参考：<http://blog.leapoahead.com/2015/09/06/understanding-jwt/>

# JSON Web Token - 在Web应用间安全地传递信息

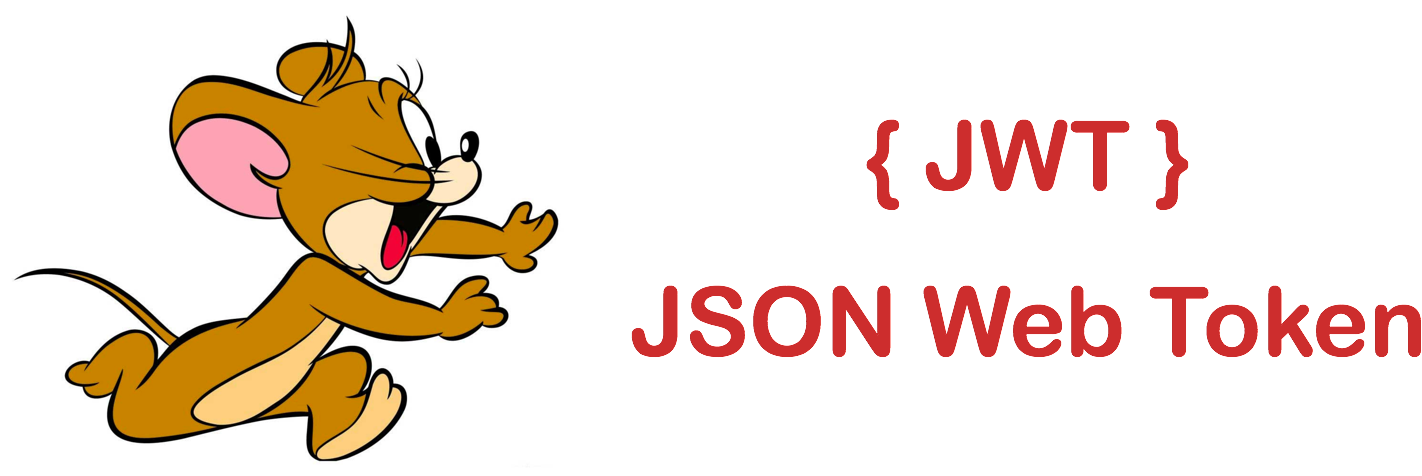
9月 06, 2015 in [Engineering](http://blog.leapoahead.com/categories/Engineering/)

JSON Web Token（JWT）是一个非常轻巧的[规范](https://tools.ietf.org/html/draft-ietf-oauth-json-web-token-32)。这个规范允许我们使用JWT在用户和服务器之间传递安全可靠的信息。

让我们来假想一下一个场景。在A用户关注了B用户的时候，系统发邮件给B用户，并且附有一个链接“点此关注A用户”。链接的地址可以是这样的

|  |  |
| --- | --- |
| 1 | https://your.awesome-app.com/make-friend/?from\_user=B&target\_user=A |

上面的URL主要通过URL来描述这个当然这样做有一个弊端，那就是要求用户B用户是一定要先登录的。可不可以简化这个流程，让B用户不用登录就可以完成这个操作。JWT就允许我们做到这点。



### JWT的组成

一个JWT实际上就是一个字符串，它由三部分组成，**头部**、**载荷**与**签名**。

##### 载荷（Payload）

我们先将上面的添加好友的操作描述成一个JSON对象。其中添加了一些其他的信息，帮助今后收到这个JWT的服务器理解这个JWT。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | {  "iss": "John Wu JWT",  "iat": 1441593502,  "exp": 1441594722,  "aud": "www.example.com",  "sub": "jrocket@example.com",  "from\_user": "B",  "target\_user": "A"  } |

这里面的前五个字段都是由JWT的标准所定义的。

* iss: 该JWT的签发者
* sub: 该JWT所面向的用户
* aud: 接收该JWT的一方
* exp(expires): 什么时候过期，这里是一个Unix时间戳
* iat(issued at): 在什么时候签发的

这些定义都可以在[标准](https://tools.ietf.org/html/draft-ietf-oauth-json-web-token-32)中找到。

将上面的JSON对象进行[base64编码]可以得到下面的字符串。这个字符串我们将它称作JWT的**Payload**（载荷）。

|  |  |
| --- | --- |
| 1 | eyJpc3MiOiJKb2huIFd1IEpXVCIsImlhdCI6MTQ0MTU5MzUwMiwiZXhwIjoxNDQxNTk0NzIyLCJhdWQiOiJ3d3cuZXhhbXBsZS5jb20iLCJzdWIiOiJqcm9ja2V0QGV4YW1wbGUuY29tIiwiZnJvbV91c2VyIjoiQiIsInRhcmdldF91c2VyIjoiQSJ9 |

如果你使用Node.js，可以用Node.js的包[base64url](https://github.com/brianloveswords/base64url)来得到这个字符串。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var base64url = require('base64url')  var header = {  "from\_user": "B",  "target\_user": "A"  }  console.log(base64url(JSON.stringify(header)))  // 输出：eyJpc3MiOiJKb2huIFd1IEpXVCIsImlhdCI6MTQ0MTU5MzUwMiwiZXhwIjoxNDQxNTk0NzIyLCJhdWQiOiJ3d3cuZXhhbXBsZS5jb20iLCJzdWIiOiJqcm9ja2V0QGV4YW1wbGUuY29tIiwiZnJvbV91c2VyIjoiQiIsInRhcmdldF91c2VyIjoiQSJ9 |

小知识：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。

##### 头部（Header）

JWT还需要一个头部，头部用于描述关于该JWT的最基本的信息，例如其类型以及签名所用的算法等。这也可以被表示成一个JSON对象。

|  |  |
| --- | --- |
| 1  2  3  4 | {  "typ": "JWT",  "alg": "HS256"  } |

在这里，我们说明了这是一个JWT，并且我们所用的签名算法（后面会提到）是HS256算法。

对它也要进行Base64编码，之后的字符串就成了JWT的**Header**（头部）。

|  |  |
| --- | --- |
| 1 | eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 |

##### 签名（签名）

将上面的两个编码后的字符串都用句号.连接在一起（头部在前），就形成了

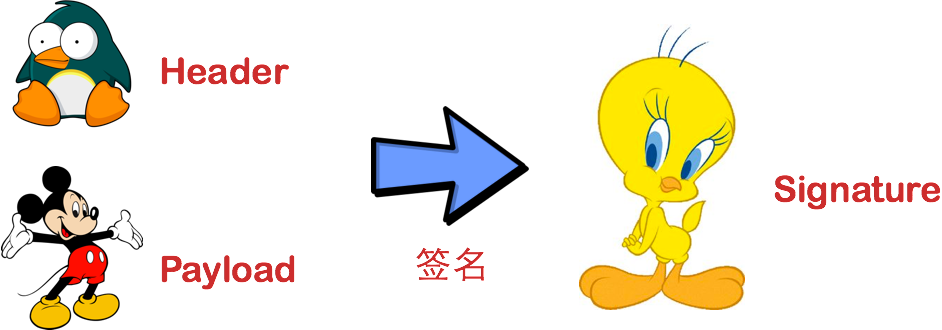
|  |  |
| --- | --- |
| 1 | eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJmcm9tX3VzZXIiOiJCIiwidGFyZ2V0X3VzZXIiOiJBIn0 |

这一部分的过程在[node-jws的源码](https://github.com/brianloveswords/node-jws/blob/master/lib/sign-stream.js)中有体现

最后，我们将上面拼接完的字符串用HS256算法进行加密。在加密的时候，我们还需要提供一个密钥（secret）。如果我们用mystar作为密钥的话，那么就可以得到我们加密后的内容

|  |  |
| --- | --- |
| 1 | rSWamyAYwuHCo7IFAgd1oRpSP7nzL7BF5t7ItqpKViM |

这一部分又叫做**签名**。



最后将这一部分签名也拼接在被签名的字符串后面，我们就得到了完整的JWT

|  |  |
| --- | --- |
| 1 | eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJmcm9tX3VzZXIiOiJCIiwidGFyZ2V0X3VzZXIiOiJBIn0.rSWamyAYwuHCo7IFAgd1oRpSP7nzL7BF5t7ItqpKViM |

于是，我们就可以将邮件中的URL改成

|  |  |
| --- | --- |
| 1 | https://your.awesome-app.com/make-friend/?jwt=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJmcm9tX3VzZXIiOiJCIiwidGFyZ2V0X3VzZXIiOiJBIn0.rSWamyAYwuHCo7IFAgd1oRpSP7nzL7BF5t7ItqpKViM |

这样就可以安全地完成添加好友的操作了！

且慢，我们一定会有一些问题：

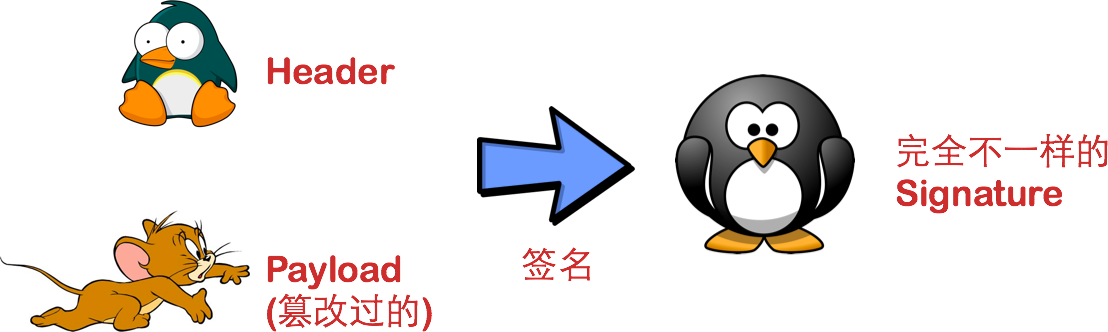
1. 签名的目的是什么？
2. Base64是一种编码，是可逆的，那么我的信息不就被暴露了吗？

让我逐一为你说明。

### 签名的目的

最后一步签名的过程，实际上是对头部以及载荷内容进行签名。一般而言，加密算法对于不同的输入产生的输出总是不一样的。对于两个不同的输入，产生同样的输出的概率极其地小（有可能比我成世界首富的概率还小）。所以，我们就把“不一样的输入产生不一样的输出”当做必然事件来看待吧。

所以，如果有人对头部以及载荷的内容解码之后进行修改，再进行编码的话，那么新的头部和载荷的签名和之前的签名就将是不一样的。而且，如果不知道服务器加密的时候用的密钥的话，得出来的签名也一定会是不一样的。



服务器应用在接受到JWT后，会首先对头部和载荷的内容用同一算法再次签名。那么服务器应用是怎么知道我们用的是哪一种算法呢？别忘了，我们在JWT的头部中已经用alg字段指明了我们的加密算法了。

如果服务器应用对头部和载荷再次以同样方法签名之后发现，自己计算出来的签名和接受到的签名不一样，那么就说明这个Token的内容被别人动过的，我们应该拒绝这个Token，返回一个HTTP 401 Unauthorized响应。

### 信息会暴露？

是的。

所以，在JWT中，不应该在载荷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。

### JWT的适用场景

我们可以看到，JWT适合用于向Web应用传递一些非敏感信息。例如在上面提到的完成加好友的操作，还有诸如下订单的操作等等。

其实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。在下一次的文章中，我将为大家系统地总结JWT在用户认证和授权上的应用。如果想要及时地收到下一篇文章的更新，您可以在下方订阅我的半月刊：）