Engineer

JWT

Kompaktes, URL-sicheres Token mit Aussagen als JSON

Was es ist

Ein JWT ist eine kurze Zeichenkette, die Aussagen einer Partei an eine andere überträgt, und zwar in einer für den Empfänger prüfbaren Form. Die Buchstaben stehen für JSON Web Token. RFC 7519 definierte das Format im Mai 2015 als kompakte Darstellung von Claims, die sich sicher in eine URL aufnehmen lässt, und hält fest, dass der Name wie das englische Wort „jot“ ausgesprochen wird. Ein Claim besteht aus einem Namen und einem Wert und enthält eine Aussage über ein Subjekt: `"sub": "24400320"` sagt, auf wen sich das Token bezieht, `"exp": 1300819380` , wann es seine Gültigkeit verliert.

Bei der Übertragung ist ein JWT eine Folge durch Punkte verbundener Base64url-Zeichenketten. Ein signiertes Token hat drei Teile: einen Header, der den Algorithmus nennt, eine Nutzlast mit den Claims und eine über die ersten beiden Teile berechnete Signatur. Ein verschlüsseltes hat fünf Teile; seine Claims lassen sich ohne Schlüssel nicht lesen. RFC 7519 legt die kryptografischen Verfahren nicht selbst fest. Für signierte oder mit einem Nachrichtenauthentifizierungscode geschützte Token stützt es sich auf JSON Web Signature (RFC 7515), für verschlüsselte auf JSON Web Encryption (RFC 7516). Die Algorithmen stehen in RFC 7518; Schlüssel können im JSON-Web-Key-Format aus RFC 7517 veröffentlicht werden.

`HS256` ist ein HMAC mit SHA-256: Ein gemeinsames Geheimnis dient sowohl zum Erstellen als auch zum Prüfen des Tokens. Jeder Dienst, der es verifizieren kann, kann es daher auch ausstellen. `RS256` und `ES256` verwenden ein Schlüsselpaar: Der Aussteller signiert mit einem privaten Schlüssel, alle anderen prüfen mit dem öffentlichen, der frei verteilt werden kann. RFC 7519 schreibt nur die Implementierung von `HS256` und dem unsignierten `none` vor und empfiehlt `RS256` und `ES256` .

Abschnitt 4.1 registriert sieben Claimnamen: `iss` (Aussteller), `sub` (Subjekt), `aud` (Zielgruppe), `exp` (Ablaufzeitpunkt), `nbf` (nicht vor), `iat` (Ausstellungszeitpunkt) und `jti` (JWT-ID, eine eindeutige Kennung, mit der sich die erneute Verwendung eines Tokens verhindern lässt). Sie haben drei Buchstaben, weil das Format kompakt sein soll, und jeder einzelne ist optional. Welche Claims ein Token enthalten muss, entscheidet die verwendende Anwendung. Neben den registrierten kann jeder öffentliche Claims definieren, entweder bei der IANA registriert oder mit einem kollisionsfreien Namen wie einer URL, oder private Claims, die zwei Parteien miteinander vereinbaren.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist JWT?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Konten, Passwörter und Berechtigungen werden so geordnet, dass alle genau den beruflich erforderlichen Zugriff haben und ihn am Tag ihres Ausscheidens verlieren.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. 1.000 RBAC-Regeln für sichere Linux-Server — 1,000 Zugriffsregeln zum Schließen einer Sicherheitslücke

1.000 RBAC-Regeln für sichere Linux-Server Kontakt aufnehmen →

Warum es wichtig ist

Der Zweck eines JWT ist, dass der Empfänger es selbstständig prüfen kann. Mit dem Schlüssel des Ausstellers und einer Uhr erkennt ein Dienst, ob das Token echt ist, für wen es bestimmt ist und ob es noch gilt, ohne Datenbankabfrage und ohne Rückfrage beim Aussteller. RFC 9068 vom Oktober 2021 hält fest, dass viele kommerzielle OAuth-Implementierungen Zugriffstoken ausstellen, die Ressourcenserver direkt parsen und validieren. Dieses Muster ist dort verbreitet, wo ausstellender und annehmender Server durch eine Grenze getrennt sind; viele verwenden dafür JWT. Der RFC standardisierte anschließend den Aufbau, damit solche Token anbieterübergreifend funktionieren.

Es ist auch das Format hinter OpenID Connect: Jedes ID-Token ist ein JWT, das an die anfragende Anwendung adressiert ist. Der Anbieter muss es signieren, sofern nicht beide Voraussetzungen für `alg` mit dem Wert `none` erfüllt sind: Bei diesem Antworttyp wird kein ID-Token vom Authorization Endpoint zurückgegeben, wie im Authorization Code Flow, und der Client hat bei der Registrierung ausdrücklich `none` angefordert. Bei der Registrierung des Medientyps `application/jwt` im Jahr 2015 wurden OpenID Connect, Salesforce, Google, Android, Windows Azure und Amazon Web Services bereits als Nutzer genannt.

RFC 7519 sieht das Format für Bereiche mit begrenztem Platz vor, darunter den Authorization-Header und Parameter in der Abfrage einer URI. Deshalb sind Claimnamen kurz, und die kodierte Zeichenkette lässt sich ohne zusätzliches Escaping in einer URL verwenden.

Dieselbe Eigenschaft, die die Prüfung eines JWT günstig macht, erschwert seinen Widerruf. Ein Empfänger, der Signatur und Uhr prüft, akzeptiert das Token bis `exp`, unabhängig davon, was inzwischen mit dem Konto geschehen ist.

Wie es in der Praxis aussieht

RFC 7519 führt ein vollständiges Beispiel durch. Der Header erklärt das Objekt zum JWT und gibt den Schutz durch HMAC SHA-256 an:

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

Die Claims nennen einen Aussteller, einen Ablaufzeitpunkt und einen öffentlichen Claim mit einer URL als Namen, damit er nicht mit einem anderswo definierten Claim kollidiert:

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

Jeder Teil wird mit Base64url kodiert. Der HMAC wird über Header und Nutzlast berechnet, die durch einen Punkt miteinander verbunden sind. Die drei Ergebnisse werden zu einer Zeichenkette verbunden. Der RFC druckt sie nur zur Darstellung mit Zeilenumbrüchen:

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

Aus diesem Beispiel folgen drei Dinge. Erstens kann jeder, der das Token besitzt, die Claims lesen: Base64url ist eine Kodierung. Der mittlere Teil lässt sich ohne Schlüssel wieder in JSON dekodieren. Zweitens deckt die Signatur die kodierten Zeichenketten genau so ab, wie sie gesendet wurden. Der Header des RFC enthält einen CRLF-Zeilenumbruch und ein führendes Leerzeichen. Deshalb führt der RFC die exakten Bytes auf. Würde das JSON schöner formatiert und erneut kodiert, entstünde eine andere Zeichenkette, und die Signatur würde nicht mehr passen. Der Schlüssel für dieses Beispiel steht in Anhang A.1 von RFC 7515. Mit ihm lässt sich die Signatur verifizieren. Drittens ist das Token längst abgelaufen: `1300819380` Sekunden nach dem 1. Januar 1970 entsprechen dem 22. März 2011 um 18:43 UTC. Ein korrekter Validator weist es heute daher zurück.

Ein Empfänger führt bei einem solchen Token eine feste Folge von Prüfungen durch, bevor er einen einzigen Claim liest. Er zerlegt die Zeichenkette an den Punkten, dekodiert den Header und bestätigt, dass es sich um gültiges JSON handelt. Darin dürfen nur Parameter und Werte vorkommen, deren Syntax und Bedeutung er versteht und unterstützt oder die laut ihrer Spezifikation bei Unkenntnis ausdrücklich zu ignorieren sind. Er vergleicht `alg` mit den Algorithmen, die er für diesen Aussteller und Schlüssel akzeptieren will. RFC 7519 erklärt dies ausdrücklich zur Anwendungsentscheidung. Ein Token mit einem nicht akzeptablen Algorithmus soll selbst bei gültiger Signatur abgewiesen werden. Die Signatur wird mit einem zum Aussteller gehörenden Schlüssel geprüft. Danach folgen die Claims: `iss` muss einen vertrauenswürdigen Aussteller nennen, `aud` den Empfänger selbst. Die aktuelle Zeit muss vor `exp` liegen, mit höchstens wenigen Minuten Toleranz für Uhrenabweichungen, und nach `nbf`, falls dieser Claim vorhanden ist. Erst dann handelt der Empfänger aufgrund der Claims.

Ein JWT-Zugriffstoken nach RFC 9068 zeigt, was ein konkretes Profil ergänzt. Sein Header lautet dort `{"typ":"at+JWT","alg":"RS256","kid":"RjEwOwOA"}`: ein expliziter Typ, ein asymmetrischer Algorithmus und die Kennung des Schlüssels im veröffentlichten Schlüsselsatz des Ausstellers. Das Profil schreibt `iss`, `exp`, `aud`, `sub`, `client_id`, `iat` und `jti` vor. Ein Ressourcenserver muss prüfen, ob `typ` den Wert `at+jwt` oder `application/at+jwt` hat, und andere Typen zurückweisen. Bei diesen Medientyp- und Subtypnamen wird Groß- und Kleinschreibung nicht unterschieden. `at+JWT` im Beispiel ist daher eine gleichwertige Schreibweise.

Die Claims im Token sind eine Momentaufnahme vom Zeitpunkt der Signatur. Alles, was sich vor `exp` ändern kann, wird daher besser auf dem Server entschieden. Eines der Projekte dieser Website, die Absicherung einer

Anwendung auf jeder Ebene, prüft die Berechtigungen, also Berechtigungsstufe, Rolle und Geltungsbereich für Organisationen und Schiffe, bei jeder Anfrage erneut auf dem Server. Die Zugangsschranken im Browser gelten ausschließlich als Bedienkomfort.

Wo es schiefgeht

Das Token wählt seinen eigenen Algorithmus. Der Header `alg` dient der Flexibilität. RFC 8725 beschreibt, wie einige Bibliotheken ihm zu weit vertrauten. Ein Angreifer konnte ihn auf `none` ändern, und die Bibliothek akzeptierte das Token ganz ohne Signatur. Oder er änderte `RS256` auf `HS256`, worauf die Bibliothek einen HMAC mit dem öffentlichen RSA-Schlüssel des Ausstellers als gemeinsamem Geheimnis prüfte: einem Schlüssel, den der Angreifer herunterladen kann. Konfigurieren Sie die Bibliothek mit den akzeptierten Algorithmen, verwenden Sie jeden Schlüssel mit genau einem Algorithmus und weisen Sie `none` grundsätzlich zurück.

Das HMAC-Geheimnis ist ein Passwort. Ein mit `HS256` und einem kurzen, merkbaren Geheimnis signiertes Token lässt sich offline angreifen. Jeder Besitzer eines einzigen Tokens kann in seinem eigenen Tempo Schlüsselkandidaten ausprobieren, ohne Ihren Server zu kontaktieren. RFC 8725 verbietet, ein für Menschen merkbares Passwort direkt als HMAC-Schlüssel zu verwenden. RFC 7518 verlangt für `HS256` mindestens 256 Bits Schlüssellänge. Erzeugen Sie den Schlüssel aus einer kryptografischen Zufallsquelle, bewahren Sie ihn in einem Speicher für Geheimnisse auf und bevorzugen Sie ein Schlüsselpaar, wenn viele Dienste nur verifizieren müssen.

Die Signatur stimmt, also wird alles geglaubt. Eine gültige Signatur beweist, wer das Token ausgestellt hat, nicht, für wen es bestimmt war. Bedient ein Aussteller mehrere Dienste und überspringt der Empfänger die Prüfung von `aud`, kann ein rechtmäßig an einen Dienst gesendetes Token bei einem anderen erneut verwendet werden. RFC 8725 nennt das Substitution und verlangt von jedem Empfänger solcher Token, die Zielgruppe zu validieren und Token ohne Zielgruppe zurückzuweisen. Vergleichen Sie `aud` mit Ihrer eigenen Kennung und prüfen Sie, ob der Signaturschlüssel zum in `iss` genannten Aussteller gehört.

Eine Tokenart wird als eine andere akzeptiert. Ein ID-Token, ein Zugriffstoken und ein Sicherheitsereignistoken desselben Ausstellers können sehr ähnlich aussehen. Wer nur Signatur und Aussteller prüft, akzeptiert möglicherweise die

falsche Art. RFC 8725 nennt dies Cross-JWT Confusion und empfiehlt eine explizite Typangabe im Header `typ`. RFC 9068 setzt das mit `at+jwt` um. Geben Sie jeder Tokenart einen eigenen Typ oder eine eigene Zielgruppe und gestalten Sie die Validierungsregeln so, dass jede Art die Prüfungen der anderen nicht besteht.

Der Header wählt den Schlüssel. Der Header `kid` nennt den zu verwendenden Schlüssel. `jku` und `x5u` können auf eine URL zum Abruf verweisen. Alles stammt vom Sender. Ein ungeprüft in eine Datenbankabfrage oder Verzeichnissuche eingesetztes `kid` wird zum Einfallstor. Ein Server, der jeder in `jku` oder `x5u` angegebenen URL folgt, lässt sich zu Anfragen an beliebige vom Sender gewählte Ziele bringen. Suchen Sie `kid` nur unter bereits vorliegenden Schlüsseln und rufen Sie Schlüssel nur von einer Positivliste erlaubter Orte ab, etwa der veröffentlichten `jwks_uri` des Ausstellers.

Jeder kann die Nutzlast lesen. Ein signiertes JWT schützt seine Claims vor Manipulation, nicht vor dem Lesen. Oft durchläuft es Browser, Protokolle und Proxys. RFC 7519 hält fest, dass ein JWT datenschutzrelevante Informationen enthalten kann und der einfachste Schutz darin besteht, sie wegzulassen. Andernfalls sollte das Token verschlüsselt oder zumindest ausschließlich über TLS gesendet werden. RFC 9068 ergänzt, dass ein Client ein Zugriffstoken überhaupt nicht untersuchen darf, weil der Aussteller sein Format ändern kann. Schreiben Sie Kennungen statt personenbezogener Daten in das Token und verschlüsseln Sie es, wenn der Inhalt selbst vertraulich bleiben muss.

Weiterführende Lektüre

- RFC 7519: JSON Web Token (JWT) — Das Format mit seinen registrierten Claims, dem vollständigen Beispiel und den Validierungsschritten.
- RFC 7515: JSON Web Signature (JWS) — Kodierung und Verbindung von Header, Nutzlast und Signatur sowie der Schlüssel des obigen Beispiels.
- RFC 8725: Bewährte aktuelle Verfahren für JSON Web Tokens — Die bekannten JWT-Angriffe und die Regeln für Bibliotheken und Anwendungen, die sie verhindern.
- RFC 7518: JSON Web Algorithms (JWA) — Die Algorithmusnamen in `alg` mit der Mindestschlüssellänge für jeden HMAC.
- RFC 9068: JWT-Profil für OAuth-2.0-Zugriffstoken — Der Standardaufbau von JWT-Zugriffstoken und die erforderlichen Prüfungen eines Ressourcenservers.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Bereitstellungsweg, die Betriebsrisiken und die Bereitschaftsverantwortlichen. Wir können eine Änderung mit Prüfungen, einem Wiederherstellungsweg und klaren Zuständigkeiten abgrenzen.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist JWT?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

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

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/jwt/