Engineer

# JWT

JSONでクレームを運ぶ、URLにも使えるコンパクトなトークン

## どのようなものか

JWTは、ある主体から別の主体へ、受信側が検査できる形で、一連の主張を運ぶ短い文字列です。JSON Web Tokenの頭文字を取っています。RFC 7519は、2015年5月に、URLに安全に入れられる、コンパクトなクレームの表現方法として定義し、発音は「jot」と説明しています。クレームは、ある主体について主張する、名前と値の組です。`"sub": "24400320"` は、誰を対象とするトークンかを示し、`"exp": 1300819380` は、いつ有効でなくなるかを示します。

通信上のJWTは、base64urlの文字列をドットでつないだものです。署名付きでは三部分あり、アルゴリズムを示すヘッダー、クレームを持つペイロード、最初の二部分から計算する署名からなります。暗号化したものは五部分で、鍵がなければクレームを読めません。RFC 7519自体は、暗号処理の方法を定めません。署名付き、またはメッセージ認証コードで保護するトークンには、JSON Web Signature（RFC 7515）を、暗号化するトークンには、JSON Web Encryption（RFC 7516）を使います。アルゴリズムはRFC 7518に列挙され、鍵はRFC 7517のJSON Web Key形式で公開できます。

`HS256` は、SHA-256を使うHMACです。一つの共有秘密で、生成も検証も行うため、検証できるサービスは、どれもトークンを発行できます。`RS256` と `ES256` は、鍵ペアを使います。発行者が秘密鍵で署名し、ほかの主体は、自由に配布できる公開鍵で検証します。RFC 7519が実装を必須とするのは、`HS256` と、署名なしの `none` だけで、`RS256` と `ES256` は推奨です。

第4.1節は、七つのクレーム名を登録しています。`iss`（発行者）、`sub`（対象主体）、`aud`（受信対象）、`exp`（有効期限）、`nbf`（有効開始時刻）、`iat`（発行時刻）、`jti`（リプレイ攻撃を防ぐためにも使える、一意のJWT ID）です。コンパクトさを目指す形式なので、すべて三文字で、どれも任意です。必須

のクレームを決めるのは、使うアプリケーションです。登録済みの集合のほかに、誰でも、IANAに登録した公開クレーム、URLなど他者と衝突しない名前を付けた公開クレーム、二者間で合意する非公開クレームを定義できます。

概念を判断に生かす

「JWTとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 アカウント、パスワード、権限を整理し、業務に必要なアクセスだけを付与し、退職者の権限は当日に解除します。

**サービスを見る** お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 Linuxサーバーを守る1,000件のRBACルールを作成 — 1,000 セキュリティ上の穴を塞ぐアクセスルール

**Linuxサーバーを守る1,000件のRBACルールを作成** お問い合わせ →

## なぜ重要なのか

JWTの要点は、受信側が単独で検査できることです。発行者の鍵と時計があれば、データベースを調べず、発行者を呼び出さずに、真正なトークンか、誰に向けたものか、まだ有効かを判断できます。2021年10月のRFC 9068は、多くの商用OAuth実装が、リソースサーバーが直接解析、検証するアクセストークンを発行していると記録しています。発行するサーバーと受け入れるサーバーが、何らかの境界で分かれる場合に一般的な方式で、その多くがJWTを使います。同RFCは、ベンダーをまたいで使えるよう、その構造を標準化しました。

OpenID Connectを支える形式でもあります。すべてのIDトークンは、それを求めたアプリケーションに宛てたJWTです。`alg` を `none` にするための二つの条件を両方満たさない限り、プロバイダーは署名しなければなりません。第一は、応答タイプが認可エンドポイント（Authorization Endpoint）からIDトークンを返さないことです。認可コードフロー（Authorization Code Flow）が、その例です。第二は、クライアントが登録時に `none` を明示的に要求していることです。2015年にメディアタイプ `application/jwt` を登録した際の文書は、利用者として既にOpenID Connect、Salesforce、Google、Android、Windows Azure、Amazon Web Servicesを挙げています。

RFC 7519は、AuthorizationヘッダーやURIのクエリーパラメーターなど、長さに制約のある場所を想定しています。そのため、クレーム名は短く、エンコードした文字列も、エスケープせずURLに使えます。

検証の負担を小さくする同じ性質が、取り消しを難しくします。署名と時計を検証する受信側は、その間にアカウントに何が起きても、`exp` までトークンを受け入れます。

# 実際の活用例

RFC 7519は、一つの例を通して説明しています。ヘッダーは、オブジェクトがJWTであり、HMAC SHA-256で保護されていると宣言します。

```
{"typ":"JWT",
 "alg":"HS256"}
```

クレームは、発行者、有効期限、URLを名前とする公開クレーム一つを示します。URLにすることで、他者が定義するクレームと衝突しないようにします。

```
{"iss":"joe",
 "exp":1300819380,
 "http://example.com/is_root":true}
```

各部分をbase64urlでエンコードし、ヘッダーとペイロードをドットでつないだ文字列からHMACを計算し、三つの結果を一つの文字列につなぎます。RFCの改行は、表示のためだけのものです。

```
eyJ0eXAiOiJKV1QiLA0KICJhbGciOiJIUzI1NiJ9
.
eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMDA4MTkzODAsDQogImh0dHA6Ly9leGFt
cGxlLmNvbS9pc19yb290Ijp0cnVlfQ
.
dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk
```

この例から、三つのことが分かります。第一に、トークンを持つ人なら、誰でもクレームを読めます。base64urlはエンコーディングなので、中央の部分をデコードすれば、鍵なしでJSONが戻ります。第二に、署名が対象にするのは、送られたとおりのエンコード済み文字列です。RFCのヘッダーには、CRLFの改行と先頭の空白があり、そのため正確なバイト列を列挙しています。JSONをきれいに整形して再エンコードすれば、違う文字列になり、署名は一致しなくなります。例の鍵は、RFC 7515の付録A.1に掲載され、その鍵なら署名を検証できます。第三に、このトークンの期限は、とうに切れています。1970年1月1日から `1300819380` 秒後は、2011年3月22日の18:43 UTCなので、正しいバリデーターは、現在、このトークンを拒否します。

このようなトークンを受け取る側は、クレームを一つでも読む前に、定めた一連の検査を行います。文字列をドットで分割し、ヘッダーをデコードします。有効なJSONであることに加え、各パラメーターと値について、構文と意味を理解して対応できるか、未知の場合に無視してよいと明示的に定められていることを確かめます。`alg` を、この発行者と鍵について受け入れると決めたアルゴリズムと比較します。RFC 7519は、これをアプリケーションの判断と明言し、署名が有効でも、アルゴリズムを受け入れられなければ拒否するべきとしています。発行者に属する鍵で署名を検証します。その後、クレームを検査します。`iss` は信頼する発行者、`aud` は受信側自身を示さなければなりません。現在時刻は、時計のずれに最大数分を認めつつも、`exp` より前であり、`nbf` があればそれより後でなければなりません。それから初めて、クレームに基づいて処理します。

RFC 9068のJWTアクセストークンは、実際のプロファイルが加えるものを示します。RFC内のヘッダーは、`{"typ":"at+JWT","alg":"RS256","kid":"RjEwOwOA"}` です。明示的な型、非対称アルゴリズム、発行者が公開する鍵集合内の鍵IDを持ちます。このプロファイルは、`iss` 、`exp` 、`aud` 、`sub` 、`client_id` 、`iat` 、`jti` を必須にします。リソースサーバーは、`typ` が `at+jwt` か `application/at+jwt` であると確認し、ほかの型を拒否しなければなりません。これらのメディアタイプとサブタイプ名は、大文字小文字を区別しません。例の `at+JWT` も、同等の表記です。

トークン内のクレームは、署名した時点のスナップショットです。そのため、`exp` までに変わり得るものは、サーバーで判断する方が適切です。このサイトが紹介するアプリケーションの全層を強化するプロジェクトは、利用権限、すなわちティア（利用区分）、ロール、組織と船舶の適用範囲を、要求ごとにサーバーで再検査します。ブラウザーの制御は、利用体験のためだけとしています。

## 問題が起きるところ

**トークンが自分のアルゴリズムを選びます。** `alg` ヘッダーは柔軟性のためにありますが、RFC 8725は、一部のライブラリーが過度に信頼した例を説明しています。攻撃者が `none` へ変えると、ライブラリーが署名なしで受け入れる可能性がありました。また、`RS256` を `HS256` に変えると、攻撃者もダウンロードできる発行者のRSA公開鍵を共有秘密として、HMACを検査してしまう可能性もありました。受け入れるアルゴリズムをライブラリーに設定し、各鍵は厳密に一つのアルゴリズムだけで使い、`none` は無条件に拒否します。

**HMACの共有秘密に、パスワードを使います。** `HS256` で、短く覚えやすい秘密を使ったトークンは、オフラインで攻撃できます。トークンを一つ持つだけで、サーバーに触れず、自分のペースで候補を試せます。RFC 8725は、人が覚えられ

るパスワードをHMAC鍵に直接使うことを禁じ、RFC 7518は、`HS256` に少なくとも256ビットの鍵を求めています。暗号学的な乱数源から鍵を生成し、シークレットストアに保管します。多くのサービスが検証だけを必要とするなら、鍵ペアを優先します。

**署名が正しいため、すべてを信じます。** 有効な署名が証明するのは、誰が発行したかであり、誰に向けたものかではありません。一つの発行者が複数のサービスに提供し、受信側が `aud` の検査を省けば、一つのサービスへ正当に送ったトークンを、別のサービスに再利用できます。RFC 8725は、これを置換と呼び、そのようなトークンを受けるすべての側に、受信対象を検証し、それがないトークンを拒否するよう求めています。 `aud` を自分の識別子と照合し、署名鍵が、`iss` の示す発行者に属することも確かめます。

**別の種類のトークンを受け入れます。** 同じ発行者のIDトークン、アクセストークン、セキュリティーイベントトークンは、よく似ています。署名と発行者だけを調べる受信側は、種類を取り違え得ます。RFC 8725は、これを異種JWTの取り違え（cross-JWT confusion）と呼び、 `typ` ヘッダーによる明示的な型付けを勧め、RFC 9068は、 `at+jwt` で適用しています。種類ごとに独自の型か受信対象を用意し、ほかの種類の検証には必ず失敗するよう、検証規則を書きます。

**ヘッダーが鍵を選びます。** `kid` ヘッダーは、使う鍵を示し、 `jku` と `x5u` は、その取得先URLを指せます。どれも、送信者から届きます。未検査の `kid` をデータベースのクエリーやディレクトリー検索へ渡すと、システムへの侵入口になります。 `jku` や `x5u` の任意のURLへ接続するサーバーは、送信者が選ぶ任意の場所へ、要求を送らされ得ます。 `kid` は、既に保有する鍵の中だけで検索し、鍵の取得も、発行者が公開する `jwks_uri` など、許可リスト内の場所だけにします。

**誰でもペイロードを読めます。** 署名付きJWTは、クレームの改ざんを防ぎますが、閲覧は防ぎません。ブラウザー、ログ、プロキシーを通ることも多くあります。RFC 7519は、プライバシーに関わる機微な情報を含むことがあり、最も簡単な保護は、含めないことと述べています。含めるなら、暗号化するか、少なくともTLSを使う接続だけで送るべきです。RFC 9068は、発行者が形式を変え得るため、クライアントはアクセストークンを一切調べてはならないとも定めています。個人データでなく識別子を入れ、内容自体を非公開にする必要があれば暗号化します。

# 参考資料

- RFC 7519：JSON Web Token（JWT） — 形式自体の定義です。登録済みクレーム、具体例、検証手順を含みます。
- RFC 7515：JSON Web Signature（JWS） — ヘッダー、ペイロード、署名のエンコードと連結、上の例で使う鍵です。
- RFC 8725：JSON Web Tokenの現行ベストプラクティス — JWTへの既知の攻撃と、それを防ぐライブラリー、アプリケーションの規則です。
- RFC 7518：JSON Web Algorithms（JWA） — `alg` で使うアルゴリズム名と、各HMACの最小鍵長です。
- RFC 9068：OAuth 2.0アクセストークンのJWTプロファイル — JWTアクセストークンの標準構造と、リソースサーバーが必要とする検査です。

用語集

実際の制約に合わせて選ぶ

デプロイの流れ、運用上のリスク、緊急対応の担当者をお知らせください。検査、復旧手段、明確な責任の所在を伴う変更の範囲を定められます。

**料金を見る** お問い合わせ →

## 用語を実務に結び付ける

「JWTとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/jwt/