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

Decoded EDIT THE PAYLOAD

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "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": "0ea13bb9-40e0-4340-81ef-25f7d96c9cef"
}
```

Encoded

eyJ0eXAiOiJKV1QiLCJhbGciOiJFUzI1NiIsImtpZCI
6ImtleS1lczI1NiJ9.eyJzY29wZSI6InJlYWQ6L3Byb
3RlY3RlZCIsImF1ZCI6Imh0dHBzOi8vZGVtby5zY2l0
b2tlbnMub3JnIiwidmVyIjoic2NpdG9rZW46Mi4wIiw
iaXNzIjoiaHR0cHM6Ly9kZW1vLnNjaXRva2Vucy5vcm
ciLCJleHAiOjE2MzQxNTQ0ODYsImlhdCI6MTYzNDE1M
zg4NiwibmJmIjoxNjM0MTUzODg2LCJqdGkiOiIwZWEx
M2JiOS00MGUwLTQzNDAtODFlZi0yNWY3ZDk2YzljZWY
ifQ.wgyCKsTauKhsRGpK2WqYJifrbYc5AGsmYIK743y
aHuX7xlVhQUs5YOO1x6oJodIJf8kVcJNFO681G2aFRs
boFg

What are tokens?

Tokens have 3 components, HEADER, PAYLOAD, and SIGNATURE

Decoded EDIT THE PAYLOAD

HEADER: ALGORITHM & TOKEN TYPE

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

PAYLOAD: DATA

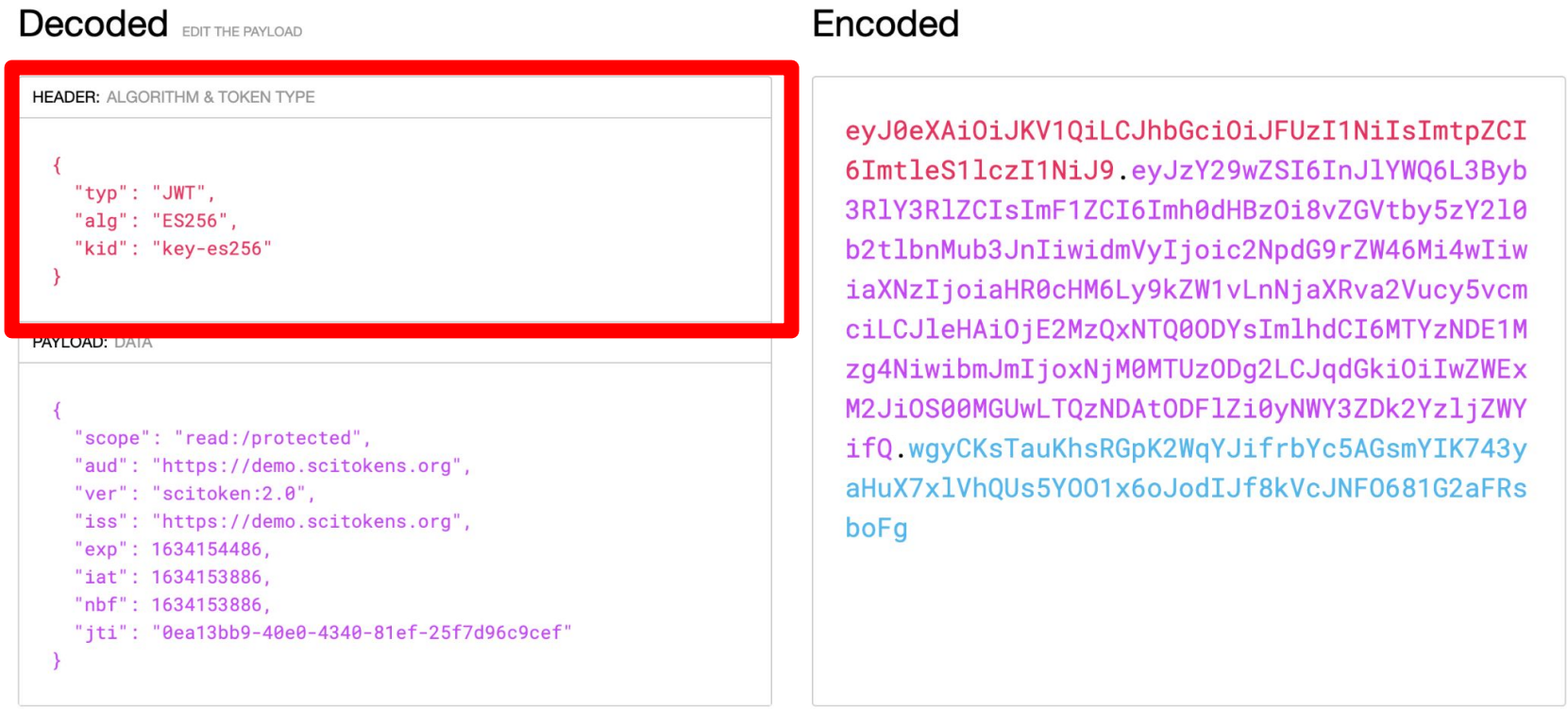
```
{
  "scope": "read:/protected",
  "aud": "https://demo.scitokens.org",
  "ver": "scitoken:2.0",
  "iss": "https://demo.scitokens.org",
  "exp": 1634154486,
  "iat": 1634153886,
  "nbf": 1634153886,
  "jti": "0ea13bb9-40e0-4340-81ef-25f7d96c9cef"
}
```

Encoded

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJFUzI1NiIsImtpZCI
6ImtleS1lczI1NiJ9.eyJzY29wZSI6InJlYWQ6L3Byb
3RlY3RlZCIsImF1ZCI6Imh0dHBzOi8vZGVtby5zY2l0
b2tlbnMub3JnIiwidmVyIjoic2NpdG9rZW46Mi4wIiw
iaXNzIjoiaHR0cHM6Ly9kZW1vLnNjaXRva2Vucy5vcm
ciLCJleHAiOjE2MzQxNTQ0ODYsImlhdCI6MTYzNDE1M
zg4NiwibmJmIjoxNjM0MTUzODg2LCJqdGkiOiIwZWEx
M2JiOS00MGUwLTQzNDAtODFlZi0yNWY3ZDk2YzljZWY
ifQ.wgyCKsTauKhsRGpK2WqYJifrbYc5AGsmYIK743y
aHuX7xlVhQUs5YOO1x6oJodIJf8kVcJNFO681G2aFRs
boFg
```

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 EDIT THE PAYLOAD

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "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": "0ea13bb9-40e0-4340-81ef-25f7d96c9cef"
}
```

Encoded

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJFUzI1NiIsImtpZCI
6ImtleS1lczI1NiJ9.eyJzY29wZSI6InJlYWQ6L3Byb
3RlY3RlZCIsImF1ZCI6Imh0dHBzOi8vZGVtby5zY2l0
b2tlbnMub3JnIiwidmVyIjoic2NpdG9rZW46Mi4wIiw
iaXNzIjoiaHR0cHM6Ly9kZW1vLnNjaXRva2Vucy5vcm
ciLCJleHAiOjE2MzQxNTQ0ODYsImlhdCI6MTYzNDE1M
zg4NiwibmJmIjoxNjM0MTUzODg2LCJqdGkiOiIwZWEx
M2JiOS00MGUwLTQzNDAtODFlZi0yNWY3ZDk2YzljZWY
ifQ.wgyCKsTauKhsRGpK2WqYJifrbYc5AGsmYIK743y
aHuX7xlVhQUs5YOO1x6oJodIJf8kVcJNFO681G2aFRs
boFg
```

What are tokens?

Encoded JSON in base64

Decoded EDIT THE PAYLOAD

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "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": "0ea13bb9-40e0-4340-81ef-25f7d96c9cef"
}
```

Encoded

eyJ0eXAiOiJKV1QiLCJhbGciOiJFUzI1NiIsImtpZCI
6ImtleS1lczI1NiJ9.eyJzY29wZSI6InJlYWQ6L3Byb
3RlY3RlZCIsImF1ZCI6Imh0dHBzOi8vZGVtby5zY2l0
b2tlbnMub3JnIiwidmVyIjoic2NpdG9rZW46Mi4wIiw
iaXNzIjoiaHR0cHM6Ly9kZW1vLnNjaXRva2Vucy5vcm
ciLCJleHAiOjE2MzQxNTQ0ODYsImlhdCI6MTYzNDE1M
zg4NiwibmJmIjoxNjM0MTUzODg2LCJqdGkiOiIwZWEx
M2JiOS00MGUwLTQzNDAtODFlZi0yNWY3ZDk2YzljZWY
ifQ.wgyCKsTauKhsRGpK2WqYJifrbYc5AGsmYIK743y
aHuX7xlVhQUs5YOO1x6oJodIJf8kVcJNFO681G2aFRs
boFg

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”.


```
{
  "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.

“**jit**” (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

Decoded EDIT THE PAYLOAD

HEADER: ALGORITHM & TOKEN TYPE

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

PAYLOAD: DATA

```
{
  "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"
}
```

Encoded


```
bGNnLnZlciI6IjEuMCIsInN1YiI6IjhiOWI3NWNmLTc
zODktNDczYS1iOWU5LTkyMDI1Zjg2NWFhNSIsImF1ZC
I6Imh0dHBzOlwvXC93bGNnLmNlcm4uY2hcL2p3dFwvd
jFcL2FueSIsIm5iZiI6MTYzNDE1MDc1Mywic2NvcGUi
OiJzdG9yYWdlLnJlYWQ6XC8iLCJpc3MiOiJodHRwczp
cL1wvd2xjZy5jbG91ZC5jbmFmLmluZm4uaXRcLyIsIm
V4cCI6MTYzNDE1NDM1MywiaWF0IjoxNjM0MTUwNzUzL
CJqdGkiOiJhYmY2ZDNkMi01MTcyLTRkMzEtYmE1OS0w
MDllZTZmZTVlNjkiLCJjbGllbnRfaWQiOiIyMThiY2I
1Ny1hNTVlLTRjOTYtYTU3Ny04ZDAzYTZkYzNmNzIifQ
.HxZSKgtI2aEHvQWA-
CaM_856qYjaOgIkESRxbC2eLxgGbRIBOguD1urYBCc2
oiEtgfmJyAzfn95liaI5AR6F3O6kIpfcZZdxR0jPlbC
dduGxgc_CWPYimKYxW33MS-
B_GkdDClqVWMZL2NhChTLIHvLP7fROZuwVgRMkqjv4z
_E
```

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!


```
{
  "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

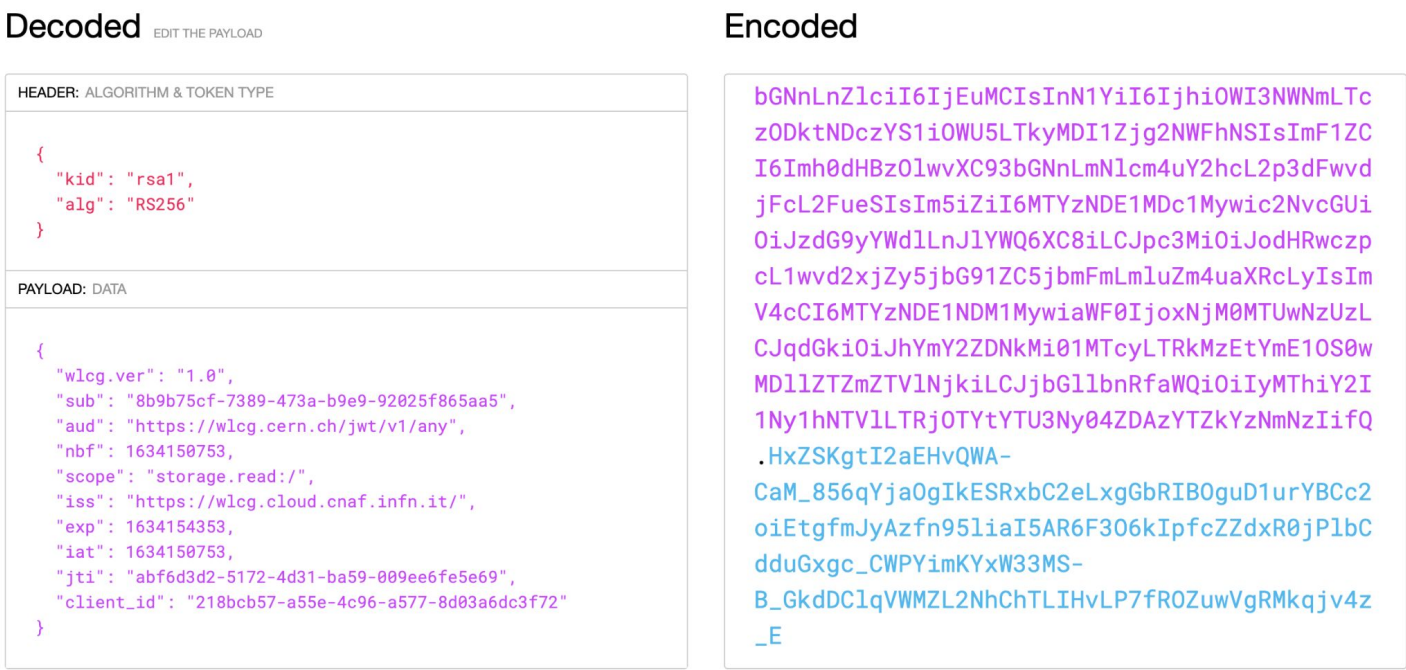
WLCG covers three use cases:

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, which is a modified copy of https://jwt.io.



The ANY audience!

Audience is designed 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”

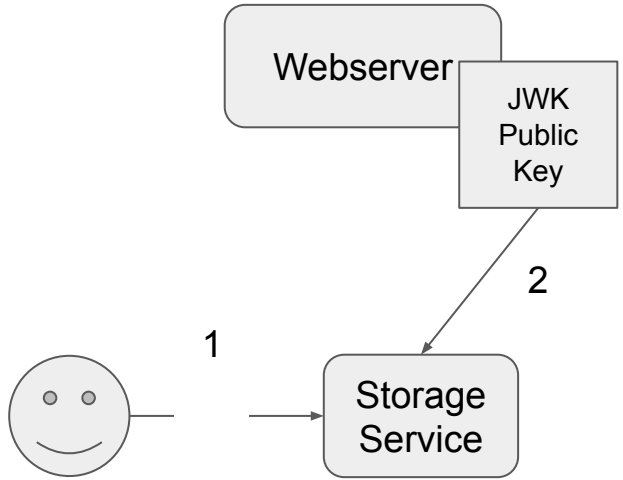
WLCG Tokens: “aud”: “https://wlcg.cern.ch/jwt/v1/any”

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": "Svjur4JHtjpmdx5w6dWVuja_tKpqZ4JQzmo9juVlWNQ=",
           "y": "NHTw__ljkLVwHQ-mRIRic9DF5lIBSHqpbVwpAJUQ0xQ="
       }
   ]
}
```

Demo

https://sciauth.org/notebook-demo

CILogon demo

We will generate a token using CILogon

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

Will use oidc-agent: https://indigo-dc.gitbook.io/oidc-agent/

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:

```
$ curl -H "Authorization: Bearer <token>" https://demo.scitokens.org/protected
```

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]
audience = https://red-gridftp4.unl.edu:1094, https://xrootd.unl.edu:1094, LIGO

[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:

```
211014 19:25:42 15492 scitokens_GenerateAcls: ACL generation from SciToken failed: token verification failed:
'aud' claim verification failed.
```