Engineer

JWT

en kompakt, URL-säker token med claims i JSON

Vad det är

En JWT är en kort teckensträng som bär en uppsättning påståenden från en part till en annan i en form som mottagaren kan kontrollera. Bokstäverna står för JSON Web Token. RFC 7519 definierade formatet i maj 2015 som ett kompakt sätt att representera claims som kan skickas oförändrat i en URL, och föreslår att namnet uttalas som det engelska ordet ”jot”. Ett claim är ett namn och ett värde som påstås om ett subjekt: `"sub": "24400320"` säger vem den handlar om, och `"exp": 1300819380` när den slutar gälla.

När den skickas är en JWT en rad base64url-strängar åtskilda av punkter. En signerad JWT har tre: en header som anger algoritmen, en payload med claims och en signatur beräknad över de två första. En krypterad JWT har fem, och dess claims går inte att läsa utan nyckeln. RFC 7519 sköter inte själv kryptografin. Den överlåter tokens som signeras eller skyddas med en MAC (message authentication code) åt JSON Web Signature (RFC 7515), och tokens som krypteras åt JSON Web Encryption (RFC 7516). Algoritmerna listas i RFC 7518, och nycklarna kan publiceras i formatet JSON Web Key från RFC 7517.

`HS256` är en HMAC med SHA-256: en och samma delade hemlighet både skapar och kontrollerar token, så varje tjänst som kan verifiera den också kan utfärda en egen. `RS256` och `ES256` använder ett nyckelpar: utfärdaren signerar med en privat nyckel, och alla andra verifierar med den publika, som kan delas ut fritt. RFC 7519 kräver bara att `HS256` och den osignerade `none` implementeras, och rekommenderar dessutom `RS256` och `ES256`.

Avsnitt 4.1 registrerar sju claimnamn: `iss` (utfärdare), `sub` (subjekt), `aud` (mottagare, på engelska audience), `exp` (utgångstid), `nbf` (giltig från), `iat` (utfärdad) och `jti` (ett unikt ID som kan användas för att hindra att token spelas upp igen). Namnen har tre bokstäver eftersom formatet ska vara kompakt,

och alla är valfria. Vilka claims en token måste innehålla bestämmer den applikation som använder den. Utöver de registrerade kan vem som helst definiera publika claims, antingen registrerade hos IANA eller med ett namn som inte kan krocka med någon annans, till exempel en URL, och privata claims som två parter kommer överens om sinsemellan.

Använd begreppet i ett beslut

Använd ”Vad är JWT?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Konton, lösenord och behörigheter ordnade så att alla har exakt den åtkomst deras arbete kräver, och de som slutar förlorar den samma dag.

Se våra tjänster Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. Skrev 1 000 RBAC-regler som säkrar Linux-servrar — 1,000 åtkomstregler, skrivna för att täppa till en säkerhetslucka

Skrev 1 000 RBAC-regler som säkrar Linux-servrar Hör av er →

Varför det spelar roll

Poängen med en JWT är att mottagaren kan kontrollera den på egen hand. Med utfärdarens nyckel och en klocka kan en tjänst avgöra om token är äkta, vem den är avsedd för och om den fortfarande gäller, utan att slå upp något i en databas och utan att fråga utfärdaren. RFC 9068 från oktober 2021 konstaterar att många kommersiella OAuth-implementationer utfärdar åtkomsttokens som resursservrar själva tolkar och validerar, att mönstret är särskilt vanligt där servern som utfärdar tokens och den som tar emot dem skiljs åt av någon form av gräns, och att många av dem använder JWT. Därefter standardiserade RFC:n uppbyggnaden, så att sådana tokens fungerar mellan olika leverantörer.

JWT är också formatet bakom OpenID Connect: varje ID-token är en JWT adresserad till den applikation som bad om den. Identitetsleverantören måste signera den om inte båda villkoren för att använda `alg` med värdet `none` är uppfyllda: svarstypen returnerar ingen ID-token från Authorization Endpoint, som i Authorization Code Flow, och klienten begärde uttryckligen `none` vid registreringen. När medietypen `application/jwt` registrerades 2015 nämnde registreringen redan OpenID Connect, Salesforce, Google, Android, Windows Azure och Amazon Web Services bland användarna.

RFC 7519 riktar in formatet på ställen med begränsat utrymme, som HTTP-headern Authorization och parametrar i en URI. Därför är claimnamnen korta, och därför använder kodningen bara tecken som kan stå i en URL utan att behöva kodas om.

Samma egenskap som gör en JWT billig att kontrollera gör den svår att dra tillbaka. En mottagare som kontrollerar signaturen och klockan godtar token fram till `exp`, oavsett vad som har hänt med kontot under tiden.

Så ser det ut i praktiken

RFC 7519 går igenom ett komplett exempel. Headern deklarerar att objektet är en JWT och att det är skyddat med HMAC SHA-256:

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

Claims anger en utfärdare, en utgångstid och ett publikt claim vars namn är en URL, så att det inte kan krocka med ett claim som någon annan definierar:

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

Varje del kodas med base64url, HMAC:en beräknas över header och payload sammanfogade med en punkt, och de tre resultaten fogas ihop till en enda sträng. RFC:n visar den med radbrytningar som bara finns där för läsbarhetens skull:

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

Tre saker följer av exemplet. För det första kan alla som har token läsa dess claims: base64url är en kodning, så den mittersta delen blir JSON igen utan någon nyckel. För det andra täcker signaturen de kodade strängarna exakt som de skickades. Headern i RFC:n innehåller en CRLF-radbrytning och ett inledande mellanslag, och RFC:n anger exakt vilka byte det rör sig om för att undanröja tvetydigheter. Formaterar man JSON-texten snyggare och kodar den igen blir strängen en annan, och signaturen stämmer inte längre. Nyckeln till exemplet

står i bilaga A.1 i RFC 7515, och med den går signaturen att verifiera. För det tredje har token för länge sedan gått ut: `1300819380` sekunder efter den 1 januari 1970 är den 22 mars 2011 kl. 18.43 UTC, så en korrekt validerare avvisar den i dag.

En mottagare som får en sådan token går igenom en fast rad kontroller innan den läser ett enda claim. Den delar strängen vid punkterna, avkodar headern och försäkrar sig om att den är giltig JSON med enbart parametrar och värden vars syntax och innebörd den förstår och stöder, eller som enligt specifikationen uttryckligen ska ignoreras om de är okända. Den jämför `alg` med de algoritmer den har bestämt sig för att godta för just den här utfärdaren och nyckeln. RFC 7519 säger uttryckligen att det är applikationens beslut, och att en token bör avvisas om algoritmen inte är godtagbar, även när signaturen är giltig. Den verifierar signaturen med en nyckel som tillhör utfärdaren. Sedan kontrollerar den claims: `iss` måste vara en utfärdare den litar på, `aud` måste ange mottagaren själv, och aktuell tid måste ligga före `exp` (med högst några minuters marginal för avvikande klockor) och efter `nbf` om det claimet finns. Först då agerar den på innehållet.

En JWT-åtkomsttoken enligt RFC 9068 visar vad en konkret profil lägger till. Headern i den RFC:n lyder `{"typ":"at+JWT","alg":"RS256","kid":"RjEwOwOA"}` : en uttrycklig typ, en asymmetrisk algoritm och ID:t för nyckeln i utfärdarens publicerade nyckeluppsättning. Profilen gör `iss` , `exp` , `aud` , `sub` , `client_id` , `iat` och `jti` obligatoriska, och en resursserver ska kontrollera att `typ` är `at+jwt` eller `application/at+jwt` och avvisa andra typer. Dessa namn på medietyper och undertyper är skiftlägesokänsliga; `at+JWT` i exemplet är därför ett likvärdigt skrivsätt.

Claims i en token är en ögonblicksbild från när den signerades, så allt som kan ändras före `exp` avgörs bäst på servern. I ett av projekten här på webbplatsen, härdning av en applikation i varje lager, kontrolleras rättigheterna (nivå, roll, organisation och avgränsning per fartyg) på nytt på servern vid varje anrop, medan spärrarna i frontenden enbart behandlas som UX.

Var det går fel

Token väljer sin egen algoritm. Headern `alg` finns där för flexibilitetens skull, och RFC 8725 beskriver hur vissa bibliotek litade för mycket på den. En angripare kunde ändra den till `none` , och biblioteket godtog token helt utan signatur. Eller så ändrade angriparen `RS256` till `HS256` , och biblioteket kontrollerade en HMAC

med utfärdarens publika RSA-nyckel som delad hemlighet, en nyckel som angriparen helt enkelt kan ladda ner. Konfigurera biblioteket med de algoritmer ni godtar, använd varje nyckel med exakt en algoritm, och avvisa `none` utan undantag.

HMAC-hemligheten är ett lösenord. En token som signerats med `HS256` under en kort hemlighet som är lätt att komma ihåg kan angripas offline: den som har en enda token kan pröva gissningar mot den i sin egen takt utan att röra er server. RFC 8725 förbjuder att ett lösenord som en människa kan komma ihåg används direkt som HMAC-nyckel, och RFC 7518 kräver en nyckel på minst 256 bitar för `HS256`. Generera nyckeln med en kryptografisk slumpkälla, förvara den i ett hemlighetslager, och välj ett nyckelpar när många tjänster bara behöver verifiera.

Signaturen stämmer, så allt annat godtas. En giltig signatur visar vem som utfärdade token, inte vem den var avsedd för. Om en utfärdare betjänar flera tjänster och mottagaren hoppar över `aud`, kan en token som legitimt skickats till en tjänst återanvändas mot en annan. RFC 8725 kallar det substitution och kräver att mottagare av sådana tokens validerar audience och avvisar en token som saknar det. Kontrollera `aud` mot er egen identifierare, och kontrollera att signeringsnyckeln tillhör den utfärdare som anges i `iss`.

En sorts token tas för en annan. En ID-token, en åtkomsttoken och en security event token från samma utfärdare kan likna varandra mycket, och en mottagare som bara kontrollerar signatur och utfärdare kan godta fel sort. RFC 8725 kallar det cross-JWT confusion och rekommenderar en uttrycklig typ i headern `typ`. RFC 9068 tillämpar just det med `at+jwt`. Ge varje sorts token en egen typ eller audience, och skriv valideringsreglerna så att varje sort fastnar i de andras kontroller.

Headern väljer nyckeln. Headern `kid` anger vilken nyckel som ska användas, och `jku` och `x5u` kan peka på en URL som nyckeln ska hämtas från. Alla kommer från avsändaren. Ett `kid`-värde som skickas okontrollerat vidare till en databasfråga eller en uppslagning i en katalogtjänst blir en väg in i det systemet, och en server som följer vilken URL som helst i `jku` eller `x5u` kan avsändaren få att skicka anrop vart som helst. Slå bara upp `kid` bland nycklar ni redan har, och hämta bara nycklar från en godkänd lista med adresser, till exempel utfärdarens publicerade `jwks_uri`.

Vem som helst kan läsa payloaden. En signerad JWT skyddar sina claims mot ändringar, inte mot att läsas, och den passerar ofta webbläsare, loggar och proxyservrar. RFC 7519 påpekar att en JWT kan innehålla integritetskänslig information och att det enklaste skyddet är att utelämna den. Annars bör token krypteras eller åtminstone bara skickas över TLS. RFC 9068 lägger till att en klient inte alls får titta in i en åtkomsttoken, eftersom utfärdaren när som helst kan byta format. Lägg identifierare i token, inte personuppgifter, och kryptera när själva innehållet måste hållas privat.

Läs mer

- RFC 7519: JSON Web Token (JWT) — själva formatet med de registrerade claimnamnen, det genomgångna exemplet och valideringsstegen.
- RFC 7515: JSON Web Signature (JWS) — hur header, payload och signatur kodas och fogas ihop, och nyckeln till exemplet ovan.
- RFC 8725: JSON Web Token Best Current Practices — de kända attackerna mot JWT och reglerna för bibliotek och applikationer som förhindrar dem.
- RFC 7518: JSON Web Algorithms (JWA) — algoritmnamnen i `alg` med minsta nyckelstorlek för varje HMAC.
- RFC 9068: JWT Profile for OAuth 2.0 Access Tokens — standardupplägget för åtkomsttokens i JWT-format och de kontroller en resursserver ska göra.

Ordlista

Välj utifrån era faktiska förutsättningar

Ta med driftsättningsvägen, driftsriskerna och dem som har jour. Vi kan avgränsa en förändring med kontroller, en väg tillbaka och tydligt ansvar.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är JWT?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

Se våra tjänster

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

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

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

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://engineer.company/sv/glossary/jwt/