文档

官网: https://jwt.io/

源码: https://github.com/golang-jwt/jwt

原理

客户端通过服务端认证之后,由服务端返回一个JSON对象,发回到客户端。客户端保存该对象用于以后服务器访问凭据,服务端完全依赖该JSON对象来验证客户端的身份。由于JSON数据容易被篡改,因此在服务器生成该对象之后会对该对象进行签名,防止数据被篡改。

jwt结构

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
JzdWIiOiIxMjM0NTY30DkwIiwibmFtZSI6Ikpva
G4gRG91IiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKx
wRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c

```
HEADER: ALGORITHM & TOKEN TYPE

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

PAYLOAD: DATA

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

VERIFY SIGNATURE

HMACSHA256(
    base64UrlEncode(header) + "." +
    base64UrlEncode(payload),
    your-256-bit-secret
)    □ secret base64 encoded
```

如图: jwt结构分为三个部分, Header、Payload、Signature

1. Header: Algorithm (算法) , 即签名算法。

2. Payload:存储信息的JSON对象

3. Signature: Payload基于Header指定算法的签名结果

- 4. Header与Payload部分,本身并没有被加密,而是做了Base64URL编码。
- 5. Base64URL与Base64算法类似,Base64中+、/和= 在URL中有特殊含义,因此Base64URL 对它们做了处理

应用场景

- 1. 充当认证令牌,替代传统的session数据存储,通过jwt完全的依赖客户端保存认证信息
- 2. 基于jwt可携带数据的特性,可将其用于数据传输、信息交换的场景

弊端

- 1. JWT默认不加密,在不加密的情况写,不要携带敏感数据
- 2. 由于是客户端保存,jwt一旦签发,将无法再使用过程中废止,除非部署额外的逻辑处理
- 3. jwt本身包含认证信息,一旦泄露,任何使用该令牌的人都可以获得相应的权限,为减少盗用的可能,应当将有效期设置的比较短。

使用

1. 依赖安装

go get -u github.com/golang-jwt/jwt/v4

2. 依赖导入

import "github.com/golang-jwt/jwt/v4"

字段说明

Payload 部分是一个 JSON 对象,用来存放实际需要传递的数据。JWT 规定了7个官方字段,供选用。

• iss (issuer): 签发人

• exp (expiration time): 过期时间

• sub (subject): 主题

• aud (audience): 受众

• nbf (Not Before): 生效时间

• iat (Issued At): 签发时间

• jti (JWT ID):编号