Engineer

JWT

Token compacto y apto para URL con declaraciones JSON

Qué es

Un JWT es una cadena corta que transporta un conjunto de declaraciones de una parte a otra de una forma que el receptor puede comprobar. Las siglas significan JSON Web Token. RFC 7519 lo definió en mayo de 2015 como una representación compacta de declaraciones apta para incluirse en una URL, y señala que el nombre se pronuncia «jot». Una declaración es un nombre y un valor que se afirman sobre un sujeto: `"sub": "24400320"` indica de quién trata el token y `"exp": 1300819380` cuándo deja de ser válido.

En la transmisión, un JWT es una serie de cadenas base64url unidas por puntos. Uno firmado tiene tres: una cabecera que identifica el algoritmo, una carga útil que contiene las declaraciones y una firma calculada sobre las dos primeras. Uno cifrado tiene cinco y sus declaraciones no pueden leerse sin la clave. RFC 7519 no realiza por sí mismo ninguna operación criptográfica. Se apoya en JSON Web Signature (RFC 7515) para los tokens firmados o protegidos mediante un código de autenticación de mensajes, y en JSON Web Encryption (RFC 7516) para los cifrados; los algoritmos se enumeran en RFC 7518 y las claves pueden publicarse en el formato JSON Web Key de RFC 7517.

`HS256` es un HMAC con SHA-256: se emplea un mismo secreto compartido para crear y comprobar el token, por lo que todo servicio capaz de verificarlo también puede emitirlo. `RS256` y `ES256` utilizan un par de claves: el emisor firma con una clave privada y los demás verifican con la pública, que puede distribuirse libremente. RFC 7519 solo exige implementar `HS256` y la opción sin firma `none`, y recomienda `RS256` y `ES256`.

La sección 4.1 registra siete nombres de declaraciones: `iss` (emisor), `sub` (sujeto), `aud` (destinatario), `exp` (momento de caducidad), `nbf` (no válido antes de), `iat` (momento de emisión) y `jti` (identificador JWT, único, que

puede impedir la repetición de un token). Tienen tres letras porque el formato pretende ser compacto y todos son opcionales. La aplicación que lo utiliza decide qué declaraciones debe contener un token. Además de las registradas, cualquiera puede definir declaraciones públicas, registradas en IANA o con un nombre que no pueda colisionar con el de nadie más, como una URL, o declaraciones privadas acordadas entre dos partes.

Utilice el concepto en una decisión

Aplique «¿Qué es un JWT?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Cuentas, contraseñas y permisos organizados para que cada persona tenga exactamente el acceso que necesita y lo pierda el mismo día que deja la empresa.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. 1.000 reglas RBAC para servidores Linux — 1,000 reglas de acceso escritas para cerrar una brecha de seguridad

1.000 reglas RBAC para servidores Linux Póngase en contacto →

Por qué importa

La idea de un JWT es que el receptor pueda comprobarlo por sí solo. Con la clave del emisor y un reloj, un servicio puede determinar si el token es auténtico, a quién va dirigido y si sigue siendo válido, sin consultar una base de datos ni llamar al emisor. RFC 9068 (octubre de 2021) recoge que muchas implementaciones comerciales de OAuth emiten tokens de acceso que los servidores de recursos analizan y validan directamente, un patrón habitual cuando existe alguna frontera entre el servidor que emite tokens y el que los acepta, y que muchas utilizan JWT. Después normalizó su estructura para que esos tokens funcionaran entre proveedores.

También es el formato que utiliza OpenID Connect: todo token de identidad es un JWT dirigido a la aplicación que lo solicitó. El proveedor debe firmarlo salvo que se cumplan ambas condiciones para utilizar `alg` con el valor `none` : el tipo de respuesta no devuelve ningún token de identidad desde el punto de conexión de autorización, como ocurre en el flujo de código de autorización, y el cliente solicitó expresamente `none` al registrarse. Cuando se registró el tipo de medio `application/jwt` en 2015, el registro ya mencionaba entre sus usuarios a OpenID Connect, Salesforce, Google, Android, Windows Azure y Amazon Web Services.

RFC 7519 dirige el formato a espacios limitados, como la cabecera Authorization y los parámetros de consulta de una URI; por eso los nombres de las declaraciones son cortos y la codificación pasa por una URL sin escapes.

La misma propiedad que hace barato comprobar un JWT dificulta revocarlo. Un receptor que verifica la firma y el reloj aceptará el token hasta `exp`, independientemente de lo que haya ocurrido con la cuenta mientras tanto.

Cómo se aplica en la práctica

RFC 7519 desarrolla un ejemplo completo. La cabecera declara que el objeto es un JWT y que está protegido con HMAC SHA-256:

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

Las declaraciones indican un emisor, un momento de caducidad y una declaración pública cuyo nombre es una URL, para evitar que colisione con otra definida por cualquiera:

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

Cada parte se codifica con base64url, el HMAC se calcula sobre la cabecera y la carga útil unidas por un punto y los tres resultados se unen en una cadena. La RFC la muestra con saltos de línea solo para facilitar su presentación:

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

De este ejemplo se desprenden tres cosas. Primero, cualquiera que posea el token puede leer las declaraciones: base64url es una codificación, así que decodificar la parte central recupera el JSON sin ninguna clave. Segundo, la firma cubre las cadenas codificadas exactamente como se enviaron. La cabecera de la RFC contiene un salto de línea CRLF y un espacio inicial, y por eso enumera los bytes exactos; formatear el JSON de manera más ordenada y codificarlo de nuevo produciría otra cadena y la firma dejaría de coincidir. La clave de este

ejemplo aparece en el apéndice A.1 de RFC 7515 y con ella la firma se verifica. Tercero, el token caducó hace mucho: `1300819380` segundos después del 1 de enero de 1970 corresponden al 22 de marzo de 2011 a las 18:43 UTC, por lo que un validador correcto lo rechaza hoy.

Un receptor que recibe un token así ejecuta un conjunto fijo de comprobaciones antes de leer una sola declaración. Divide la cadena por los puntos, decodifica la cabecera y confirma que sea JSON válido que solo contenga parámetros y valores cuya sintaxis y significado entiende y admite, o que la especificación indique expresamente que deben ignorarse cuando sean desconocidos. Compara `alg` con los algoritmos que ha decidido aceptar para ese emisor y esa clave: RFC 7519 deja claro que es una decisión de la aplicación y que un token debería rechazarse si su algoritmo no es aceptable, aunque la firma sea válida. Verifica la firma con una clave que pertenezca al emisor. Después comprueba las declaraciones: `iss` debe ser un emisor de confianza, `aud` debe identificar al propio receptor, la hora actual debe ser anterior a `exp`, permitiendo como máximo unos pocos minutos de desfase de reloj, y posterior a `nbf` si esa declaración está presente. Solo entonces actúa según las declaraciones.

Un token de acceso JWT conforme a RFC 9068 muestra qué añade un perfil real. Su cabecera en esa RFC es `{"typ":"at+JWT","alg":"RS256","kid":"RjEwOwOA"}`: un tipo explícito, un algoritmo asimétrico y el identificador de la clave dentro del conjunto publicado por el emisor. El perfil hace obligatorios `iss`, `exp`, `aud`, `sub`, `client_id`, `iat` y `jti`; un servidor de recursos debe comprobar que `typ` sea `at+jwt` o `application/at+jwt` y rechazar otros tipos. Estos nombres de tipo y subtipo de medio no distinguen mayúsculas de minúsculas; `at+JWT`, en el ejemplo, es una escritura equivalente.

Las declaraciones de un token son una instantánea del momento de la firma, por lo que todo lo que pueda cambiar antes de `exp` conviene decidirlo en el servidor. Uno de los proyectos de este sitio, reforzar la seguridad de una aplicación en todas sus capas, vuelve a comprobar los permisos —nivel, rol, organización y ámbito por buque— en el servidor en cada petición y trata los controles del navegador únicamente como elementos de la experiencia de usuario.

Dónde surgen los problemas

El token elige su propio algoritmo. La cabecera `alg` aporta flexibilidad, y RFC 8725 describe cómo algunas bibliotecas confiaban demasiado en ella. Un atacante podía cambiarla a `none` y la biblioteca aceptaba el token sin firma alguna; o cambiar `RS256` a `HS256`, y la biblioteca comprobaba un HMAC utilizando como secreto compartido la clave RSA pública del emisor, que el atacante puede descargar. Configure la biblioteca con los algoritmos que acepta, utilice cada clave con exactamente un algoritmo y rechace `none` sin excepciones.

El secreto HMAC es una contraseña. Un token firmado con `HS256` mediante un secreto corto y fácil de recordar puede atacarse sin conexión: quien posea un único token puede probar conjeturas a su ritmo, sin tocar su servidor. RFC 8725 prohíbe utilizar directamente una contraseña memorizable como clave HMAC y RFC 7518 exige para `HS256` una clave de al menos 256 bits. Genere la clave con una fuente de aleatoriedad criptográfica, guárdela en un almacén de secretos y prefiera un par de claves cuando muchos servicios solo necesiten verificar.

La firma es válida y por eso se cree todo. Una firma válida demuestra quién emitió el token, no a quién iba dirigido. Si un emisor atiende a varios servicios y el receptor omite `aud`, un token legítimamente enviado a un servicio puede reutilizarse contra otro. RFC 8725 lo llama sustitución y exige a todo receptor de esos tokens que valide el destinatario y rechace un token que no lo indique. Compruebe `aud` frente a su propio identificador y que la clave de firma pertenezca al emisor indicado en `iss`.

Un tipo de token se acepta como otro. Un token de identidad, uno de acceso y uno de eventos de seguridad del mismo emisor pueden parecerse mucho; un receptor que solo compruebe firma y emisor puede aceptar el tipo equivocado. RFC 8725 lo llama confusión entre JWT y recomienda indicar un tipo explícito mediante la cabecera `typ`; RFC 9068 lo aplica con `at+jwt`. Asigne a cada clase de token su propio tipo o destinatario y escriba las reglas de validación para que cada clase falle las comprobaciones de las demás.

La cabecera elige la clave. La cabecera `kid` indica qué clave utilizar, y `jku` y `x5u` pueden apuntar a una URL de la que obtenerla. Todos llegan del remitente. Un `kid` pasado sin comprobar a una consulta de base de datos o una búsqueda en directorios se convierte en una vía de entrada a ese sistema; un servidor que sigue cualquier URL recibida en `jku` o `x5u` puede ser inducido a enviar

peticiones adonde el remitente elija. Busque `kid` solo entre las claves que ya posee y obtenga claves únicamente de una lista de ubicaciones permitidas, como el `jwks_uri` publicado por el emisor.

Cualquiera puede leer la carga útil. Un JWT firmado protege sus declaraciones frente a modificaciones, no frente a su lectura, y suele pasar por navegadores, registros y proxies. RFC 7519 señala que un JWT puede contener información sensible para la privacidad y que la protección más sencilla es no incluirla; de lo contrario, el token debería cifrarse o, al menos, enviarse solo mediante TLS. RFC 9068 añade que un cliente no debe inspeccionar un token de acceso en absoluto, ya que el emisor puede cambiar su formato. Incluya identificadores en el token, no datos personales, y cifre cuando el propio contenido deba seguir siendo privado.

Lecturas adicionales

- RFC 7519: JSON Web Token (JWT) — el propio formato, con las declaraciones registradas, el ejemplo desarrollado y los pasos de validación.
- RFC 7515: JSON Web Signature (JWS) — cómo se codifican y unen cabecera, carga útil y firma, y la clave utilizada en el ejemplo anterior.
- RFC 8725: prácticas recomendadas actuales para JSON Web Token — ataques conocidos a JWT y reglas para bibliotecas y aplicaciones que los impiden.
- RFC 7518: algoritmos web JSON (JWA) — los nombres de algoritmo utilizados en `alg`, con el tamaño mínimo de clave para cada HMAC.
- RFC 9068: perfil JWT para tokens de acceso OAuth 2.0 — la estructura estándar de los tokens de acceso JWT y las comprobaciones que debe ejecutar un servidor de recursos.

Glosario

Elija según sus restricciones reales

Describa la ruta de despliegue, los riesgos operativos y quiénes se encargan de las guardias. Podemos definir el alcance de un cambio con comprobaciones, una vía de recuperación y responsabilidades claras.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es un JWT?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

welcome@engineer.company +45 50 10 61 84 Copenhague, Dinamarca

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

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

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/jwt/