JWT介绍

一、什么是JWT

JSON Web Token(JWT)是一种基于JSON的开放标准(RFC 7519)，它定义了一种紧凑的、独立的的方式，用于在作为JSON对象的各方之间安全地传输信息。这些信息可以被验证和信任，因为它是数字签名的。JWTs可以使用一个秘密(与HMAC算法)或使用RSA的公钥/私钥对进行签名。

概念解释：

紧凑的：由于其较小，JWTs可以通过URL、POST参数或HTTP头部发送。此外，大小比较小也就意味着传输速度很快。

独立的：有效payload包含关于用户所需的所有信息，避免了多次查询数据库的需求。

二、什么时候使用JWT

下面是一些JWT有用的场景：

身份验证：这是使用JWT最常见的场景。用户登录后，每个后续请求将包括JWT，允许用户访问该token所允许的路由、服务和资源。单点登录是目前广泛使用JWT的一项功能，因为它的开销很小，而且在不同的域中很容易使用。

信息交换：JWT是在各方之间安全地传输信息的一种好方法。因为JWTs可以签署——例如，使用公共/私有密匙对——可以确定发送者是谁。此外，由于签名是使用头和有效负载进行计算的，您还可以验证内容没有被篡改。

三、JWT的结构是怎样的

JWT标记由由点(.)分隔的三个部分组成，即:

Header
Payload
Signature

因此，JWT通常看起来如下。

xxxxx.yyyyy.zzzzz

下面分别介绍下这三部分

Header

header通常由两个部分组成：token的类型，即JWT，以及加密的算法，通常直接使用例如HMAC SHA256或RSA。

例如：

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

然后，这个JSON是Base64Url编码，以形成JWT的第一部分。

Payload

JWT第二部分是负载，它包含权限的声明。权限是关于实体(通常是用户)和附加元数据的声明。有三种类型的权限:注册的声明、公开权限和私有权限。

标准中注册的声明：

iss: jwt签发者
sub: jwt所面向的用户
aud: 接收jwt的一方
exp: jwt的过期时间，这个过期时间必须要大于签发时间
nbf: 定义在什么时间之前，该jwt都是不可用的.
iat: jwt的签发时间
jti: jwt的唯一身份标识，主要用来作为一次性token,从而回避重放攻击。

公共的声明：

这些可以由使用JWTs的人来定义。但是为了避免冲突，应该在IANA JSON Web Token注册表中定义它们，或者定义为包含冲突命名空间的URI。

私有的声明 ：

私有声明是提供者和消费者所共同定义的声明，一般不建议存放敏感信息，因为base64是对称解密的，意味着该部分信息可以归类为明文信息。

一个payload 例子可以是:

```
{
 "sub": "1234567890",
 "name": "John Doe",
 "admin": true
}
```

payload是Base64Url编码，以形成JWT的第二部分。

Signature

要创建签名部分，您必须使用编码的header、编码的payload 、一个密钥、在头中指定的算法，并签署。 例如，如果您想使用HMAC SHA256算法，那么签名将以如下方式创建:

```
HMACSHA256(
 base64UrlEncode(header) + "." +
 base64UrlEncode(payload),
 secret)
```

签名用于验证JWT的发送者是谁，并确保消息在过程中不会被更改。

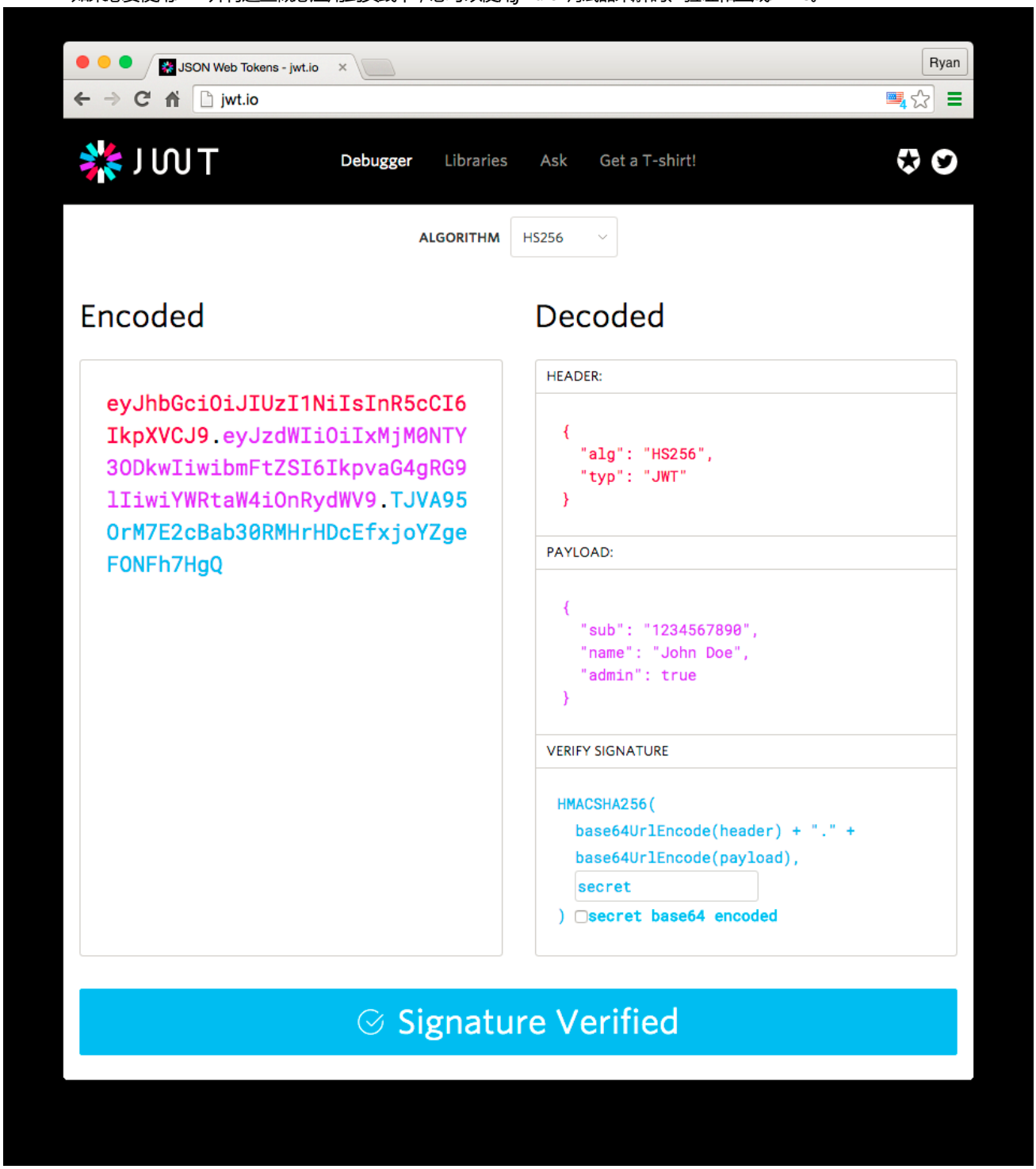
把三部分放在一起

输出是三个Base64字符串，这些字符串由点分隔开，可以很容易地在HTML和HTTP环境中传递，而与SAML等基于xml的标准相比，它们更紧凑。

下面显示了一个JWT，它有前面的header和payload编码，并且它是一个加盐的。

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaXNTb2NpYWwiOnRydWV9.
4pcPyMD09olPSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```

如果想要使用JWT并将这些概念应用到实践中，您可以使用jwt.io 调试器来解码、验证和生成JWTs。



四、JWT如何工作

在身份验证中，当用户成功登录使用他们的凭证时，将返回一个JWT，并且必须在本地保存(通常在本地存储中，但也可以使用

cookie)，而不是传统方法在服务器中创建会话并返回cookie。

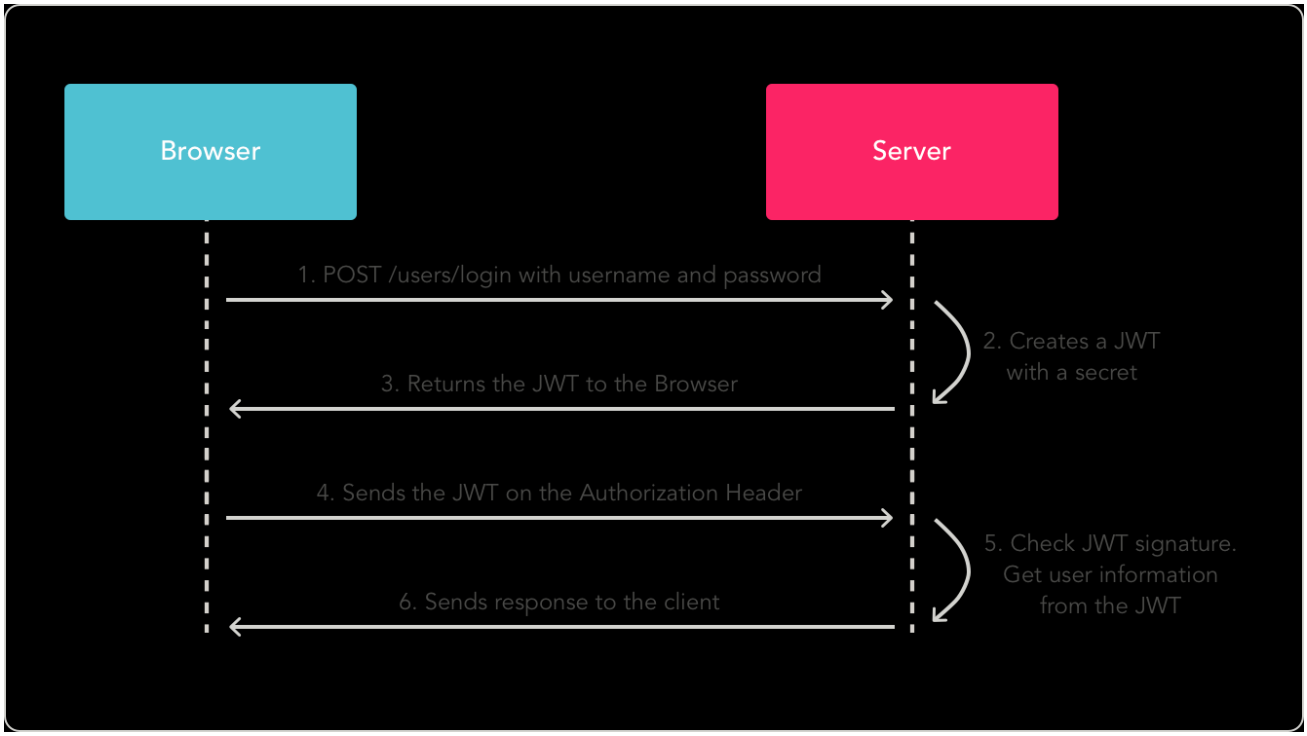
当用户想要访问受保护的路由或资源时，用户代理应该发送JWT，通常是使用无记名模式的授权头。标题的内容应该如下:

```
Authorization: Bearer <token>
```

这是一个无状态的身份验证机制，因为用户状态永远不会保存在服务器内存中。服务器的受保护路由将在授权标头中检查一个有效的JWT，如果存在，则允许用户访问受保护的资源。由于JWTs是自包含的，所有必要的信息都在那里，从而减少了对数据库进行多次查询的需要。

这允许您完全依赖无状态的数据api，甚至向下游服务发出请求。不管哪个域为您的api提供服务，所以跨源资源共享(CORS)不会是一个问题，因为它不使用cookie。

下图显示了这个过程:



https://jwt.io/
https://github.com/jwtk/jjwt
https://stormpath.com/blog/jwt-java-create-verify

https://stormpath.com/blog/jwt-java-create-verify
https://stormpath.com/blog/where-to-store-your-jwts-cookies-vs-html5-web-storage