Engineer

JWT

et kompakt, URL-sikkert token med claims i JSON

Hva det er

Et JWT er en kort tekststreng som bærer et sett med utsagn fra én part til en annen i en form mottakeren kan kontrollere. Bokstavene står for JSON Web Token. RFC 7519 definerte formatet i mai 2015 som en kompakt måte å representere claims på som kan sendes uendret i en URL, og foreslår at navnet uttales som det engelske ordet «jot». Et claim er et navn og en verdi som sies om et subjekt: `"sub": "24400320"` forteller hvem tokenet gjelder, og `"exp": 1300819380` forteller når det slutter å gjelde.

Når det sendes, er et JWT en rekke base64url-strenger skilt med punktum. Et signert JWT har tre: en header som angir algoritmen, en payload med claimene og en signatur beregnet over de to første. Et kryptert JWT har fem, og claimene kan ikke leses uten nøkkelen. RFC 7519 står ikke selv for kryptografien. Den overlater tokener som signeres eller beskyttes med en MAC (message authentication code), til JSON Web Signature (RFC 7515), og tokener som krypteres, til JSON Web Encryption (RFC 7516). Algoritmene er listet i RFC 7518, og nøklene kan publiseres i formatet JSON Web Key fra RFC 7517.

`HS256` er en HMAC med SHA-256: én delt hemmelighet både lager og kontrollerer tokenet, så enhver tjeneste som kan verifisere det, også kan utstede det. `RS256` og `ES256` bruker et nøkkelpar: utstederen signerer med en privat nøkkel, og alle andre verifiserer med den offentlige, som fritt kan deles ut. RFC 7519 krever bare at `HS256` og den usignerte `none` er implementert, og anbefaler i tillegg `RS256` og `ES256`.

Avsnitt 4.1 registrerer sju claimnavn: `iss` (utsteder), `sub` (subjekt), `aud` (mottaker, på engelsk audience), `exp` (utløpstidspunkt), `nbf` (gyldig fra), `iat` (utstedt) og `jti` (en unik ID som kan brukes til å hindre at tokenet spilles av på nytt). Navnene har tre bokstaver fordi formatet skal være kompakt, og alle er

valgfrie. Hvilke claims et token må inneholde, bestemmer applikasjonen som bruker det. Utover de registrerte kan hvem som helst definere offentlige claims, enten registrert hos IANA eller med et navn som ikke kan kollidere med andres, for eksempel en URL, og private claims som to parter avtaler seg imellom.

Bruk begrepet i en beslutning

Bruk «Hva er JWT?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Kontoer, passord og tilganger ordnet slik at folk har nøyaktig den tilgangen jobben krever, og de som slutter, mister den samme dag.

Se tjenestene våre Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. Skrev 1 000 RBAC-regler for å sikre Linux-servere — 1,000 tilgangsregler skrevet for å tette et sikkerhetshull

Skrev 1 000 RBAC-regler for å sikre Linux-servere Ta kontakt →

Hvorfor det er viktig

Poenget med et JWT er at mottakeren kan kontrollere det på egen hånd. Med utstederens nøkkel og en klokke kan en tjeneste avgjøre om tokenet er ekte, hvem det er til, og om det fortsatt gjelder, uten å slå opp i en database og uten å spørre utstederen. RFC 9068 fra oktober 2021 slår fast at mange kommersielle OAuth-implementasjoner utsteder tilgangstokener som ressursservere selv parser og validerer, at mønsteret er særlig vanlig der serveren som utsteder tokener, og den som tar imot dem, er skilt av en eller annen grense, og at mange av dem bruker JWT. Deretter standardiserte RFC-en oppbygningen, slik at slike tokener fungerer på tvers av leverandører.

JWT er også formatet bak OpenID Connect: hvert ID-token er et JWT adressert til applikasjonen som ba om det. Identitetsleverandøren må signere det med mindre begge vilkårene for å sette `alg` til `none` er oppfylt: responstypen returnerer ikke noe ID-token fra Authorization Endpoint, som i Authorization Code Flow, og klienten ba uttrykkelig om `none` ved registreringen. Da medietypen `application/jwt` ble registrert i 2015, nevnte registreringen allerede OpenID Connect, Salesforce, Google, Android, Windows Azure og Amazon Web Services blant brukerne.

RFC 7519 sikter mot steder med begrenset plass, som HTTP-headeren Authorization og parametere i en URI. Derfor er claimnavnene korte, og derfor bruker kodingen bare tegn som kan stå i en URL uten å måtte kodes om.

Den egenskapen som gjør et JWT billig å kontrollere, gjør det også vanskelig å trekke tilbake. En mottaker som kontrollerer signaturen og klokken, godtar tokenet fram til `exp`, uansett hva som har skjedd med kontoen i mellomtiden.

Slik ser det ut i praksis

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

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

Claimene angir en utsteder, et utløpstidspunkt og ett offentlig claim med en URL som navn, slik at det ikke kan kollidere med et claim som noen andre definerer:

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

Hver del kodes med base64url, HMAC-en beregnes over header og payload satt sammen med et punktum, og de tre resultatene settes sammen til én streng. RFC-en viser den med linjeskift som bare er der for lesbarhetens skyld:

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

Tre ting følger av eksemplet. For det første kan alle som har tokenet, lese claimene: base64url er en koding, så den midterste delen blir til JSON igjen uten noen nøkkel. For det andre dekker signaturen de kodede strengene nøyaktig slik de ble sendt. Headeren i RFC-en inneholder et CRLF-linjeskift og et innledende mellomrom, og RFC-en lister opp de nøyaktige bytene for å unngå tvetydighet. Formaterer man JSON-en penere og koder den på nytt, får man en annen streng, og signaturen stemmer ikke lenger. Nøkkelen til eksemplet står i vedlegg A.1 i

RFC 7515, og med den lar signaturen seg verifisere. For det tredje er tokenet for lengst utløpt: `1300819380` sekunder etter 1. januar 1970 er 22. mars 2011 kl. 18.43 UTC, så en korrekt validator avviser det i dag.

En mottaker som får et slikt token, kjører en fast rekke kontroller før den leser et eneste claim. Den deler strengen ved punktumene, dekoder headeren og forsikrer seg om at den er gyldig JSON med bare parametere og verdier der den forstår og støtter både syntaksen og betydningen, eller som ifølge spesifikasjonen uttrykkelig skal ignoreres når de er ukjente. Den sammenligner `alg` med algoritmene den har bestemt seg for å godta for akkurat denne utstederen og nøkkelen. RFC 7519 sier uttrykkelig at dette er applikasjonens beslutning, og at et token bør avvises hvis algoritmen ikke er akseptabel, selv om signaturen er gyldig. Den verifiserer signaturen med en nøkkel som tilhører utstederen. Deretter kontrollerer den claimene: `iss` må være en utsteder den stoler på, `aud` må nevne mottakeren selv, og det nåværende tidspunktet må være før `exp` (med høyst noen få minutters slingringsmonn for klokkeavvik) og etter `nbf`, hvis det claimet er med. Først da handler den ut fra innholdet.

Et JWT-tilgangstoken etter RFC 9068 viser hva en konkret profil legger til. Headeren i den RFC-en lyder `{"typ":"at+JWT","alg":"RS256","kid":"RjEwOwOA"}`: en uttrykkelig type, en asymmetrisk algoritme og ID-en til nøkkelen i utstederens publiserte nøkkelsett. Profilen gjør `iss`, `exp`, `aud`, `sub`, `client_id`, `iat` og `jti` obligatoriske, og en ressursserver skal kontrollere at `typ` er `at+jwt` eller `application/at+jwt`, og avvise andre typer. Det skilles ikke mellom store og små bokstaver i disse navnene på medietyper og undertyper; `at+JWT` i eksemplet er derfor en likeverdig skrivemåte.

Claimene i et token er et øyeblikksbilde fra da det ble signert, så alt som kan endre seg før `exp`, avgjøres best på serveren. I ett av prosjektene her på nettstedet, herding av en applikasjon i hvert lag, kontrolleres rettighetene (abonnementsnivå, rolle, organisasjon og avgrensning per skip) på nytt på serveren ved hver forespørsel, mens sperrene i frontenden bare behandles som UX.

Hvor det går galt

Tokenet velger sin egen algoritme. Headeren `alg` er der for fleksibilitetens skyld, og RFC 8725 beskriver hvordan noen biblioteker stolte for mye på den. En angriper kunne endre den til `none`, og biblioteket godtok tokenet helt uten signatur. Eller angriperen endret `RS256` til `HS256`, og biblioteket kontrollerte en

HMAC med utstederens offentlige RSA-nøkkel som delt hemmelighet, en nøkkel angriperen uten videre kan laste ned. Konfigurer biblioteket med algoritmene dere godtar, bruk hver nøkkel med nøyaktig én algoritme, og avvis `none` uten unntak.

HMAC-hemmeligheten er et passord. Et token signert med `HS256` under en kort hemmelighet som er lett å huske, kan angripes offline: den som har ett eneste token, kan prøve gjetninger mot det i sitt eget tempo uten å berøre serveren deres. RFC 8725 forbyr å bruke et passord som et menneske kan huske, direkte som HMAC-nøkkel, og RFC 7518 krever en nøkkel på minst 256 bit for `HS256`. Generer nøkkelen med en kryptografisk tilfeldighetskilde, oppbevar den i et hemmelighetslager, og velg et nøkkelpar når mange tjenester bare skal verifisere.

Signaturen holder, så alt annet blir trodd. En gyldig signatur viser hvem som utstedte tokenet, ikke hvem det var til. Hvis én utsteder betjener flere tjenester og mottakeren hopper over `aud`, kan et token som ble sendt lovlig til én tjeneste, gjenbrukes mot en annen. RFC 8725 kaller dette substitusjon og krever at mottakere av slike tokener validerer audience og avviser et token som mangler det. Kontroller `aud` mot deres egen identifikator, og kontroller at signeringsnøkkelen tilhører utstederen som står i `iss`.

Én slags token går for en annen. Et ID-token, et tilgangstoken og et security event token fra samme utsteder kan ligne mye på hverandre, og en mottaker som bare kontrollerer signatur og utsteder, kan godta feil slags. RFC 8725 kaller dette cross-JWT confusion og anbefaler en uttrykkelig type i headeren `typ`. RFC 9068 gjør nettopp det med `at+jwt`. Gi hver slags token sin egen type eller audience, og skriv valideringsreglene slik at hver slags stopper i de andres kontroller.

Headeren velger nøkkelen. Headeren `kid` angir hvilken nøkkel som skal brukes, og `jku` og `x5u` kan peke på en URL nøkkelen skal hentes fra. Alle kommer fra avsenderen. En `kid`-verdi som sendes ukontrollert videre til en databasespørring eller et oppslag i en katalogtjeneste, blir en vei inn i det systemet, og en server som følger enhver URL i `jku` eller `x5u`, kan avsenderen få til å sende forespørsler hvor som helst. Slå bare opp `kid` blant nøkler dere allerede har, og hent bare nøkler fra en godkjent liste over adresser, for eksempel utstederens publiserte `jwks_uri`.

Alle kan lese payloaden. Et signert JWT beskytter claimene sine mot endringer, ikke mot å bli lest, og det passerer ofte nettlesere, logger og proxyer. RFC 7519 påpeker at et JWT kan inneholde personvernsensitive opplysninger, og at den enkleste beskyttelsen er å utelate dem. Ellers bør tokenet krypteres eller i det minste bare sendes over TLS. RFC 9068 legger til at en klient overhodet ikke må se inn i et tilgangstoken, fordi utstederen når som helst kan endre formatet. Legg identifikatorer i tokenet, ikke personopplysninger, og krypter når selve innholdet skal holdes privat.

Les mer

- RFC 7519: JSON Web Token (JWT) — selve formatet med de registrerte claimene, det gjennomgåtte eksemplet og valideringsstegene.
- RFC 7515: JSON Web Signature (JWS) — hvordan header, payload og signatur kodes og settes sammen, og nøkkelen til eksemplet ovenfor.
- RFC 8725: JSON Web Token Best Current Practices — de kjente angrepene på JWT og reglene for biblioteker og applikasjoner som forhindrer dem.
- RFC 7518: JSON Web Algorithms (JWA) — algoritmenavnene i `alg` med minste nøkkelstørrelse for hver HMAC.
- RFC 9068: JWT Profile for OAuth 2.0 Access Tokens — standardoppbygningen av JWT-tilgangstokener og kontrollene en ressursserver skal kjøre.

Ordliste

Velg ut fra deres faktiske rammer

Ta med utrullingsløpet, driftsrisikoene og dem som har vakt. Vi kan avgrense en endring med kontroller, en vei tilbake og tydelig ansvar.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er JWT?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

Se tjenestene våre

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

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/glossary/jwt/