JWT JSON WEB TOKEN

JWT (JSON WEB TOKEN) ES UN ESTÁNDAR QUÉ ESTÁ DENTRO DEL DOCUMENTO RFC 7519.

EN EL MISMO SE DEFINE UN MECANISMO PARA PODER **PROPAGAR ENTRE DOS PARTES**, Y **DE FORMA SEGURA, LA IDENTIDAD DE UN DETERMINADO USUARIO**, ADEMÁS CON UNA SERIE DE CLAIMS O PRIVILEGIOS.

ESTOS **PRIVILEGIOS** ESTÁN **CODIFICADOS** EN OBJETOS DE TIPO **JSON**, QUE SE INCRUSTAN DENTRO DE DEL PAYLOAD O CUERPO DE UN MENSAJE QUE VA FIRMADO DIGITALMENTE.

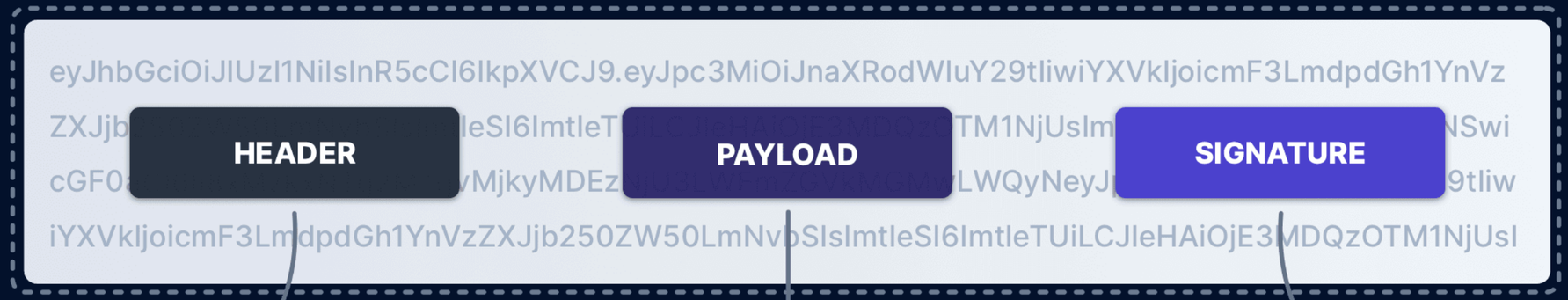
JWT JSON WEB TOKEN

- **CABECERA**: EL ENCABEZADO NORMALMENTE CONSTA DE DOS PARTES: EL TIPO DE TOKEN, QUE ES JWT, Y EL ALGORITMO QUE SE UTILIZA, COMO HMAC.SHA256 O RSASHA256 . ESBASE64URL CODIFICADO PARA FORMAR LA PRIMERA PARTE DEL JWT.

- **CARGA ÚTIL:** LA CARGA ÚTIL CONTIENE LAS RECLAMACIONES. HAY UN CONJUNTO DE RECLAMACIONES REGISTRADAS, POR EJEMPLO,ISS (EDITOR),EXP (TIEMPO DE EXPIRACIÓN),SUB (SUJETO), YAUD (AUDIENCIA). ESTOS RECLAMOS NO SON OBLIGATORIOS, PERO SE RECOMIENDAN PARA PROPORCIONAR UN CONJUNTO DE RECLAMOS ÚTILES E INTEROPERABLES. LA CARGA ÚTIL TAMBIÉN PUEDE INCLUIR ATRIBUTOS ADICIONALES QUE DEFINEN RECLAMOS PERSONALIZADOS, COMO EL ROL DEL EMPLEADO. NORMALMENTE, LA DECLARACIÓN DE SUJETO SE UTILIZA PARA CREAR EL SUJETO DE USUARIO DE OPENID CONNECT. SIN EMBARGO, EL SERVIDOR LIBERTY JVM SE PUEDE CONFIGURAR PARA UTILIZAR UNA NOTIFICACIÓN ALTERNATIVA. LA CARGA ÚTIL ESBASE64URL CODIFICADO PARA FORMAR LA SEGUNDA PARTE DEL JWT.

- **FIRMA:** PARA CREAR LA PARTE DE LA FIRMA, EL ENCABEZADO CODIFICADO Y LA CARGA ÚTIL CODIFICADA SE FIRMAN UTILIZANDO EL ALGORITMO DE FIRMA DEL ENCABEZADO. LA FIRMA SE UTILIZA PARA VERIFICAR QUE EL EMISOR DEL JWT ES QUIEN DICE SER Y PARA GARANTIZAR QUE EL MENSAJE NO HAYA CAMBIADO EN EL CAMINO.

ESTRUCTURA DE UN JWT

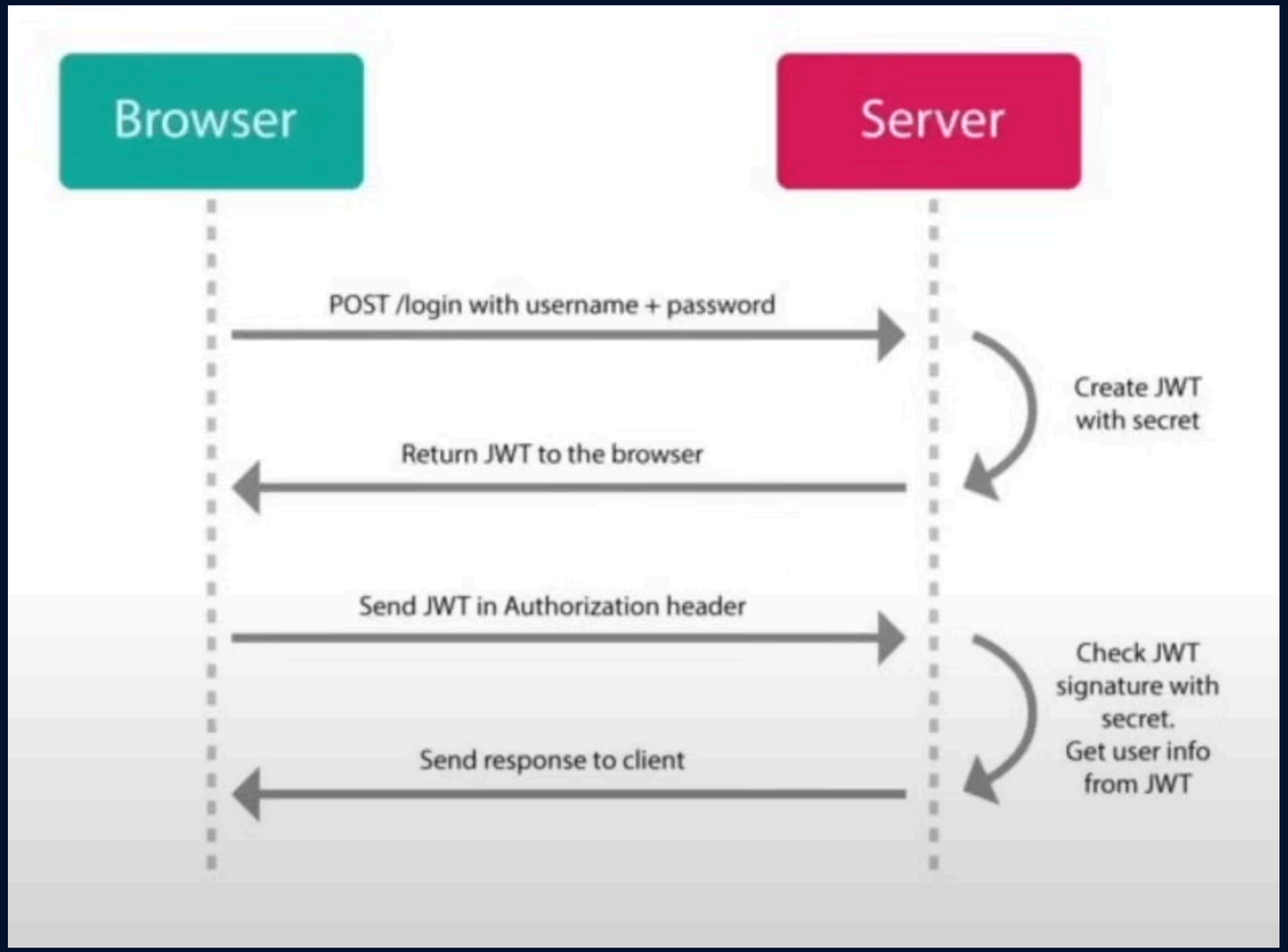



```
{
  “alg”:“RS256”,
  “typ”:“JWT”,
  …
}
```

```
{
  "name" : "Richard Hendricks",
  "exp": 1703037180,
  "sub": "00000000-0000-0000-0000-111111111111",
  "admin": true,
  …
}
```

```
<cryptographic
signature to ensure
integrity>
```

FLUJO COMUN DEL JWT



LOGIN - BACKEND

```
const jwt = require("jsonwebtoken");
const express = require("express");
const cookieParser = require("cookie-parser");

const app = express();
app.use(cookieParser());

app.post("/login", (req, res) => {
    const user = { id: 1, name: "Victor" };
    const token = jwt.sign(user, "secreto_super_seguro", { expiresIn: "1h" });

    res.cookie("jwt", token, {
        httpOnly: true,  // Evita acceso desde JavaScript en el navegador
        secure: true,    // Solo se envía por HTTPS
        sameSite: "Strict",
        maxAge: 1000*60*60
    });

    res.json({ message: "Autenticado correctamente" });
});
```

FETCH POST-LOGIN FRONTEND

```
fetch("https://api.tusitio.com/protegido", {
    method: "GET",
    credentials: "include"  // Importante para enviar cookies
})
.then(res => res.json())
.then(data => console.log(data))
.catch(err => console.error(err));
```

LOGOUT - BACKEND

```
app.post('/logout', (req, res) => {
  res
    .clearCookie(       "jwt"       )
    .json({ message: 'Logout successful' })
})
```

ALGUNAS PROBLEMÁTICAS

🔐 1. No se pueden invalidar fácilmente

A diferencia de sesiones almacenadas en bases de datos, **JWT no se puede revocar antes de que expire.** Si un usuario cierra sesión, el token sigue siendo válido hasta que se agote el tiempo de expiración.

Soluciones posibles:

- Usar una **lista negra** de tokens en la base de datos para invalidarlos manualmente.
- Reducir el tiempo de expiración del JWT y usar **refresh tokens** para renovar la sesión.

ALGUNAS PROBLEMÁTICAS

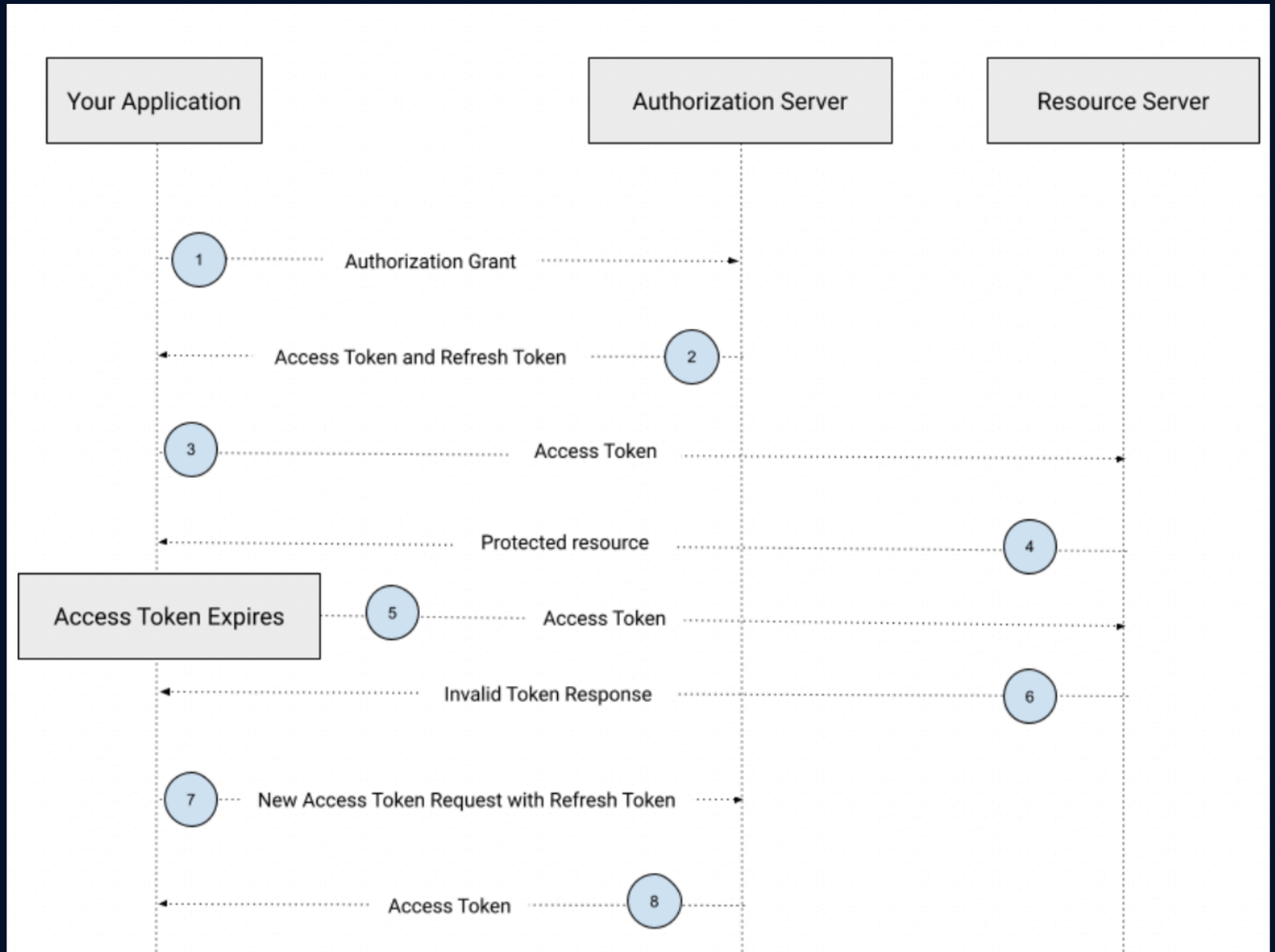
🚨 2. Si un JWT se filtra, es reutilizable hasta que expira

Si un atacante obtiene un JWT, puede usarlo sin restricciones hasta que el token expire, lo que representa un problema de seguridad.

Soluciones posibles:

- Usar `httpOnly` y `secure` en cookies para evitar que JavaScript acceda al token.
- Implementar **rotación de tokens** para que se generen nuevos tokens con cada petición.
- Restringir el uso del JWT por IP o `User-Agent`.

REFRESH TOKENS



PERMISOLOGIA

```
{
  "roles": {
    "admin": {
      "description": "Administrator with full access",
      "permissions": {
        "resources": {
          "user": {
            "methods": ["create", "read", "update", "delete"],
            "conditions": {}
          },
          "product": {
            "methods": ["create", "read", "update", "delete"],
            "conditions": {}
          }
        }
      }
    },
    "editor": {
      "description": "Editor with limited access",
      "permissions": {
        "resources": {
          "product": {
            "methods": ["read", "update"],
            "conditions": {
              "maxPrice": 5000
            }
          }
```