用できる API です。いずれのケースでも、社外のエンジニアが使用しても問題がないように、管理を厳重にし、文書化を徹底する必要があります。

注意すべき重要な点は、インターネット経由でアクセスできるプライベート API は、厳密な意味ではプライベートではないということです。たとえば、ACME の B2C API が ACME のモバイルアプリ（ACME の社内エンジニアが開発したアプリ）での使用に限定されるとします。この API をプライベート API と呼びたくなるかもしませんが、この API へのトラフィックは、インターネット（社外）を経由します。したがって、実際にはプライベートではありません。単に外部のユーザーには公開されていないだけです。ハッカーは、トラフィックを傍受し、モバイルアプリをリバースエンジニアリングして、該当する API を探し出し、このような API を日常的に攻撃しているのです。

API セキュリティの説明

API セキュリティとは

API セキュリティは、エンタープライズ全体のすべての API を可視化し、厳密なテストを実施し、保護するための戦略です。これには、アプリケーション、ビジネスプロセス、クラウドワークロードに不可欠な API が含まれます。しかし、内部 API と外部 API の両方が非常に迅速かつ大量に作成されているため、組織全体の API 環境を完全に把握することは困難です。多くの組織では、実際に使用している API の数と、呼び出された際に機密データを返す API の種類を把握できていません。API セキュリティリスクを特定して緩和するためには、このような可視性とデータ分析を提供する高度なセキュリティ制御が必要です。保護が必要な API の例を以下に示します。

- 顧客やビジネスパートナーがデータに簡単にアクセスできるようにする API
- ビジネスパートナーが利用する API
- 標準化されたスケーラブルな方法でアプリケーションの機能やデータをさまざまなシステムやユーザーインターフェースを介して利用できるようにする、内部的に実装され使用される API

効果的な API セキュリティ戦略には、リスクとその潜在的な影響を評価し、適切な緩和策を講じることができるよう体系化された技法を盛り込む必要があります。リスク評価の第一歩は、まず組織が公開し使用している API のうち、許可しているものと許可していないものをすべて目録化することです。この目録に含めるべき属性は次のとおりです。

- 最低限、「機密性がない」、「機密性がある」、「機密性が高い」データを区別したデータ分類
- API の脆弱性や設定ミスなどのリスク指標

その他にも、API の可視化とリスク緩和策は、以下に示すように、潜在的な脅威のさまざまな組み合わせを考慮したものでなければなりません。

- 許可されていないシャドウ API の使用を検知し、防止する（サイドバー参照）
- 攻撃者が悪用する可能性がある API の脆弱性や設定ミスを特定し、修正する
- ビジネスロジックの悪用やデータスクレイピングなどのさまざまな API 悪用を防止する

API セキュリティとアプリケーションセキュリティの違い

API セキュリティと従来のアプリケーションセキュリティは関連性の高い分野ではありますが、規模と問題の複雑さという 2 つの大きな理由から、API セキュリティは明確に異なる問題となっています。

規模の拡大

以下に示すように、API の利用が急増した要因は 3 つあります。

1. マイクロサービス（サービス間通信に API 使用を前提としたアーキテクチャ）の利用が拡大しています。
2. ダイレクト・ユーザー・チャネルでは、React、Angular、Vue などの最新のフロントエンド・アプリケーション・フレームワークが API を使用し、従来型の Web アプリに取って代わろうとしています。
3. まったく新しいチャネル（パートナー、IoT、業務自動化など）の対応にも API が使用されるようになっています。

柔軟性は複雑さにつながる

Web アプリケーションとは異なり、API はさまざまな方法でプログラマティックに使用するように設計されているため、正当な使用と攻撃や悪用を区別するのは極めて困難です。

セキュリティチームが理解しておくべき API の分類手法

セキュリティの立場から見たときの API の一般的な分類とその説明を以下に示します。

許可された API

公開 API（Swagger ドキュメンテーションまたは同等のものを含む）

許可されていない API

- シャドウ API
- 不正な API
- ゾンビ API
- 埋もれている API

古い API

- 非推奨の API
- 古いタイプの API
- ゾンビ API
- 孤立した API

API 保護のベストプラクティス

API セキュリティの強化は、次に示すベストプラクティスから始まります。

- API セキュリティの標準と実践を組織のソフトウェア開発ライフサイクルに組み込みます。
- 継続的インテグレーション／継続的デリバリー（CI／CD）パイプラインに API の文書化と自動化されたセキュリティテストを組み込みます。
- 適切で効果的な認証と認可の制御が API に確実に適用されるようにします。
- API の悪用や過負荷を避けるために、レート制限を導入します。
- 分散型サービス妨害（DDoS）攻撃のリスクを緩和するために、専用のゲートウェイやコンテンツ・デリバリー・ネットワークを導入し、レート制限やアプリケーションレベルの対策を強化します。
- アプリケーションテストのプロセスの幅を広げ、API セキュリティテストをその一部として盛り込みます。
- API の探索を継続して行うようにします。
- OWASP Top 10 API Security Risks など、一般的な API の脆弱性を特定し、修正するための体系的なアプローチを採用します。
- 既知の API 攻撃に対する基本的なレベルの防御として、シグネチャーベースの脅威検知と防御を使用します。
- 新しい脅威に対する API の脅威検知のスケーラビリティ、正確さ、ビジネスとの関連性、耐障害性を高めるために、シグネチャーベースの検知を AI とふるまい分析で補強します。
- いくつもの API セッションに対して API セキュリティの監視や分析のプロセスを何週間にもわたって実施できるようにします。
- 脅威ハンター、開発者、DevOps、サポート担当者が使用する API の目録とアクティビティデータにオンデマンドでアクセスできるようにして、API セキュリティの監視とアラートを補強します。

これらの API セキュリティのベストプラクティスを実装できるかどうかは、API セキュリティ戦略の成熟度によって異なります（サイドバーを参照）。

API セキュリティ成熟度の段階

ステージ 1：可視性と探索

自動化されたアプローチを使用して、すべての API とそれらがサポートするマイクロサービスを探索するプロセスにあるとします。見落とされた API（使用しなくなった API など）は攻撃者にとって格好の標的となるため、対象範囲の広さは非常に重要です。

ステージ 2：テスト

すべての API をテストし、正しくコード化され、目的の機能が実行されていることを確認します。API を展開する前に実行されるテストは、この成熟度ステージの最上位に位置づけられます。API が本番環境に移行する前にリスクが排除され、必要な修正にかかるコストも大幅に抑えられます。

ステージ 3：リスク監査

API 環境全体を継続的に監査し、誤って設定された API やその他のエラーを特定します。また、監査により、すべての API を適切に文書化し、機微な情報が含まれているか、適切なセキュリティ制御が施されているかを確認します。

ステージ 4：ランタイム保護

自動ランタイム保護機能を備えたソリューションを使用しているため、通常の API アクティビティと異常な API アクティビティを区別できます。この方法で API インタラクションを監視することで、脅威を示すふるまいをリアルタイムで検知できます。

ステージ 5：対応

疑わしい API のふるまいに対応するソリューション、たとえば WAF や API ゲートウェイなどを導入して、疑わしいトラフィックが重要なリソースにアクセスする前にブロックできるようにしています。ソリューションでは、カスタマイズされた自動化ルールが使用されます。

ステージ 6：脅威の検知

過去の脅威データに対してフォレンジック分析を定期的に実施し、アラートが脅威を正確に特定したかどうか、また、高度なツールとヒューマンインテリジェンスを組み合わせて事前対応型の脅威ハンティングを可能にするパターンが出現したかどうかを確認します。

API のセキュリティリスクと悪用

API の脆弱性とは

API の脆弱性は、ソフトウェアのバグやシステムの設定エラーです。攻撃者はこれを悪用して、機密性の高いアプリケーション機能や機微な情報にアクセスしたり、API を悪用したりします。OWASP Top 10 API Security Risks は、最も広く悪用されていて組織が特定し修正する必要のある API 脆弱性の概要を提供するものであり、有用です。

OWASP Top 10 API Security Risks には、すべての API 脆弱性が含まれているか

API セキュリティ対策の向上を図ろうとしている組織にとって、OWASP API Security Top 10 はちょうどよいスタート地点となるでしょう。そのカテゴリーは、API の潜在的なリスクを幅広くカバーしています。しかし、OWASP API Security Top 10 に含まれるカテゴリーは非常に広範囲なので、それぞれのサブエリアに掘り下げていくことが重要です。API の攻撃者は、かなりの頻度で（OWASP が広くカバーしている）認可の問題を悪用しようと試みますが、ロジック上のバグの悪用のように、OWASP API Security Top 10 ではカバーされていない API リスクも存在します。

API 悪用の仕組み

API の攻撃や悪用の方法は多岐にわたり、最も一般的な例としては次のようなものがあります。

- **脆弱性の悪用：**基盤インフラに技術的な脆弱性が存在すると、サーバーの侵害につながります。これらの例は、Apache Struts の脆弱性（CVE-2017-9791、CVE-2018-11776）から Log4j の脆弱性（CVE-2021-44228）まで多岐にわたります。

- **ビジネスロジックの悪用：**ロジックの悪用とは、攻撃者がアプリケーションの設計上または実装上の欠陥を突いて、想定外のふるまいや許可されていない動きをさせることです。これらのシナリオは、従来のセキュリティ制御が役に立たないため、CISO やそのチームにとってストレスとなります。

- **不正なデータアクセス：**よくある API 悪用の 1 つとして挙げられるのが、不備のある認証メカニズムを悪用して、アクセスされるべきではないデータにアクセスすることです。このような脆弱性には、オブジェクトレベルの認可の不備（BOLA）、安全でないオブジェクト直接参照（IDOR）、機能レベルの認可の不備（BFLA）といったさまざまな名称が付けられています。

- **アカウントの乗っ取り：**認証情報の窃盗やクロスサイトスクリプティング（XSS）攻撃の後には、アカウントが乗っ取られる可能性があります。そのようなことが起こると、極めてよく記述され完全にセキュリティが確保された API でさえ悪用される可能性があります。ふるまい分析を備えた API セキュリティソリューションを使用すると、認証されたアクティビティと不正使用を区別できます。

- **データスクレイピング：**組織がパブリック API を介してデータセットを利用可能にすると、攻撃者はそのリソースを積極的にクエリーして大量の貴重なデータセットを大規模に取得することができます。

- **ビジネスサービス妨害（DoS）：**API 攻撃者は、バックエンドに重いタスクを実行するようにリクエストすることで、アプリケーションレイヤーで「サービスの崩壊」、つまり完全な DoS を引き起こすことができます（これは GraphQL の非常に一般的な脆弱性ですが、リソース集約型のあらゆる API エンドポイント実装で発生する可能性があります）。

ゾンビ API とは

市場やビジネスのニーズの変化に伴い、API の状況も常に変化しています。新しいビジネスニーズへの対応、バグの修正、技術的な改善のために新しいエンドポイントが実装されると、古いバージョンのエンドポイントは使われなくなります。しかし、古いエンドポイントの廃止プロセスを管理することは、それほど簡単な話ではありません。廃止されたはずのエンドポイントがいまだに利用可能で、アクセスできるというようなことはよくあります。このようなエンドポイントをゾンビエンドポイントと呼びます。

さまざまなタイプのシャドウ API の見つけ方

エンタープライズ全体のシャドウ API を探索する方法の 1 つは、ネットワーク上の API トラフィックをインジェストして分析することです。API トラフィックソースの例として、以下があります。

利用可能なすべてのソースから生データを収集したら、AI 手法を使用して、すべての API、エンドポイント、パラメーターを、包括的な目録に変換することができます。そこからさらに分析を行い、これらの要素を分類し、シャドウ API のうち排除すべきもの、正式なガバナンスプロセスに組み込むべきものを振り分けることができます。

内部 API と B2B API を保護する方法

実際には、「内部」の定義が何かによって大きく左右されます。インターネット経由で組織の Web アプリケーションやモバイルアプリケーションにアクセスするための公開 API のことを「内部 API」と呼ぶチームもあります。会社の従業員や業務委託先しかこのような API のドキュメントを閲覧できませんが、ハッカーにとって、Burp Suite のようなアプリ逆アセンブルツールキットやプロキシを使用して、アプリを分析し、そのアプリへの API をリバースエンジニアリングすることは、たやすいことなのです。

しかし、「内部 API」の定義が East-West API であり、組織外からアクセスできないものであれば、主な脅威はインサイダーの脅威に絞られます。East-West API と B2B API は、他のほとんどの API と同様に、以下の方法によって保護します。ソフトウェア開発ライフサイクル（SDLC）のセキュリティを確保し、アクセスが認証および認可されていることを確認します。また、クォータ管理機能、レート制限、および Spike Arrests を実装することもできます。さらに、WAF や WAAP を使用して、既知の脅威から API を保護することもできます。B2B API の場合は、トランザクションの機密性が高く、かつ大量に行われることが多いことを考慮し、mTLS などの厳格な認証メカニズムを追加することを検討します。

次に、East-West API と B2B API 両方のセキュリティを確保するために、ふるまい分析を採用することをお勧めします。特に、関係するエンティティが多く、正当なふるまいと不当なふるまいを区別しにくい場合にお勧めします。以下に例を示します。

特定のユーザーの API 認証情報が漏えいしたかどうかを確認する場合

パートナーが請求書 API を悪用し、請求書番号を通じてアカウントデータを盗んでいるかどうかを検知する場合

B2B API と East-West API の保護には、IP アドレスや API トークンなどの技術的要素だけを分析しても得られないビジネスコンテキストが必要です。機械学習やふるまい分析を使用してビジネス関連のエンティティを可視化する方法が、リスクを効果的に把握、管理する唯一の手段です。ユーザーやパートナーのような個別のエンティティごと、あるいは場合によってはビジネスプロセスのエンティティ（請求書、支払い書、注文書など）ごとに API が正常に使用されているかどうかをビジネスコンテキストや過去のベンチマークで調べれば、それ以外の方法では検知できないような異常も見つけることができます。

API ゲートウェイは十分なリスク保護を提供しているのか

API に対して戦略的なアプローチを採っている組織の多くが、API ゲートウェイを使用しています。ほとんどの API ゲートウェイは、組織が活用すべきセキュリティ機能を豊富に取り揃えています。その機能の中でも筆頭なのが、認証（および OpenID Connect を活用できる場合は認可）です。しかし、API ゲートウェイで認証、認可、クォータの管理を実行するだけでは不十分です。その理由は次のとおりです。

API ゲートウェイで探索できないもの：API ゲートウェイは、管理するように設定された API のみを可視化し、制御するため、シャドウ API やエンドポイントを効果的に検知することができません。

API ゲートウェイのセキュリティギャップ：API ゲートウェイは、認証スキームと、ある程度の認可スキームを強制することはできますが、（WAF や WAAP とは異なり）ペイロードを検査することも、ふるまいを評価して悪用を検知することもできません。

最もよく見られる API の設定ミス

API の用途は多岐にわたるため、API の設定ミスを挙げようとしたらきりがありません。ただし、設定ミスにはいくつかの共通したテーマがあります。

認証処理に問題があるか、そもそも認証処理がない

API を介してアクセスできる機微な情報のセキュリティを確保するうえで、認証は不可欠です。最初の一歩は、機微な情報にアクセスする API がすべて、初めから認証されるようにすることです。しかし、総当たり攻撃、Credential Stuffing、レート制限を悪用した認証トークンの窃盗から認証メカニズムを保護することも重要です。設定に誤りがあると、API 利用者が認証メカニズムを迂回できることもありますが、それは得てしてトークン管理（たとえば、悪名高い JWT の検証の問題、トークンのスコープをチェックしないという問題）に関する問題になりがちです。

認可処理に問題がある

API の最も一般的な用途の 1 つは、機微な情報を含むデータやコンテンツにアクセスできるようにすることです。認可とは、API 利用者がアクセスしようとしているデータにアクセスする資格があることを、実際に利用する前に検証するプロセスです。このプロセスは、オブジェクトやリソースのレベル（自分の注文書にはアクセスできるが他人の注文書にはアクセスできないなど）で行われることもあれば、職務レベル（管理権限など）で行われることもあります。エッジのケースと条件は大量に存在し、また、マイクロサービス間で API コールが取り得るフローは多岐にわたるため、認可を正しく行うことは至難の技です。認可エンジンが一元化されていない場合は、API の実装に、BOLA や BFLA のような脆弱性が存在する可能性が高くなります。

セキュリティ上の設定ミス

上記の認証および認可の問題に加えて、安全でない通信（SSL/TLS の不使用や脆弱な暗号スイートの使用など）、保護されていないクラウドストレージ、過度に許可されているクロスオリジンリソース共有ポリシーなど、セキュリティ上の設定ミスにはさまざまな種類があります。

リソースとレート制限の欠如

API を実装するにあたって、API 利用者の呼び出しの回数に制限がない場合、攻撃者は、システムリソースをいくらでも消費できるため、サービスの低下や大規模な DoS 攻撃が発生する可能性があります。最低限でも、エンドポイントの認証は極めて重要であるため、認証されていないエンドポイントに対するアクセスにはレート制限を適用する必要があります。そうしないと必然的に、総当たり攻撃、Credential Stuffing、認証情報検証攻撃を受けることになります。

API 攻撃とは

API 攻撃とは、API を悪意のある目的、あるいは許可されていない目的で使用する試みのことです。API 攻撃には、次のようなさまざまな形態があります。

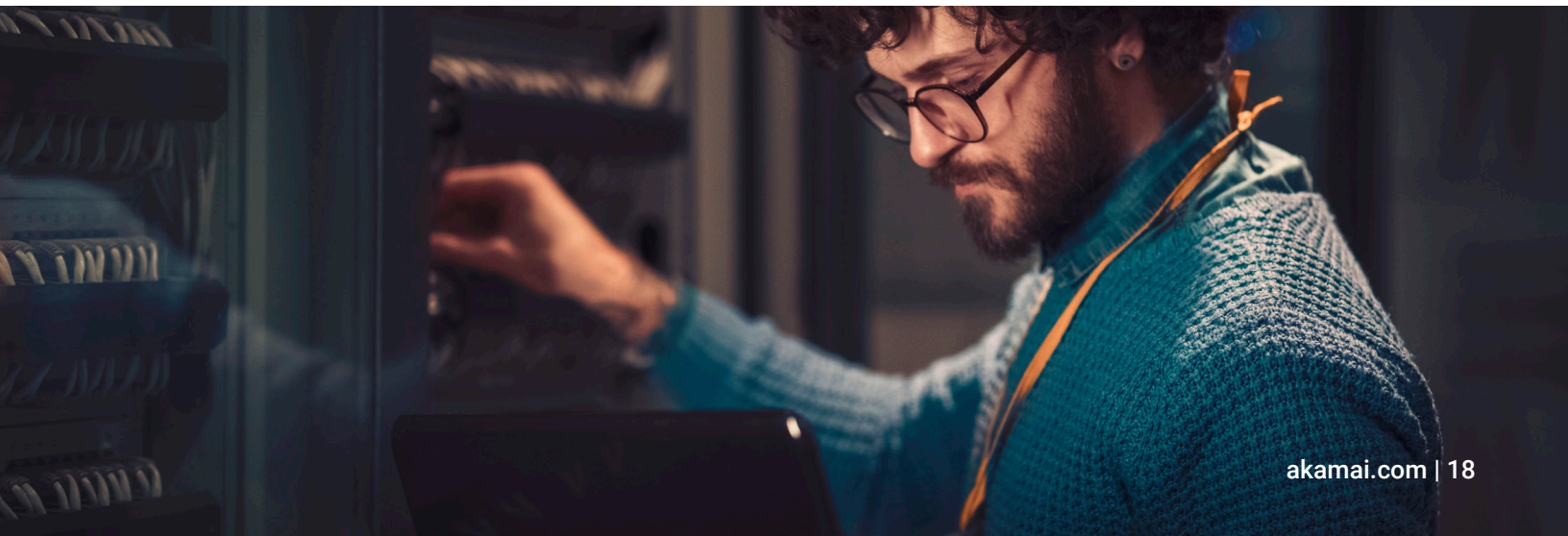
- API 実装における技術的な脆弱性の悪用
- 正当なユーザーになりすますために、盗んだ認証情報やその他のアカウント乗っ取り技術を使用
- ビジネスロジックを悪用すれば想定外の方法で API を使用することが可能

API の Credential Stuffing とは

Web サイトや Software as a Service（SaaS）プラットフォームからのユーザー ID とパスワードの流出は、日常茶飯事となっています。多くの場合、このように流出した大量の認証情報は、オンラインで広く共有されてしまいます。Credential Stuffing とは、過去に攻撃を受けた Web サイトから流出した認証情報を使用して、自動的に他の Web サイトにログインしようとする攻撃のことです。この手法は、一定の割合のユーザーが複数のサイトで同じ認証情報を使用しているという前提に基づいています。攻撃者はますます API に直接狙いを定めるようになり、その認証メカニズムを標的にしています。API は簡単に利用できるようになっているため、攻撃者にとって自動的な攻撃の対象にしやすいのです。

API 経由のデータ窃取とは

データ窃取は、API 攻撃と悪用が成功した場合に頻繁に発生します。攻撃者が API 攻撃を通して機密性の高い非公開情報を盗むことを指す場合もあります。しかし、API の悪用は、一般に公開されているデータを積極的にデータスクレイピングして、集合体として価値のある大規模なデータセットを作成するような、それほど深刻ではない場合にも使用されます。



API セキュリティの対策と傾向

API セキュリティの最新動向

セキュリティ関連の責任者が API セキュリティの戦略を立てるうえで考慮すべき主な傾向を以下に示します。

ふるまい分析と異常検知：リスクを緩和するために、発生し得る攻撃を予測しようとしたり、シグネチャーベースの検知やあらかじめ定義されたポリシー（WAF など）だけに頼ったりするのではなく、機械学習やふるまい分析を徐々に追加していき、ビジネスコンテキストに従って API のアクティビティを視覚化し、異常を検知できるようにする組織が増えています。

オンプレミスから SaaS への移行：第 1 世代の API セキュリティ製品の多くはオンプレミス環境に展開されてきましたが、SaaS ベースのアプローチは、展開の速さと容易さに加えて、規模に応じて機械学習の能力を利用できるため、徐々に普及しています。

時間枠を広くとった分析：個々の API コールや短期的なセッション活動のみを分析する API セキュリティのアプローチは廃れつつあり、その代わりに、基本的な WAF ポリシー最適化の自動化から、ふるまい分析や異常検知に至るまで、数日、場合によっては数週間にわたって API の動きを分析することができるプラットフォームが台頭してきています。

DevSecOps ― セキュリティ以外のステークホルダーの関与：API のリスクを軽減する最善の方法の 1 つは、API セキュリティの戦略やツールと、API の作成、実装、設定に関与している開発者やシステムとのつながりを強めることです。

API の情報に基づく API セキュリティ：活発な API 攻撃や悪用の事例を検知し、緩和することが重要である一方、先進的な組織は、脅威ハンティング、インシデント対応、API 開発の実践方法を改善するために、API セキュリティに関するデータや知見にオンデマンドでアクセスできるようにしています。

シグネチャーベースの API セキュリティとは

シグネチャーベースの API セキュリティ手法とは、既知の攻撃の特徴やパターンが発生していないかどうかを監視し、それに合致するものがあれば、セキュリティのアラートを発したり、自動的に対応したりする手法です。この典型的な例が WAF です。シグネチャーベースの検知の価値は、インラインであること、つまり即座に脅威をブロックできることです。したがって、API のトラフィックを受信したときに、そのトラフィックが攻撃であること、あるいは異常なふるまいをすることがわかっていれば、シグネチャーベースの API セキュリティで脅威を即座にブロックできます。

より大規模な WAAP ソリューションの一部である WAF は、攻撃シグネチャーパターンから学習し、規模に応じてアジャイルに対応できる機械学習を通じて高度な検知を提供できます。ふるまい分析や応答のカスタマイズ機能を備えた API セキュリティソリューションと連携できる WAAP であれば、両方のメリットを活かすことができます。このようにソリューションを連携させれば、社内外で包括的に API の可視化、検知、対応ができるようになります。

API の検知と対応とは

API の検知と対応は、過去のデータを深く分析することに焦点を当てた、新しいカテゴリーの API セキュリティです。その目的は以下のとおりです。

- すべての API 利用者の行動のベースラインを判断する
- API の悪用や誤用の可能性を示す攻撃や異常を検知する

機械学習の手法に使用されるデータセットは巨大であり、リソースを大量に消費するため、規模に応じて効果的に API の検知と対応を行うためには、SaaS モデルを使用する以外に方法がありません。

高度な API 脅威防御とは

高度な API 脅威防御とは、ふるまい分析と脅威ハンティングを組み合わせた API セキュリティを SaaS ベースで実装するアプローチのことです。その目的は以下のとおりです。

- 組織で使用されている API（シャドウ API やゾンビ API を含む）をすべて探索する
- API がどのように使用されているか、悪用されているかというビジネスコンテキストを加味するために機械学習を応用する
- API および API のアクティビティデータで、ふるまい分析と脅威ハンティングを実行する

API セキュリティプラットフォームとは

API セキュリティプラットフォームとは、次の目的に特化した SaaS ベースのサービスのことです。

- エンタープライズ規模で使用されているすべての API（認可されているかどうかにかかわらず）を目録化し、継続的に更新する
- API とその使用状況を分析して、ビジネスコンテキストを探索し、想定されるふるまいのベースラインを判断する
- API の使用状況の中から異常を検知し、必要に応じて、セキュリティ情報およびイベント管理（SIEM）、セキュリティのオーケストレーション、自動化および応答（SOAR）ワークフローにアラートとその裏付けとなるデータを送る
- セキュリティのステークホルダーと、セキュリティ以外のステークホルダーの両方が、API の目録やアクティビティ、脅威に関する情報にオンデマンドでアクセスできるようにする

API セキュリティ企業とは

今では IT 担当やセキュリティ担当の責任者が API をより戦略的に使用できるようになり、API 専用のパートナーとの連携が必要になったと言えるでしょう。API 企業のタイプを大別すると以下の 3 つになります。

- API コールを一元的に受け付け、適切なバックエンドリソースやマイクロサービスに振り分けるテクノロジーを保有する API ゲートウェイ企業
- 組織がすべてのアクティブな API と潜在的なリスクを認識し、攻撃や悪用の事例を検知し、包括的なセキュリティテストを可能にし、API の使用状況に関する多様なデータを保有している API セキュリティプラットフォーム企業
- API トラフィックデータをシームレスに転送することができ、同時に内外のプラットフォームの API を探索する能力を備えた WAAP および API セキュリティプラットフォーム企業（ベンダーとの連携とデジタルギャップの縮小に理想的）

API における脅威ハンティングとは

脅威ハンティングでは、未知の脅威や過去に検知されなかった脅威を積極的に検索します。このプロアクティブなアプローチは、これまでに見られなかった新たな脅威を特定し、重大な損害を引き起こす前に緩和するために重要です。ふるまい分析は、脅威ハンティングで使用される主な手法の 1 つです。これには、API のふるまいを分析して、疑わしいアクティビティや異常なアクティビティを特定することが含まれます。たとえば、API が短期間に数千件のレコードを突然要求した場合、API ビジネスロジックが侵害されている可能性があります。最新の API セキュリティソリューションは、セキュリティチームが潜在的な脅威を早期に特定し、対策を講じるための脅威ハンティング機能を備えています。

WAAP とは

Web Application and API Protection（WAAP）は、調査会社 Gartner が、新たに出現する Web および API の保護ソリューションに関する業界カバレッジに使用している分類です。これは、WAF 市場の以前の業界カバレッジが進化したもので、API セキュリティの戦略的重要性の高まりと、WAF プラットフォームがマネージド型 SaaS としてクラウドに移行していることに対応しています。

API ドキュメンテーションの例

RESTful API（最も一般的な Web API）に関する API ドキュメンテーションでよく使われる形式は、OpenAPI 仕様に準拠した Swagger ファイルの集まりです。API の設計時または実装時に開発者が API ドキュメンテーションを作成することが理想です。しかし実際には、API ドキュメンテーションは古くなることが非常に多く、その結果、API の実際の使い方とドキュメンテーションの記載内容が一致しなくなります。この問題に対処するために、一部の API セキュリティプラットフォームは、実際の API アクティビティをもとに Swagger ファイルを生成することができるため、ドキュメンテーションの記載内容と、実際に展開されている API との間の不一致を明確にできます。まさに、あらゆる API リスク評価に不可欠なコンポーネントです。

企業が従うべき API セキュリティチェックリストとは

効果的な API セキュリティには、組織に特有のいくつもの細かいステップと継続的な実践が必要になりますが、以下に示す API チェックリストを使用すれば、セキュリティチームが API セキュリティの向上を図るうえで良いスタートを切れるでしょう。

- API セキュリティのアプローチに、エンタープライズ規模の API 探索を継続させるための仕組みが盛り込まれているか
- API 対策管理は、組織のより広範なセキュリティおよびリスク管理の実践に統合されているか
- 個別のデータセンターモデルやクラウド・インフラ・モデルに捉われない汎用的な API セキュリティアプローチを採っているか
- 観測している API アクティビティと潜在的なリスクを正しく理解するうえで必要なビジネスコンテキストがチームに伝達されるアプローチになっているか
- API セキュリティプラットフォームとその他の関連ビジネスプロセス（SIEM／SOAR、脅威ハンティング、ドキュメンテーション、DevOps ツールなど）との間で、自動的に双方向でやり取りできる仕組みを導入しているか
- 開発者のようなセキュリティ以外のステークホルダーが API セキュリティのツールやプロセスにスムーズに馴染めるような段取りを組んでいるか

Akamai のセキュリティは、パフォーマンスや顧客体験を損なうことなく、ビジネスを推進するアプリケーションをあらゆる場面で保護します。当社のグローバルプラットフォームの規模と脅威に対する可視性を活用して、お客様と Akamai が提携し、脅威を防止、検知、緩和することで、ブランドの信頼を構築し、ビジョンを実現できます。Akamai のクラウドコンピューティング、セキュリティ、コンテンツデリバリーの各ソリューションの詳細については、akamai.com および akamai.com/blog をご覧いただくか、X（旧 Twitter）と LinkedIn で Akamai Technologies をフォローしてください。公開日：2024 年 9 月。