文章编号：2096-1472(2019)-12-18-03
DOI:10.19644/j.cnki.issn2096-1472.2019.12.005

一种基于JWT认证token刷新机制研究

周 虎

(江苏联合职业技术学院徐州财经分院，江苏 徐州 221008)

摘 要：JWT(JsonWebToken)认证作为一种服务器端无状态验证方式，在分布式开发中得到了广泛的应用。但是，由于token信息的本身特点，支持的有效时间是固定的，当在token有效时间内用户操作没有完成，操作就会中断，需要重新登录验证。本文通过对比传统Cookie/Session身份验证机制在分布式开发中存在的不足，提出了一种基于JWT认证过程中动态刷新token的方法，有效地解决了分布式开发中会话共享问题，并通过在webApi开发中得到了具体的应用，表明了该方法的有效性和实用性。

关键词：JWT；token；认证；webApi
中图分类号：TP311.11 **文献标识码**：A

Research on a Refresh Mechanism of Authentication Token Based on JWT

ZHOU Hu

(*Xuzhou Finance and Economics Branch,Jiangsu Union Technical Institute,Xuzhou* 221008,*China*)

Abstract:JWT (JSON Web Token) authentication,as a server-side stateless authentication method,has been widely used in distributed development.However,due to the characteristics of token information itself,the supported valid time is fixed.When the user operation is not completed within the token valid time,the operation will be interrupted and the login verification needs to be re-registered.By comparing the traditional Cookie/Session authentication mechanism in distributed development of deficiencies,this paper proposes a method based on dynamic refresh token JWT certification process,solving effectively the problem of the sharing Session in distributed development.The application in the webApi development shows that the method is effective and practical.

Keywords:JWT;token;authentication;webApi

1 引言(Introduction)

目前，随着Web应用技术的不断发展，应用程序的规模不断扩大，分布式应用开发逐渐成了主流趋势。传统的Web程序认证技术基于Session和cookie技术的弊端已经逐步显露出来，在多台服务器之间如何实现会话共享，保持会话一致，即使实现了会话复制，随着服务器数量的不断增加，Session复制性能也会急剧下降[1]，给服务器的性能带来损失，它们已经不能满足现代Web应用程序的发展需要。客户端和服务器端会话分离，保持服务器端无状态提供服务，同时又要保证服务器端资源的安全性，为了克服这些不足，基于JWT认证技术得到了广泛关注，其通过特定的算法生成用户唯一的令牌token对象，然后在用户数据请求过程中携带此token即可完成身份有效性校验，一方面保证了前后端分离，同时保护用户访问安全[2]。它与Session机制最大的区别就是一个将用户认证信息保存在服务器端，一个保存在客户端，大大减轻了服务器端的存储压力，同时提供了跨域访问和分布式服务器集群访问的身份认证。

基于JWT的认证方式虽然给身份认证方式带来了极大的方便，但是也存在一些弊端，当服务器端签发token时，它包含了签发时间和失效时间，当在有效时间内，如果用户操作没有完成，服务器端将要求客户端重新登录，一方面可能导致数据丢失，同时给用户体验造成了负面影响。所以针对这种情况，本文提出了一种基于客户端异步刷新token方式，一方面延长了token的有效时间，同时改善了用户体验，并通过JWT结合.netCore中的WebApi技术证明了该方法的可行性。

2 JWT(Json Web Token)

JWT是一种基于Json格式Web token，定义了一个紧凑的自包含的方式在不同实体之间安全传输信息[3]。它是在

Web环境下两个实体之间传输数据的一项标准。实际上传输的就是一个字符串。token信息包含了三个部分：HEAD头部，用来声明token类型和加密的算法；PAYLOAD载荷部分，主要用于传递信息的载体，常用的有iss(签发者)、iat(签发时间)、exp(过期时间)、sub(面向的用户)、aud(接收方)；SIGNATURE部分，用于将Header和Payload编码后的字符串拼接后再用HS256签名算法(Header中alg指明的算法)加密，在加密的过程中还需加上secret(密钥)，最后得到签名，具体结构如图1所示。

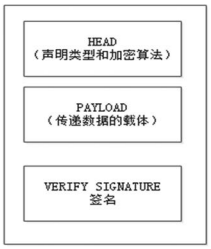

图1 token结构

Fig.1 Token structure

对于JWT认证方式，首先由客户端发起用户登录请求，在服务端进行用户信息验证，验证成功后，将给客户端签发token，签发成功后，客户端将签发的token保存到客户端，在进行服务器资源访问时，需要在请求头中放置签发的token，服务器端接收到客户端发送的token时，对其进行验证是否合法，是否在有效时间内，如果在有效时间内并且用户身份合法，将返回服务器资源，具体流程如图2所示。

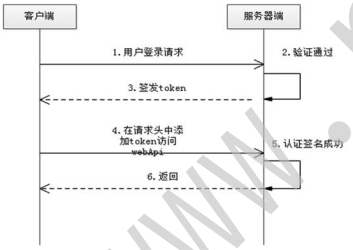

图2 token认证方式

Fig.2 Token authentication method

基于JWT的认证方式，由于Json的通用性，所以JWT是可以进行跨语言支持，有了PAYLOAD部分，所以JWT可以在自身存储一些其他业务逻辑所必要的非敏感信息。JWT构成非常简单，字节占用很小，所以它是非常便于传输的，并且不在服务端保存会话信息,更易于应用的扩展[4]。

3　token动态刷新功能的实现(Implementation of token dynamic refresh function)

当客户端发出认证请求，服务端接收客户端发送的用户信息并验证合法性，验证通过后给客户端返回颁发的token、颁发时间和失效时间，客户端根据接收的信息，计算客户端当前时间和token的颁发时间的时间偏差，将token、颁发时间、失效时间和时间偏差保存到浏览器的localstorage中。客户端在进行WebApi访问时，客户端从localstorage中取出信息计算当前时间加上时间偏差并和token的失效时间进行比较，如果距离失效时间不在限定时间范围内(比如2分钟)，直接进行请求WebApi，否则，先进行异步请求登录认证，然后在进行WebApi请求[5]，具体流程如图3所示。

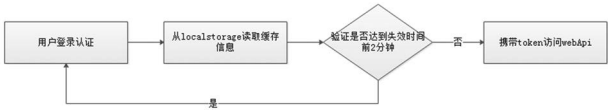

图3 token动态刷新流程图

Fig.3 Token dynamic refresh flow chart

当客户端重新请求时，服务器端将重新颁发token，并将颁发时间、失效时间和客户端的时间偏差值重新保存到客户端的localstorage中，再次请求WebApi时将携带新的token。

为了验证客户端动态刷新token功能，本文采用JWT结合.netCore平台中的WebApi技术，实现对客户端访问控制，主要实现两个部分：服务端认证并签发token和客户端请求token。

3.1　服务端认证并签发token

当客户端发送请求用户名和密码进行验证时，服务器端先验证用户的合法性，如果用户合法，将给用户颁发token，实现代码如下：

```
if (validateUser(userInfo)//验证用户合法
        {
    long curTime=DateTimeOffset.UtcNow.ToUnixTimeMilliseconds();
    long endTime=DateTimeOffset.UtcNow.AddMinutes(20).ToUnixTimeMilliseconds();
    LoginInfo pUserLoginInfo=new LoginInfo() { Name=UserName,Password=PassWord,exp=endTime };
    var token=new JwtBuilder()//签发token
    .WithAlgorithm(new HMACSHA256Algorithm())//指定算法
     .WithSecret(m_Secret)//密钥
     .ExpirationTime(DateTime.UtcNow.AddMinutes(5))//过期时间
     .AddClaim("issuser","xzcx")//颁发者
    .AddClaim("audience",userInfo.Name)//接收者
                .Build();
                jwtResult=new JwtResult()
                {
                    token=token,
                    statusCode="200",
                    message="success",
                    assTime=curTime,//签发时间
```

```
                endTime=endTime
            };
        return JsonConvert.SerializeObject(jwtResult);
        }
```

当服务器端签发token时，指定签发时间和失效时间，在上面程序中，指定token失效时间为当前时间后20分钟，并且所有时间格式按照Unix格式时间戳来保存。签发token成功后将token和状态码，返回消息以及签发时间和失效时间以JSON格式响应到客户端[6]。

3.2 客户端保存token并请求WebApi

当客户端接收到服务端响应的token信息，将其保存到客户端的localstorage中，实现代码如下：

```
$.ajax({
    url:'https://localhost:44337/api/service',
    data:{UserName:username,PassWord:password},
    success:function (data) {
    var result=$.parseJSON(data);
    localStorage.setItem("token",result.token);//保存token
    localStorage.setItem("assTime",result.assTime);//保存颁发时间
    localStorage.setItem("endTime",result.endTime);//保存失效时间
    localStorage.setItem("offsetTime",new Date().getTime()-result.curTime);//保存客户端和服务端时间差
            }
})
```

当用户请求WebApi时，将从localstorage中取出token信息，并加入请求head中，但是，在请求前需要判断一下当前时间和token的失效时间差是否在设定时间内，如果在设定内，需要重新请求认证并保存新的token，然后再次访问WebApi[7]，实现代码如下：

```
var token=localStorage.getItem("token");
    var curTime=localStorage.getItem("curTime");
    var offsetTime=localStorage.getItem("offsetTime");
    var endTime=localStorage.getItem("endTime");
    if ((parseInt(endTime)-(new Date().getTime()+parseInt(offsetTime))<120000)) {//如果时间差在2分钟以内则重新请求认证
    var promise=new Promise(function (resolve,reject) {
            login()//重新登录验证
        })
    promise.then(visitApi(url));    //请求成功后再访问WebApi
        }
        else {
    visitApi(url);//如果时间差大于2分钟直接访问WebApi
        }
```

在实现重新请求认证时，需要用到Promise对象，因为在重新请求认证和访问WebApi时有先后顺序，必须要等到服务端重新颁发token并且客户端保存成功后再次携带新的token访问WebApi，否则将访问失败。经过测试发现，客户端动态刷新token方法在不影响用户操作体验的情况下动态延长token有效期限，有效地解决了token失效问题[8]。

4 结论(Conclusion)

本文通过对目前基于JWT认证机制下token失效问题进行了深入研究，为了更好保证在不影响用户操作的前提下，动态延长token的有效期限，提出了一种基于客户端在设定时间内动态刷新token的方法，并在.netCore平台下的WebApi技术中得到了具体的验证，该方法简单有效地解决了token失效问题，同时大大减轻了服务器端的负担，也为前后端分离提供有效地解决方案[9]。

参考文献(References)

[1] 柳纲,张毅.服务端无状态技术研究[J].电力信息与通信技术,2017(11):49-54.

[2] 陈宇收,饶宏博,王英明,等.基于JWT 的前后端分离程序设计研究[J].电脑编程技巧与维护,2019(09):11-12.

[3] 范展源,罗福强.JWT认证技术及其在Web中的应用[J].数字技术与应用,2016(02):114.

[4] 项武铭,鲍亮,俞少华.基于JWT的RESTful API角色权限验证方案设计[J].现代计算机(专业版),2018(12):82-85.

[5] 黄伟民,陈可新.基于Token的物联网云平台系统身份认证机制研究[J].智库时代,2018(42):195-196.

[6] 王威.物联网的体系结构与相关技术[J].电子技术与软程,2018(22):14-15.

[7] 王守昌.JWT从入门到精通[EB/OL].https://www.cnblogs.com/wangshouchang/p/9551748.html,2018-08-28.

[8] LIU Gang,HE Jing.Design of a high performance WEB cluster[J].Electric Power Information and Communication Technology,2017,15(3):95-100.

[9] Woo-Suk Park,Dong-Yeop Hwang.A TOTP-Based Two Factor Authentication Scheme for Hyperledger Fabric Blockchain[J].2018 Tenth International Conference on Ubiquitous and Future Networks(ICUFN),2018(05):78-81.

作者简介：

周 虎(1977-)，男，硕士，讲师.研究领域：应用程序设计.