基于 JWT 的接口权限验证

主讲教师：（大地）
合作网站：www.itying.com （IT 营）
我的专栏：https://www.itying.com/category-79-b0.html

一、关于接口的安全验证

关于接口安全验证的解决方案有很多：

1、可以用 Session 来实现安全验证
2、对请求接口的参数进行签名的签名验证
3、使用 JWT 实现接口的验证。
...

1、基于 Session 的安全验证

Session 存储在服务器，用户用户比较少的话是一种简单的安全验证机制，但是涉及到跨域的话需要进行一些配置。用户量非常非常大的话会耗费一定的服务器资源（中小项目互联不计）。

关于 cookie 和 session 跨域可以参考:
http://bbs.itying.com/topic/5c45639b3da5ae17b03606b0

2、对请求参数进行加密的签名验证

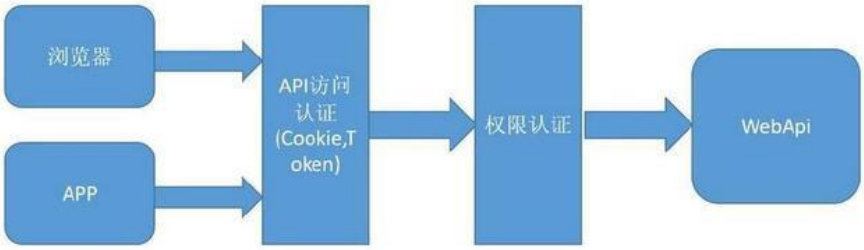
涉及公钥、私钥、签名等

3、JWT

JWT 全称 JSON Web Token，是目前比较流行的另一种跨域身份验证解决方案。也是被很多人用坏的一种安全验证机制。

二、关于 JWT

JWT 全称 JSON Web Token，是目前比较流行的另一种跨域身份验证解决方案。



三、Nodejs 中使用 JWT 实现接口的安全验证

1、生成 token

安装 jsonwebtoken

```
cnpm install jsonwebtoken --save
```

生成 tooken

```
var jwt = require('jsonwebtoken');
router.get('/', function (req, res, next) {
    var token = jwt.sign({ name: '张三' }, 'this is sign',{
        expiresIn:60
    });
    res.send(token);
});
```

2、获取请求头里面的 tooken

```
cnpm install basic-auth --save
```

```
router.get('/addressList', function (req, res, next) {
    var tooken = auth(req);
})
```

3、验证客户端传过来 tooken 的合法性

```
router.get('/addressList', function (req, res, next) {
    var tooken = auth(req);
    if (tooken) {
        try {
            var v = jwt.verify(tooken.name, 'this is sign');
            console.log(v);
            if (v) {
                res.send('有权限');
            } else {
                res.send('没有权限');
            }
        } catch (error) {
            res.send(error);
        }
    }else{
        res.send('没有权限');
    }
});
```

四、Vue React Angular 使用 Axios 访问基于 Jwt 的接口

```
axios.get("http://localhost:3000/api/list", {
                auth: {
                    username:
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuYW1lIjoi5byg5LiJIiwiaWF0IjoxNTcxMTIwNTE3LCJleHA
iOjE1NzExMjIzMTd9.Kcbvg7AGqZlmVyUb8CKyO3fqf-zEwqIWEN2nsUSl17Q",
                    password: "123456"
                }
            })
            .then(function(response) {
                // handle success
                console.log(response);
            })
            .catch(function(error) {
```

```
    // handle error
    console.log(error);
})
.finally(function() {
    // always executed
});
```

五、关于 Jwt 的一些问题

（1）JWT 默认是不加密，但也是可以加密的。生成原始 Token 以后，可以用密钥再加密一次。

（2）JWT 不加密的情况下，不能将秘密数据写入 JWT。

（3）JWT 不仅可以用于认证，也可以用于交换信息。有效使用 JWT，可以降低服务器查询数据库的次数。

（4）JWT 的最大缺点是，由于服务器不保存 session 状态，因此无法在使用过程中废止某个 token，或者更改 token 的权限。也就是说，一旦 JWT 签发了，在到期之前就会始终有效，除非服务器部署额外的逻辑。

（5）JWT 本身包含了认证信息，一旦泄露，任何人都可以获得该令牌的所有权限。为了减少盗用，JWT 的有效期应该设置得比较短。对于一些比较重要的权限，使用时应该再次对用户进行认证。

（6）为了减少盗用，JWT 不应该使用 HTTP 协议明码传输，要使用 HTTPS 协议传输。