Engineer

JWT

et kompakt, URL-sikkert token med claims i JSON

Hvad det er

Et JWT er en kort tekststreng, der bærer en række udsagn fra én part til en anden i en form, modtageren kan kontrollere. Bogstaverne står for JSON Web Token. RFC 7519 definerede formatet i maj 2015 som en kompakt måde at repræsentere claims på, der kan sendes uændret i en URL, og foreslår, at navnet udtales som det engelske ord »jot«. Et claim er et navn og en værdi, der siges om et subjekt: `"sub": "24400320"` fortæller, hvem tokenet handler om, og `"exp": 1300819380` fortæller, hvornår det holder op med at gælde.

Når det sendes, er et JWT en række base64url-strenge adskilt af punktummer. Et signeret JWT har tre: en header, der angiver algoritmen, en payload med claims og en signatur beregnet over de to første. Et krypteret JWT har fem, og dets claims kan ikke læses uden nøglen. RFC 7519 står ikke selv for kryptografien. Den lader JSON Web Signature (RFC 7515) tage sig af tokens, der signeres eller beskyttes med en MAC (message authentication code), og JSON Web Encryption (RFC 7516) af dem, der krypteres. Algoritmerne er listet i RFC 7518, og nøglerne kan offentliggøres i formatet JSON Web Key fra RFC 7517.

`HS256` er en HMAC med SHA-256: én delt hemmelighed både laver og kontrollerer tokenet, så enhver tjeneste, der kan verificere det, også kan udstede det. `RS256` og `ES256` bruger et nøglepar: udstederen signerer med en privat nøgle, og alle andre verificerer med den offentlige, som frit kan deles ud. RFC 7519 kræver kun, at `HS256` og den usignerede `none` er implementeret, og anbefaler desuden `RS256` og `ES256`.

Afsnit 4.1 registrerer syv claimnavne: `iss` (udsteder), `sub` (subjekt), `aud` (modtager, på engelsk audience), `exp` (udløbstidspunkt), `nbf` (gyldig fra), `iat` (udstedt) og `jti` (et entydigt ID, der kan bruges til at forhindre, at tokenet afspilles igen). Navnene er på tre bogstaver, fordi formatet skal være

kompakt, og de er alle valgfrie. Hvilke claims et token skal indeholde, afgør den applikation, der bruger det. Ud over de registrerede kan enhver definere offentlige claims, enten registreret hos IANA eller med et navn, der ikke kan kollidere med andres, for eksempel en URL, og private claims, som to parter aftaler indbyrdes.

Brug begrebet i en beslutning

Brug »Hvad er JWT?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Konti, adgangskoder og rettigheder sat op, så folk har præcis den adgang, jobbet kræver, og fratrådte mister den samme dag.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har skrevet 1.000 RBAC-regler til Linux-servere — 1,000 adgangsregler skrevet for at lukke et sikkerhedshul

Har skrevet 1.000 RBAC-regler til Linux-servere Kontakt os →

Hvorfor det betyder noget

Pointen med et JWT er, at modtageren kan kontrollere det på egen hånd. Med udstederens nøgle og et ur kan en tjeneste afgøre, om tokenet er ægte, hvem det er til, og om det stadig gælder, uden at slå op i en database og uden at spørge udstederen. RFC 9068 fra oktober 2021 konstaterer, at mange kommercielle OAuth-implementeringer udsteder adgangstokens, som ressourceservere selv parser og validerer, at mønstret er særlig udbredt, hvor den server, der udsteder tokens, og den, der modtager dem, er adskilt af en eller anden grænse, og at mange af dem bruger JWT. Derefter standardiserede RFC'en opbygningen, så den slags tokens fungerer på tværs af leverandører.

JWT er også formatet bag OpenID Connect: hvert ID-token er et JWT stilet til den applikation, der bad om det. Identitetsudbyderen skal signere det, medmindre begge betingelser for at sætte `alg` til `none` er opfyldt: svartypen returnerer ikke noget ID-token fra Authorization Endpoint, som i Authorization Code Flow, og klienten bad udtrykkeligt om `none` ved registreringen. Da medietypen `application/jwt` blev registreret i 2015, nævnte registreringen allerede OpenID Connect, Salesforce, Google, Android, Windows Azure og Amazon Web Services blandt brugerne.

RFC 7519 sigter mod steder med begrænset plads, som HTTP-headeren Authorization og parametre i en URI. Derfor er claimnavnene korte, og derfor bruger kodningen kun tegn, der kan stå i en URL uden at blive escapet.

Den egenskab, der gør et JWT billigt at kontrollere, gør det også svært at trække tilbage. En modtager, der kontrollerer signaturen og uret, accepterer tokenet indtil `exp`, uanset hvad der er sket med kontoen i mellemtiden.

Sådan ser det ud i praksis

RFC 7519 gennemgår et komplet eksempel. Headeren erklærer, at objektet er et JWT, og at det er beskyttet med HMAC SHA-256:

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

Claims angiver en udsteder, et udløbstidspunkt og ét offentligt claim, hvis navn er en URL, så det ikke kan støde sammen med et claim, som andre definerer:

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

Hver del kodes med base64url, HMAC'en beregnes over header og payload sat sammen med et punktum, og de tre resultater sættes sammen til én streng. RFC'en viser den med linjeskift, der kun er der af hensyn til læsbarheden:

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

Tre ting følger af eksemplet. For det første kan alle, der har tokenet, læse dets claims: base64url er en kodning, så den midterste del bliver til JSON igen uden nogen nøgle. For det andet dækker signaturen de kodede strenge præcis, som de blev sendt. Headeren i RFC'en indeholder et CRLF-linjeskift og et indledende mellemrum, og RFC'en angiver de nøjagtige bytes for at undgå tvivl. Formaterer man JSON'en pænere og koder den igen, får man en anden streng, og signaturen passer ikke længere. Nøglen til eksemplet står i appendiks A.1 i RFC 7515, og

med den kan signaturen verificeres. For det tredje er tokenet for længst udløbet: `1300819380` sekunder efter 1. januar 1970 er 22. marts 2011 kl. 18.43 UTC, så en korrekt validator afviser det i dag.

En modtager, der får sådan et token, kører en fast række kontroller, før den læser et eneste claim. Den deler strengen ved punktummerne, afkoder headeren og sikrer sig, at den er gyldig JSON med kun parametre og værdier, hvis syntaks og betydning den forstår og understøtter, eller som ifølge specifikationen udtrykkeligt skal ignoreres, hvis de er ukendte. Den sammenligner `alg` med de algoritmer, den har besluttet at acceptere for netop denne udsteder og nøgle. RFC 7519 er udtrykkelig om, at det er applikationens beslutning, og at et token bør afvises, hvis algoritmen ikke er acceptabel, også selvom signaturen er gyldig. Den verificerer signaturen med en nøgle, der hører til udstederen. Derefter kontrollerer den claims: `iss` skal være en udsteder, den stoler på, `aud` skal nævne modtageren selv, tidspunktet skal ligge før `exp`, med højst få minutters margin for forskelle mellem urene, og efter `nbf`, hvis det claim er med. Først da handler den på indholdet.

Et JWT-adgangstoken efter RFC 9068 viser, hvad en konkret profil lægger til. Headeren i den RFC lyder `{"typ":"at+JWT","alg":"RS256","kid":"RjEwOwOA"}`: en udtrykkelig type, en asymmetrisk algoritme og ID'et på nøglen i udstederens offentliggjorte nøglesæt. Profilen gør `iss`, `exp`, `aud`, `sub`, `client_id`, `iat` og `jti` obligatoriske, og en ressourceserver skal kontrollere, at `typ` er `at+jwt` eller `application/at+jwt`, og afvise andre typer. Der skelnes ikke mellem store og små bogstaver i disse medietype- og undertypenavne; `at+JWT` i eksemplet er derfor en ækvivalent skrivemåde.

Claims i et token er et øjebliksbillede fra det tidspunkt, det blev signeret, så alt, hvad der kan ændre sig før `exp`, er bedre afgjort på serveren. I et af sitets projekter, hærdning af en applikation i hvert lag, kontrolleres rettighederne (tier, rolle, organisation og afgrænsning til skibe) igen på serveren ved hver request, mens kontrollerne i frontenden kun betragtes som UX.

Hvor det går galt

Tokenet vælger selv sin algoritme. Headeren `alg` er der for fleksibilitetens skyld, og RFC 8725 beskriver, hvordan nogle biblioteker stolede for meget på den. En angriber kunne ændre den til `none`, og biblioteket accepterede tokenet helt uden signatur. Eller angriberen ændrede `RS256` til `HS256`, og biblioteket kontrollerede en HMAC med udstederens offentlige RSA-nøgle som delt

hemmelighed, en nøgle, angriberen bare kan hente. Konfigurér biblioteket med de algoritmer, I accepterer, brug hver nøgle med præcis én algoritme, og afvis `none` helt.

HMAC-hemmeligheden er en adgangskode. Et token signeret med `HS256` under en kort hemmelighed, der er til at huske, kan angribes offline: den, der har ét eneste token, kan prøve gæt mod det i sit eget tempo uden at røre jeres server. RFC 8725 forbyder at bruge en adgangskode, et menneske kan huske, direkte som HMAC-nøgle, og RFC 7518 kræver en nøgle på mindst 256 bit til `HS256`. Generér nøglen med en kryptografisk tilfældighedskilde, opbevar den i et hemmelighedslager, og vælg et nøglepar, når mange tjenester kun skal verificere.

Signaturen holder, så alt andet bliver troet. En gyldig signatur viser, hvem der udstedte tokenet, ikke hvem det var til. Hvis én udsteder betjener flere tjenester, og modtageren springer `aud` over, kan et token, der lovligt blev sendt til én tjeneste, genbruges mod en anden. RFC 8725 kalder det substitution og kræver, at modtagere af den slags tokens validerer audience og afviser et token uden. Kontrollér `aud` mod jeres egen identifikator, og kontrollér, at signeringsnøglen hører til den udsteder, der står i `iss`.

Én slags token går for en anden. Et ID-token, et adgangstoken og et security event token fra samme udsteder kan ligne hinanden meget, og en modtager, der kun kontrollerer signatur og udsteder, kan acceptere den forkerte slags. RFC 8725 kalder det cross-JWT confusion og anbefaler en udtrykkelig type i headeren `typ`. RFC 9068 bruger netop det med `at+jwt`. Giv hver slags token sin egen type eller audience, og skriv valideringsreglerne, så hver slags falder igennem i de andres kontroller.

Headeren vælger nøglen. Headeren `kid` angiver, hvilken nøgle der skal bruges, og `jku` og `x5u` kan pege på en URL, nøglen skal hentes fra. De kommer alle fra afsenderen. Et `kid`, der sendes ukontrolleret videre til en databaseforespørgsel eller et opslag i en katalogtjeneste, bliver en vej ind i det system, og en server, der følger enhver URL i `jku` eller `x5u`, kan afsenderen få til at sende forespørgsler hvorhen som helst. Slå kun `kid` op blandt nøgler, I allerede har, og hent kun nøgler fra en godkendt liste af adresser, for eksempel udstederens offentliggjorte `jwks_uri`.

Alle kan læse payloaden. Et signeret JWT beskytter sine claims mod ændringer, ikke mod at blive læst, og det kommer ofte forbi browsere, logfiler og proxyer. RFC 7519 bemærker, at et JWT kan indeholde personfølsomme oplysninger, og at den enkleste beskyttelse er at lade dem blive ude. Ellers bør tokenet krypteres eller i det mindste kun sendes over TLS. RFC 9068 tilføjer, at en klient slet ikke må se ind i et adgangstoken, fordi udstederen kan skifte format når som helst. Læg identifikatorer i tokenet, ikke personoplysninger, og kryptér, når selve indholdet skal holdes privat.

Læs mere

- RFC 7519: JSON Web Token (JWT) — selve formatet med de registrerede claims, det gennemgåede eksempel og valideringstrinnene.
- RFC 7515: JSON Web Signature (JWS) — hvordan header, payload og signatur kodes og sættes sammen, og nøglen til eksemplet ovenfor.
- RFC 8725: JSON Web Token Best Current Practices — de kendte angreb på JWT og reglerne for biblioteker og applikationer, der forhindrer dem.
- RFC 7518: JSON Web Algorithms (JWA) — algoritmenavnene i `alg` med mindste nøglestørrelse for hver HMAC.
- RFC 9068: JWT Profile for OAuth 2.0 Access Tokens — standardopbygningen af JWT-adgangstokens og de kontroller, en ressourceserver skal køre.

Ordliste

Vælg ud fra jeres faktiske vilkår

Medbring udrulningsforløbet, driftsrisiciene og dem, der har vagten. Vi kan afgrænse en ændring med kontroller, en vej tilbage og tydeligt ansvar.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er JWT?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

welcome@engineer.company +45 50 10 61 84 København, Danmark

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/jwt/