Token Authorization and Validation for Site Services

Derek Weitzel - University of Nebraska

This work was partially supported by the NSF grants 2030508 (PATh), and 1836650 (IRIS-HEP), 2114989 (SciAuth)
What are tokens?

Tokens you might see are in OSG/WLCG:

1. SciTokens (https://scitokens.org)
2. WLCG Tokens (https://doi.org/10.5281/zenodo.3460258)
3. Macaroons - Not covered here (http://macaroons.io/)
What are tokens?

SciTokens and WLCG tokens are both based on JSON Web Tokens

<table>
<thead>
<tr>
<th>Decoded</th>
<th>Encoded</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HEADER: ALGORITHM &amp; TOKEN TYPE</strong></td>
<td></td>
</tr>
<tr>
<td>```json</td>
<td></td>
</tr>
<tr>
<td>`{</td>
<td></td>
</tr>
<tr>
<td>&quot;typ&quot;: &quot;JWT&quot;,</td>
<td></td>
</tr>
<tr>
<td>&quot;alg&quot;: &quot;ES256&quot;,</td>
<td></td>
</tr>
<tr>
<td>&quot;kid&quot;: &quot;key-es256&quot;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>```</td>
<td></td>
</tr>
<tr>
<td><strong>PAYLOAD: DATA</strong></td>
<td></td>
</tr>
<tr>
<td>```json</td>
<td></td>
</tr>
<tr>
<td>`{</td>
<td></td>
</tr>
<tr>
<td>&quot;scope&quot;: &quot;read:/protected&quot;,</td>
<td></td>
</tr>
<tr>
<td>&quot;aud&quot;: &quot;<a href="https://demo.scitokens.org">https://demo.scitokens.org</a>&quot;,</td>
<td></td>
</tr>
<tr>
<td>&quot;ver&quot;: &quot;scitoken:2.0&quot;,</td>
<td></td>
</tr>
<tr>
<td>&quot;iss&quot;: &quot;<a href="https://demo.scitokens.org">https://demo.scitokens.org</a>&quot;,</td>
<td></td>
</tr>
<tr>
<td>&quot;exp&quot;: 1634154486,</td>
<td></td>
</tr>
<tr>
<td>&quot;iat&quot;: 1634153886,</td>
<td></td>
</tr>
<tr>
<td>&quot;nbf&quot;: 1634153886,</td>
<td></td>
</tr>
<tr>
<td>&quot;jti&quot;: &quot;0ea13bb9-40e0-4340-81ef-25f7d96c9cef&quot;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>```</td>
<td><code>eyJ0eXAiOiJKV1QiLCJhbGciOiJFUzI1NiIsImtpZCI6ImtleS1lcI1NiJ9.eyJJYyZ29wZSI6InJyYWQ6L3Byb3RlY3RlZCI6IkZCI6Imh0dHBzOi8vZGVtby5zY2loYWtlbnM3JnIiwidmVyIjoic2NpG9rZW46Mi4wiwiXNzIjoiaHR0cHM6Ly9kZW1vLnNjaXRva2Vucy5vcmiLCJlJieHAiOiE2MzQxNTQ0ODYsIm1hdCI6MTUzNDE1Mzg4NiwiYmJmIjoXNjM0MTUzODg2LCJqdGkiOiIwZWE=MTjOS00MGUwLTQzNDAtODFjZi0yNWY3ZDKyZi1jZYifQ.wgyCKsTauKhsRGP2WqYJifrbYc5AGsmYIK743yaHuX7x1VhQU5Y001x6oJodIIf8kVcJNFO681G2aFRsboFg</code></td>
</tr>
</tbody>
</table>
What are tokens?

Tokens have 3 components, HEADER, PAYLOAD, and SIGNATURE

Decoded

```
{  
  "typ": "JWT",
  "alg": "ES256",
  "kid": "key-es256"
}
```

Encoded

eyJ0eXAiOiJKV1QiLCJhbGciOiJFUzI1NiIsImtpZCI6ImtleS1lcI1NiJ9.eyJJYWNvbGlQYW5rZW46Mi4wIiwiaXNzIjoiaHR0cHM6Ly9kZW1vLnNjaXRva2Vu5vcmciLCJleHAiOiJEMzQxNTQ0ODYsImI6ci5CI6MTYzNDE1Mzg4NiwibmJmIjoiNjM0MTUzODg2LCJqdGkiOiIwZWExM2JiOS00MGUwLTQzNDAtODFlZi0yNjIzZDk2Y2JiZWNyZWF0aW9ucyJ9.wgyCKsTaKhsRGpK2WqYJifrbYc5AGsmYIK743yaHuX7xlVhQUs5Y001x6oJodIIf8kVcJNFO681G2aFRsboFg
What are tokens?

Header gives very basic information about the token, including signature algorithm.
What are tokens?

Payload gives all of the really important information for authorization

Decoded

```json
{
    "typ": "JWT",
    "alg": "ES256",
    "kid": "key-es256"
}
```

Encoded

```plaintext
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6ImtleS1lczI1NiJ9.eyJzY29wZSI6InJlYWQ6L3Byb3R1Y3RlZCIsImF1ZCI6Imh0dHBzOi8vZGVtby5zY2l0b2tlbnMub3JnIiwidmVyIjoic2NpdG9rZW46Mi4wIiwiaXNzIjoiaHR0cHM6Ly9kZW1vLnNjaXRva2VuY2cvcmciLCJleHAiOiJEMzQxMjU2MTMwNzE5MDE1MzIwNTQ4MTg4ODE0MjA3OnByZWFkIiwicmVxdWVzdF9kYyI6ImF1ZCI6NzczOS4wMDA1MjIwNiIsInVzZXJfaXR5IjoxODkyNTg4NDU2LCJpc3MiOiJodHRwOi8vY29tLmNvbW9kZXN0ZXN0cy5jb20iLCJzdWtlIjoxODkyNTg4NDU2LCJpYXQiOjE2MDQ4ODExOTUsImNvbnBlcklkIjoyMjQiLCJ0eXBlIjoidW4iLCJleHRvZCI6MzA4OCwiaWF0IjoiMTY4NzkyMjQwNzA3ODg2IiwiYXV0aF9zdWIiOiJnYWxscGxhY2UifQ.wgyCKsTauKhsRGpK2WqYJifrbYc5AGsmYIK743yaHuX7x1VhQUs5Y001x6oJodIJf8kVcJNFO681G2aFRsboFg
```
What are tokens?

Encoded JSON in base64

**Encoded**

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6ImtleS1lcI1NiJ9.eyJzY29wZSI6ImJ1YWQ6L3Byb3RlY3RlZCI6ImF1ZCI6ImNh0dHBzOi8vZGVtby5zY2l0b2tlbnMub3JnIiwiXGQiOiJib29yZCJ9

**Decoded**

```
{
  "typ": "JWT",
  "alg": "ES256",
  "kid": "key-es256"
}
```

**PAYLOAD: DATA**

```
{
  "scope": "read:protected",
  "aud": "https://demo.scitokens.org",
  "ver": "scitoken:2.0",
  "iss": "https://demo.scitokens.org",
  "exp": 1634154486,
  "iat": 1634153886,
  "nbf": 1634153886,
  "jti": "0ea13bb9-4e00-4340-81ef-25f7d96c9ce9"
}
```
Payload Attributes

“scope”: Permissions given to the bearer of the token (more later)

“ver” (version): The token profile that this is following. Profiles have different validation rules.

“aud” (audience): What service is this token meant for. The issuer uses this to restrict where this token can be used. Has a special case of “ANY”.

“iss” (issuer): What service created this token. A service will trust an “issuer”.

```json
{
    "scope": "read:/protected",
    "aud": "https://demo.scitokens.org",
    "ver": "scitoken:2.0",
    "iss": "https://demo.scitokens.org",
    "exp": 1634154486,
    "iat": 1634153886,
    "nbf": 1634153886,
    "jti": "0ea13bb9-40e0-4340-81ef-25f7d96c9cef"
}
```
Payload Attributes

“exp” (expiration): Unix epoch that the token expires at.

“iat” (issued at): Unix epoch that the token was issued.

“nbf” (not before): Unix epoch that the token is not valid before.

“jti” (JWT ID): A unique identifier for this token.

```
{
    "scope": "read:/protected",
    "aud": "https://demo.scitokens.org",
    "ver": "scitoken:2.0",
    "iss": "https://demo.scitokens.org",
    "exp": 1634154486,
    "iat": 1634153886,
    "nbf": 1634153886,
    "jti": "0ea13bb9-40e0-4340-81ef-25f7d96c9cef"
}
```
What are tokens?

WLCG Tokens have a few different attributes

<table>
<thead>
<tr>
<th>Decoded</th>
<th>Encoded</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HEADER: ALGORITHM &amp; TOKEN TYPE</strong></td>
<td><strong>Encoded</strong></td>
</tr>
</tbody>
</table>
| ```json
{
    "kid": "rsa1",
    "alg": "RS256"
}
``` | ```
{"kid": "rsa1",
"alg": "RS256"
}
``` |
| **PAYLOAD: DATA** | |
# Payload Attributes

**"wlcg.ver"**: Same as "ver", but specifically the WLCG profile.

**"sub"** (subject): An identifier of the user that authenticated and should have received this token. Optional in SciTokens.

**"client_id"**: Client that requested this token for traceability.

**"scope"**: Different format than SciTokens. More later!

```json
{
    "wlcg.ver": "1.0",
    "sub": "8b9b75cf-7389-473a-b9e9-92025f865aa5",
    "aud": "https://wlcg.cern.ch/jwt/v1/any",
    "nbf": 1634150753,
    "scope": "storage.read:/",
    "iss": "https://wlcg.cloud.cnaf.infn.it/",
    "exp": 1634154353,
    "iat": 1634150753,
    "jti": "abf6d3d2-5172-4d31-ba59-009ee6fe5e69",
    "client_id": "218bcb57-a55e-4c96-a577-8d03a6dc3f72"
}
```
Differences between WLCG and SciTokens

WLCG covers three use cases:

1. Identity Tokens with groups
2. Access Tokens with groups
3. **Access Tokens with capabilities**

SciTokens only implements #3
## Quick tool to view tokens!

I generated all those views with [https://demo.scitokens.org](https://demo.scitokens.org), which is a modified copy of [https://jwt.io](https://jwt.io).

### Decoded

**HEADER: ALGORITHM & TOKEN TYPE**

```json
{
    "kid": "rsa1",
    "alg": "RS256"
}
```

**PAYLOAD: DATA**

```json
{
    "wlcg.ver": "1.0",
    "sub": "8b9b75cf-7389-473a-b9e9-92025f865aa5",
    "aud": "https://wlcg.cern.ch/jwt/v1/any",
    "nbf": 1634150753,
    "scope": "storage.read:",
    "iss": "https://wlcg.cloud.cern.ch/",
    "exp": 1634154533,
    "iat": 1634150753,
    "jti": "abf6d3d2-5172-4d31-ba59-0899e6fe5e69",
    "client_id": "218bcb57-a55e-4e96-a577-8d03a6dc3f72"
}
```

### Encoded

```
bGNLnZ1ciI6JEvMCtIsInN1YiI6Ijh1O0l3N0NnMlTC
zODktNDCzYS1iOUW5LTkyMDI1Zjg2NWFhNIXlISmF1ZC
I6Imh8dHBzOiJvXC93bGNNlMlNcm4uY2hcL2p3dFwvd
jFclL2FueSISim5iI6MTYzNDE1MDc1Mywic2NvcGUi
OiJzdG9yYWdlLnJ1YXQ6XC8iLCJpc3MiOiJodHRwczp
cl1wvd2xjZy5jbG91ZCSj5jbmFmLm1uZm4uaXRcLyIsIm
V4cCI6MTYzNDE1NjMxMywiaWF0IjoxNjM0MTUwNzUzLC
CjQdGkiOiJhYmY2ZDNkMi81MTEyLTrkMzEYmE1OS0wMD
I1ZTznMTV1NjkiLCJbGlbnRfQioiIyMTHiY2I1Ny7hNTV1LTRjOTYtYTU3Ny04ZDAzYTZkYzNmNzIiFQ
.HxZSKgT2aEHyQWA-CaM_856qYja0gIKESRxbC2eLxgGbRIOBoguB1urYBCc2
oiEtgfmjUYzfnn95liaI5AR6f306k1pfcZZxR0jPlbC
dduGxyc_CWPY1mkXyXW33MS-B_GkddC1qVWML2NhChTLIHvLP7fR0zuWgRmkqv4z
_E
```
The ANY audience!

Audience is designed to stop malicious actors that steal the token from using it anywhere but the targeted audience.

But... In some cases, you want the token to be allowed at MANY places that are impossible to predict, for example a caching infrastructure (StashCache)

Both token types have an idea of tokens that can work everywhere!

SciTokens: “aud”: “ANY”

How libraries validate tokens

Storage service will contact the issuer to download the public key

Validate the token signature with the public key

```
{
  "keys": [
    {
      "crv": "P-256",
      "kid": "6804",
      "kty": "EC",
      "use": "sig",
      "x": "Svjur4JHtpmdx5w6dWVuja_tKpqZ4JQzmo9juVlWNQ=",
      "y": "NHTw__1jkLVwhQ-mRIRic9DF5lIBSHqpbVwpAJUQQxQ="
    }
  ]
}
```
Demo

https://sciauth.org/notebook-demo
CI Logon demo

We will generate a token using CI Logon

Each issuer has their own policies, CI Logon issuer will have different policies than WLCG.

OIDC-Agent steps

Install: https://indigo-dc.gitbook.io/oidc-agent/

$ eval $(oidc-agent)
$ oidc-gen -w device -m cilogon
Issuer: https://test.cilogon.org/
Client_id: cilogon:/weitzel/demo
Client secret is anything you want, it's ignored.
Just hit enter when asking for scopes
Hit enter when asks for redirect_uris
Enters through the encryption

$ oidc-token cilogon
Look at the token in demo.scitokens.org
Then try curl on demo.scitokens.org:
OIDC Renewal

If you have a service that requires a frequent updated tokens, OSG created a service for that.

OSG Token Renewer:
https://github.com/opensciencegrid/osg-token-renewer
XRootD Configuration

[Global]

[Issuer CMS]
issuer = https://scitokens.org/cms
base_path = /user/uscms01/pnfs/unl.edu/data4/cms
# For CMS, there is no relationship between local usernames
# and the VO name.
map_subject = False
default_user = cmsprod004

Audience that the server looks for is set at the top

For CMS, the allowed issuer is https://scitokens.org/cms

base_path is the base of which the permissions are allowed. Therefore, read:/ in the token really means read:/user/uscms01/pnfs/unl.edu/data4/cms

The default_user is the mapped user for this issuer. For HDFS, it is the owner of the files written.
Debugging XRootD + Tokens

The logs are actually pretty good!

Audience is not set correctly on the token:

```
```