ToiletPaper #76

JSON Web Token

Andreas Scharf / Senior Software Architect / Business Division New Business

✕ Problem

Du möchtest keine Session Cookies verwenden und nicht mit Sticky Session Load Balancing arbeiten.
Du möchtest lieber einen **sicheren, unabhängigen Token** zur Authentifizierung und zum Informationsaustausch erstellen.

✓ Lösung

JSON Web Token (JWT) ist ein offener Standard (RFC 7519) zur sicheren Kommunikation eines JSON-Objekts. In der Regel muss sich der Benutzer erfolgreich anmelden und erhält dann einen Token für API-Calls. Der Token besteht aus drei Teilen: Header, Payload und Signatur. Da die Payload alle erforderlichen Informationen enthält, ist für die Authentifizierung keine Datenbankabfrage erforderlich. Das ist besonders für die Skalierung zustandsloser Backend-Architekturen super!

Ein JWT kann von jedem dekodiert und gelesen werden. Tatsächlich ist das für den Client und das Debuggen nützlich. Die Payload wird nicht verschlüsselt (für Verschlüsselung s. JWE), aber gültige Signaturen können nur erstellt werden, wenn man ein Geheimnis kennt. Jedes Mal, wenn ein Token empfangen wird, muss eine Integritätsprüfung die Signatur bestätigen. So wird sichergestellt, dass der Token nicht manipuliert wurde. Danach wird sein Inhalt als vertrauenswürdig eingestuft.

Die Gültigkeitsdauer des Token kann begrenzt werden, indem zeitbezogene Claims wie *iat* ("Issued At"), *nbf* ("Not Before") und *exp* ("Expiration Time") einbezogen werden. Ein abgelaufener JWT ist zwar weiterhin gültig, da es jedoch nicht möglich ist, Einträge zu manipulieren, ohne die Signatur zu zerstören, wird bei der serverseitigen Integritätsprüfung das Ablaufdatum des Tokens gelesen und der Zugriff verweigert.

Ein Nachteil ist das sehr komplizierte Widerrufen eines Tokens. Eine gängige Lösung ist das Blacklisting. Außerdem helfen kurze Gültigkeitsdauern.

➔ Beispiel

In diesem Beispiel erstellen wir einen von "jambit" ausgegebenen Token zum Thema "jambitee". Die Ablaufzeit wird festgelegt, um die Gültigkeit des Tokens zu begrenzen. Wir haben auch die Benutzerrolle hinzugefügt. Jede Information könnte Teil des Tokens sein, es sollten jedoch so wenig wie möglich enthalten sein, um den Token kompakt zu halten. Die Ausgabe besteht aus drei Base64-codierten Zeichenfolgen, die durch Punkte getrennt sind. In der Praxis verwendet man eine Bibliothek, um einen Token zu erstellen oder seine Claims zu lesen. Diese Bibliotheken sind für viele Programmiersprachen verfügbar.

Header	Payload	Signatur
Der Header enthält den Token-**Typ** und **Algorithmus** der Signatur.	Public Claims wie *iss* (“Issuer”), *sub* (“Subject”) and *exp* (“Expiration Time”) sind im Standard definiert. Zusätzliche Private Claims können im speziellen Anwendungsfall verwendet werden (z. B. **roles**).	Um die Integrität zu prüfen, werden Header- und Payload-Inhalte mit **HMAC** oder via **RSA** mit einem public/private Schlüsselpaar signiert.
{ "alg": "HS256", "typ": "JWT" }	{ "iss": "jambit", "sub": "jambitee", "exp": 1412345678, "roles": ["ROLE_USER"] }	HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), secret)

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJqYW1iaXQiLCJzdWIiOiJqYW1iaXRlZSIsImV4cCI6MTQyMjc3OTYzOCwicm9sZXMiOlsiUk9MRV9VU0VSIl19.PXXM0IdEJlWM1q0xDLXGyhlmsUtvpwfQrQ1uPy2R2m4

✚ Weiterführende Aspekte

- JWT.io – Projektwebseite mit Online-Debuggern und Bibliotheken
- Identity APIs wie Auth0 und Stormpath, die JWT unterstützen
- Verfügbare Bibliotheken: .NET, C, Clojure, Crystal, D, Delphi, Elixir, Go, Haskell, Java, JavaScript, Lua, Node.js, Objective-C, Perl, PHP, Python, Q, Ruby, Rust, Scala, Swift
- Mehr Informationen:

https://tools.ietf.org/html/rfc7519 https://auth0.com/learn/json-web-tokens/
https://medium.com/@berto168/jwt-the-basics-and-more-63e7f1fc43c6
https://developer.atlassian.com/cloud/bitbucket/understanding-jwt-for-apps/