JSON Web Token (JWT)

¿Qué es un JSON Web Token (JWT)?

Un JSON Web Token (JWT) es un estándar (RFC 7519) abierto utilizado para compartir información de forma segura y compacta entre diferentes partes. Está diseñado para ser legible, seguro y auto-contenido, lo que lo convierte en una opción popular para el manejo de autenticación y autorización en aplicaciones web.

JWT es ampliamente utilizado en aplicaciones web y APIs RESTful porque permite verificar la identidad del usuario sin necesidad de almacenar su estado en el servidor. Los tokens son firmados criptográficamente, lo que asegura que no puedan ser alterados sin invalidarlos.

Por este motivo, los JWT son especialmente populares en los procesos de autentificación. Con este estándar es posible cifrar mensajes cortos, dotarlos de información sobre el remitente y demostrar si este cuenta con los derechos de acceso requeridos. Los propios usuarios solo entran en contacto con el token de manera indirecta: por ejemplo, al introducir el nombre de usuario y la contraseña en una interfaz. La comunicación como tal entre las diferentes aplicaciones se lleva a cabo en el lado del cliente y del servidor.

Componentes de un JWT

Un JWT está compuesto por tres partes principales: el **header** (encabezado), el **payload** (carga útil), y la **signature** (firma). Cada parte es una cadena de texto codificada en base64 y están separadas por un punto (`.`), creando un formato `header.payload.signature`.

1. Header (Encabezado)

El encabezado usualmente contiene dos partes:

- **Tipo de token**, que en este caso es `JWT`.
- **Algoritmo de firma** que se utilizará, como `HS256` (HMAC-SHA256) o `RS256` (RSA-SHA256).

Ejemplo de un encabezado codificado en JSON:

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

Este JSON se codifica en base64, resultando en la primera parte del JWT.

Siempre se recomienda introducir JWT como tipo, que hace referencia al tipo de medio application/jwt de la IANA. En el ejemplo anterior, el header indica que HMAC-SHA256, abreviado como HS256, se utiliza para firmar el token. Otros métodos de cifrado típicos son RSA, con SHA-256 (RS256), y ECDSA, con SHA-256 (ES256). No se recomienda prescindir del cifrado, aunque sí se puede especificar none si los datos no requieren un nivel de protección alto. Los posibles valores están estandarizados por JSON-Web-Encryption según el RFC 7516.

En el caso de los JSON Web Tokens complejos firmados o cifrados, también existe el parámetro cty para content type, que se rellena del mismo modo, con el valor JWT. En el resto de casos, este parámetro se omite.

2. Payload (Carga útil)

El payload contiene las declaraciones (claims), es decir, la información que queremos enviar. Existen tres tipos de declaraciones:

- **Declaraciones registradas**: Son estandarizadas y están definidas en la especificación, como iss (emisor), exp (fecha de expiración), sub (asunto), entre otras.
- **Declaraciones públicas**: Son definidas libremente, aunque deben estar registradas en el IANA JSON Web Token Registry para evitar conflictos.
- **Declaraciones privadas**: Son definidas en una implementación específica, como el id del usuario, rol, entre otros.

Ejemplo de un payload en JSON:

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
```

Todos los claims son opcionales, por lo que no es obligatorio utilizar todos los claims registrados. En general, el payload puede contener un número ilimitado de claims, aunque es aconsejable limitar la información del JWT al mínimo. Cuanto más extenso sea el JWT, más recursos necesitará para la codificación y la descodificación.

Este JSON también se codifica en base64 y se convierte en la segunda parte del JWT.

3. Signature (Firma)

La firma asegura la integridad del token y verifica su autenticidad. Para crear la firma:

1. Se toma el encabezado y el payload codificados.
2. Se usa un algoritmo y una clave secreta para firmarlos.

Ejemplo de creación de la firma:

```
HMACSHA256(
  base64UrlEncode(header) + "." + base64UrlEncode(payload),
  secret
)
```

Esta firma se convierte en la tercera parte del JWT y sirve para confirmar que el mensaje no ha sido alterado.

La estructura viene definida por JSON Web Signature (JWS), un estándar establecido en el RFC 7515. Para que la firma sea eficaz, es necesario utilizar una clave secreta que solo conozca la aplicación original. Por un lado, la firma verifica que el mensaje no se ha modificado por el camino. Por otro, si el token está firmado con una clave privada, también garantiza que el remitente del JWT sea el correcto.

Estructura de un JWT

Un JWT completo tiene el siguiente formato:

Codificado:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0Ijox
NTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

Decodificado: HEADER

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

PAYLOAD:

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

Cada sección (`header`, `payload` y `signature`) está separada por un punto (`.`).

¿Cómo funciona un JSON Web Token?

El inicio de sesión de usuario ejemplifica bien la función del JSON Web Token. Antes de utilizar el JWT, hay que establecer una clave secreta. Una vez que el usuario ha introducido correctamente sus credenciales, el JWT se devuelve con la clave y se guarda localmente. La transmisión debe realizarse a través de HTTPS para que los datos estén mejor protegidos.

De esta manera, cada vez que el usuario accede a recursos protegidos, como a una API o a una ruta protegida, el user agent utiliza el JWT como parámetro (por ejemplo, jwt para peticiones GET) o como header de autorización (para POST, PUT, OPTIONS y DELETE). La otra parte puede descifrar el JSON Web Token y ejecutar la solicitud si la verificación se realiza correctamente.

¿En qué casos se utiliza JSON Web Token?

JSON Web Token ofrece varias ventajas en comparación con el método tradicional de autentificación y autorización con cookies, por lo que se utiliza en las siguientes situaciones:

1. Aplicaciones REST: En las aplicaciones REST, el JWT garantiza la ausencia de estado enviando los datos de autentificación directamente con la petición.
2. Intercambio de recursos de origen cruzado: JSON Web Token envía información mediante el llamado cross-origin resource sharing, lo cual le da una gran ventaja sobre las cookies, que no suelen enviarse con este procedimiento.
3. Uso de varios frameworks: JSON Web Token está estandarizado y puede utilizarse una y otra vez. Cuando se emplean múltiples frameworks, los datos de autentificación pueden compartirse más fácilmente.