

前后端分离之JWT用户认证



作者 lion1ou (/u/5caf48f19592) + 关注

2017.01.22 19:11 字数 2765 阅读 1351 评论 8 喜欢 36

(/u/5caf48f19592)

在前后端分离开发时为什么需要用户认证呢？原因是由于HTTP协定是不储存状态的 (stateless)，这意味着当我们透过帐号密码验证一个使用者时，当下一个request请求时它就把刚刚的资料忘了。于是我们的程序就不知道谁是谁，就要再验证一次。所以为了保证系统安全，我们就需要验证用户否处于登录状态。

传统方式

前后端分离通过Restful API进行数据交互时，如何验证用户的登录信息及权限。在原来的项目中，使用的是最传统也是最简单的方式，前端登录，后端根据用户信息生成一个 token，并保存这个 token 和对应的用户id到数据库或Session中，接着把 token 传给用户，存入浏览器 cookie，之后浏览器请求带上这个cookie，后端根据这个cookie值来查询用户，验证是否过期。

但这样做问题就很多，如果我们的页面出现了 XSS 漏洞，由于 cookie 可以被 JavaScript 读取，XSS 漏洞会导致用户 token 泄露，而作为后端识别用户的标识，cookie 的泄露意味着用户信息不再安全。尽管我们通过转义输出内容，使用 CDN 等可以尽量避免 XSS 注入，但谁也不能保证在大型的项目中不会出现这个问题。

在设置 cookie 的时候，其实你还可以设置 httpOnly 以及 secure 项。设置 httpOnly 后 cookie 将不能被 JS 读取，浏览器会自动的把它加在请求的 header 当中，设置 secure 的话，cookie 就只允许通过 HTTPS 传输。secure 选项可以过滤掉一些使用 HTTP 协议的 XSS 注入，但并不能完全阻止。

httpOnly 选项使得 JS 不能读取到 cookie，那么 XSS 注入的问题也基本不用担心了。但设置 httpOnly 就带来了另一个问题，就是很容易的被 XSRF，即跨站请求伪造。当你浏览器开着这个页面的时候，另一个页面可以很容易的跨站请求这个页面的内容。因为 cookie 默认被发了出去。

另外，如果将验证信息保存在数据库中，后端每次都需要根据 token 查出用户 id，这就增加了数据库的查询和存储开销。若把验证信息保存在session中，有加大了服务器端的存储压力。那我们可不可以不要服务器去查询呢？如果我们生成 token 遵循一定的规律，比如我们使用对称加密算法来加密用户 id 形成 token，那么服务端以后其实只要解密该 token 就可以知道用户的 id 是什么了。不过呢，我只是举个例子而已，要是真这么做，只要你的对称加密算法泄露了，其他人可以通过这种加密方式进行伪造 token，那么所有用户信息都不再安全了。恩，那用非对称加密算法来做呢，其实现在有个规范就是这样做的，就是我们接下来要介绍的 JWT。

Json Web Token (JWT)

JWT 是一个开放标准(RFC 7519)，它定义了一种用于简洁，自包含的用于通信双方之间以 JSON 对象的形式安全传递信息的方法。JWT 可以使用 HMAC 算法或者是 RSA 的公钥密钥对进行签名。它具备两个特点：

- 简洁(Compact)

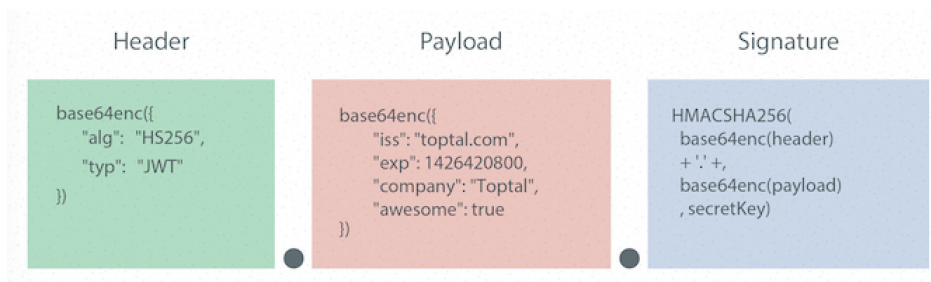


可以通过URL, POST 参数或者在 HTTP header 发送, 因为数据量小, 传输速度快

- 自包含(Self-contained)

负载中包含了所有用户所需要的信息, 避免了多次查询数据库

JWT 组成



- Header 头部

头部包含了两部分, token 类型和采用的加密算法

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

它会使用 Base64 编码组成 JWT 结构的第一部分, 如果你使用 Node.js, 可以用 Node.js 的包 `base64url` 来得到这个字符串。

Base64 是一种编码, 也就是说, 它是可以被翻译回原来的样子来的。它并不是一种加密过程。

- Payload 负载

这部分就是我们存放信息的地方了, 你可以把用户 ID 等信息放在这里, JWT 规范里面对于这部分有进行了比较详细的介绍, 常用的由 `iss` (签发者), `exp` (过期时间), `sub` (面向的用户), `aud` (接收方), `iat` (签发时间)。

```
{
  "iss": "lion1ou JWT",
  "iat": 1441593502,
  "exp": 1441594722,
  "aud": "www.example.com",
  "sub": "lion1ou@163.com"
}
```

同样的, 它会使用 Base64 编码组成 JWT 结构的第二部分

- Signature 签名

前面两部分都是使用 Base64 进行编码的, 即前端可以解开知道里面的信息。Signature 需要使用编码后的 header 和 payload 以及我们提供的一个密钥, 然后使用 header 中指定的签名算法 (HS256) 进行签名。签名的作用是保证 JWT 没有被篡改过。



三个部分通过 . 连接在一起就是我们的 JWT 了，它可能长这个样子，长度貌似和你的加密算法和私钥有关系。

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjU3ZmVmMTY0ZTU0YWY2NGZmYzUzZGJkNSIsInhzcmYiOiI0ZWE1YzUwOGE2NTY2ZTc2MjQwNTQzZjhmZWlWbmZkdDU3Nzc3YmUzOTU0OWM0MDE2NDM2YWZkYTY1ZDIzMzBlIiwiaWF0IjoxNDc2NDI3OTMzfQ.PA3QjeyZSUh7H0GF0vJaKW4LjKJuC3dVLQiY4hi18s
```

其实到这一步可能就会有人会想了，HTTP 请求总会带上 token，这样这个 token 传来传去占用不必要的带宽啊。如果你这么想了，那你可以去了解下 HTTP2，HTTP2 对头部进行了压缩，相信也解决了这个问题。

• 签名的目的

最后一步签名的过程，实际上是对头部以及负载内容进行签名，防止内容被篡改。如果有人对头部以及负载的内容解码之后进行修改，再进行编码，最后加上之前的签名组合形成新的JWT的话，那么服务器端会判断出新的头部和负载形成的签名和JWT附带上的签名是不一样的。如果要对新的头部和负载进行签名，在不知道服务器加密时用的密钥的话，得出来的签名也是不一样的。

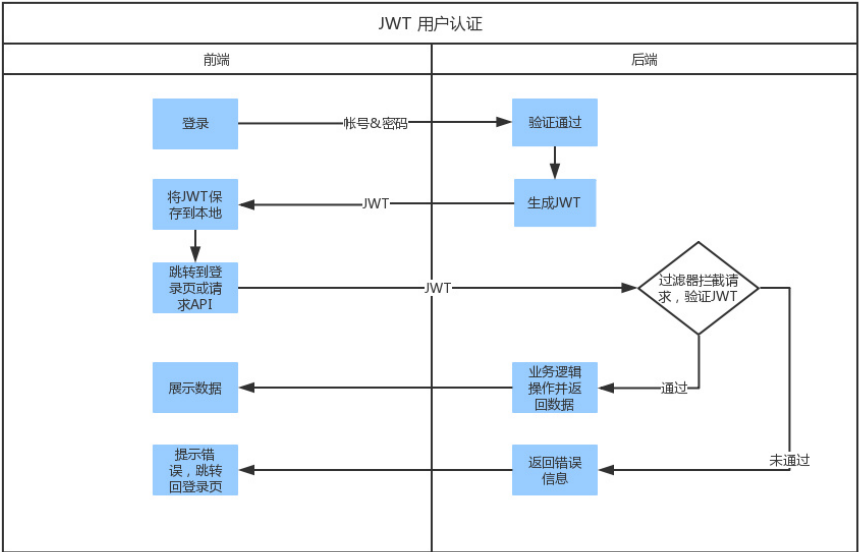
• 信息暴露

在这里大家一定会问一个问题：Base64是一种编码，是可逆的，那么我的信息不就被暴露了吗？

是的。所以，在JWT中，不应该在负载里面加入任何敏感的数据。在上面的例子中，我们传输的是用户的User ID。这个值实际上不是什么敏感内容，一般情况下被知道也是安全的。但是像密码这样的内容就不能被放在JWT中了。如果将用户的密码放在了JWT中，那么怀有恶意的第三方通过Base64解码就能很快地知道你的密码了。

因此JWT适合用于向Web应用传递一些非敏感信息。JWT还经常用于设计用户认证和授权系统，甚至实现Web应用的单点登录。

JWT 使用



- 1. 首先，前端通过Web表单将自己的用户名和密码发送到后端的接口。这一过程一般是一个HTTP POST请求。建议的方式是通过SSL加密的传输（https协议），从而避免

敏感信息被嗅探。

2. 后端核对用户名和密码成功后，将用户的id等其他信息作为JWT Payload（负载），将其与头部分别进行Base64编码拼接后签名，形成一个JWT。形成的JWT就是一个形同`lll.zzz.xxx`的字符串。
3. 后端将JWT字符串作为登录成功的返回结果返回给前端。前端可以将返回的结果保存在localStorage或sessionStorage上，退出登录时前端删除保存的JWT即可。
4. 前端在每次请求时将JWT放入HTTP Header中的Authorization位。（解决XSS和XSRF问题）
5. 后端检查是否存在，如存在验证JWT的有效性。例如，检查签名是否正确；检查Token是否过期；检查Token的接收方是否是自己（可选）。
6. 验证通过后后端使用JWT中包含的用户信息进行其他逻辑操作，返回相应结果。

和Session方式存储id的差异

Session方式存储用户id的最大弊病在于Session是存储在服务器端的，所以需要占用大量服务器内存，对于较大型应用而言可能还要保存许多的状态。一般而言，大型应用还需要借助一些KV数据库和一系列缓存机制来实现Session的存储。

而JWT方式将用户状态分散到了客户端中，可以明显减轻服务端的内存压力。除了用户id之外，还可以存储其他的和用户相关的信息，例如该用户是否是管理员、用户所在的分组等。虽说JWT方式让服务器有一些计算压力（例如加密、编码和解码），但是这些压力相比磁盘存储而言可能就不算什么了。具体是否采用，需要在不同场景下用数据说话。

• 单点登录

Session方式来存储用户id，一开始用户的Session只会存储在一台服务器上。对于有多个子域名的站点，每个子域名至少会对应一台不同的服务器，例如：`www.taobao.com`，`nv.taobao.com`，`nz.taobao.com`，`login.taobao.com`。所以如果要实现在`login.taobao.com`登录后，在其他的子域名下依然可以取到Session，这要求我们在多台服务器上同步Session。使用JWT的方式则没有这个问题的存在，因为用户的状态已经被传送到客户端。

总结

JWT的主要作用在于（一）可附带用户信息，后端直接通过JWT获取相关信息。（二）使用本地保存，通过HTTP Header中的Authorization位提交验证。但其实关于JWT存放到哪里一直有很多讨论，有人说存放到本地存储，有人说存 cookie。个人偏向于放在本地存储，如果你有什么意见和看法欢迎提出。

参考：1 (<https://segmentfault.com/a/1190000005783306>) 2 (<http://blog.rainy.im/2015/06/10/react-jwt-pretty-good-practice/>) 3 (<https://ruiming.me/archives/41>)

转载请标注原文地址

<http://lion1ou.win/2017/01/18/> (<http://lion1ou.win/2017/01/18/>)

(end)



lion1ou (/u/5caf48f19592)

写了 17577 字，被 29 人关注，获得了 64 个喜欢 (/u/5caf48f19592)

+ 关注

如果觉得我的文章对您有用，请随意赞赏。您的支持将鼓励我继续创作！

赞赏支持

喜欢 (/sign_in) | 36



更多分享

(http://cwb.assets.jianshu.io/notes/images/8639454/v



登录 (/sign_in) 后发表评论

8条评论 只看作者

按喜欢排序 按时间正序 按时间倒序



半隻饅頭 (/u/e01edda0af05)

2楼 · 2017.03.18 23:48 (/u/e01edda0af05)

有个问题：
服务端生成token后直接返回给前端？
后面前端带token再次请求api，服务端拿什么去验证前端带过来的token？

赞

回复

lion1ou (/u/5caf48f19592)：@半隻饅頭 (/users/e01edda0af05) 服务端通过解析前端请求时带的jwt,然后通过jwt内带有的信息验证用户是否正确并有效。
2017.03.18 23:51 回复


半隻饅頭 (/u/e01edda0af05)：@lion1ou (/users/5caf48f19592) 如果别人拿到这个token,是不是可以一直请求？
2017.03.19 03:33 回复


lion1ou (/u/5caf48f19592)：@半隻饅頭 (/users/e01edda0af05) 每一条jwt都可以设置过期时间的。
2017.03.19 03:49 回复


添加新评论

还有4条评论，展开查看

被以下专题收入，发现更多相似内容

- 

程序员 (/c/NEt52a?utm_source=desktop&utm_medium=notes-included-collection)
- 

前端开发 (/c/3da0aa33ba76?utm_source=desktop&utm_medium=notes-included-collection)
- 

WEB前端程序开发 (/c/8df1f704f517?)

↑

🔗

utm_source=desktop&utm_medium=notes-included-collection)



开源工具技巧 (/c/10f89e9948b9?

utm_source=desktop&utm_medium=notes-included-collection)



mygeneseeq (/c/b3d002510354?

utm_source=desktop&utm_medium=notes-included-collection)



JWT (/c/f47cf78d4aa4?utm_source=desktop&utm_medium=notes-

included-collection)



架构 (/c/49584a70aece?utm_source=desktop&utm_medium=notes-

included-collection)

