JWT

Julian F. Latorre

Julio 2024

¿Qué es JWT?

- JWT (*JSON Web Token*) es un estándar abierto (RFC 7519).
- Transmite información de manera compacta y autónoma entre dos partes como un objeto JSON.
- Se utiliza principalmente para la autenticación y la autorización en aplicaciones web.
- La información puede ser verificada y es confiable porque está firmada digitalmente.

Estructura de un JWT

Un JWT está compuesto por tres partes:

1. **Header (Encabezado)**
   - Define el tipo de token y el algoritmo de firma.
   - Ejemplo: `{"alg": "HS256", "typ": "JWT"}`
2. **Payload (Carga útil)**
   - Contiene las declaraciones o *claims*.
   - Ejemplo: `{"sub": "1234567890", "name": "John Doe", "admin": true}`
3. **Signature (Firma)**
   - Asegura que el token no ha sido alterado.
   - Creada a partir del encabezado, payload y una clave secreta.

Tipos de Claims en JWT

- **Registered Claims:** Datos estándar recomendados como `iss`, `exp`, `sub`, y `aud`.
- **Public Claims:** Definidos por los usuarios para intercambiar información entre partes.
- **Private Claims:** Personalizados y específicos para una necesidad particular.

Proceso de Autenticación con JWT

1. El usuario envía sus credenciales al servidor.
2. El servidor verifica las credenciales y genera un JWT.
3. El cliente almacena el JWT y lo utiliza para futuras solicitudes.
4. El servidor verifica el JWT en cada solicitud.

Ventajas de JWT

- **Escalabilidad:** No se requiere mantener un estado de sesión en el servidor.
- **Autonomía:** El token contiene toda la información necesaria.
- **Seguridad:** Los JWT se pueden firmar para asegurar los datos.

Desventajas de JWT

- **Tamaño:** Los JWT pueden ser más grandes que los identificadores de sesión tradicionales.
- **Revocación de Tokens:** No hay un mecanismo fácil para invalidar un JWT antes de su expiración.

Implementación Básica con Node.js y Express

```
const app = express(); const secretKey = 'supersecretkey';
app.post('/login', (req, res) =>  const userId = 1; const toke
= jwt.sign( id: userId , secretKey,  expiresIn: '1h' ); res.js
auth: true, token: token ); ); app.listen(3000, () =>  console
en puerto 3000'); );
```

Consideraciones de Seguridad

- Utiliza algoritmos de firma seguros, como RS256.
- Protege la clave secreta y utiliza HTTPS para la transmisión.
- Configura la expiración de los tokens para minimizar riesgos.

Conclusión

- Los JWT son una solución potente para autenticación y autorización.
- Facilitan la escalabilidad en aplicaciones web distribuidas.
- Aseguran la transmisión de datos de manera segura y eficiente.