swt

Jun 03, 2020

Contents:

1 Using RSA SHA256 3

2 API Reference 5

3 Indices and tables 7

Python Module Index 9

Index 11

To use this module, simply implement and use your own algorithm specific SWT class.

More[1] info about *Simple Web Tokens* can be found at Microsoft who originated the spec. back in 2009.

[1] Although not a lot

CHAPTER 1

Using RSA SHA256

Create your own token class that extends from `SWT_RSA_SHA256`, and implement the key locators you need.

If you only want to verify tokens, then you can skip implementing the `get_private_key()` method.

```
class MySWT(SWT_RSA_SHA256):

    def get_public_key(self):
        return Path(f'/keys/{self.issuer}-public.pem').read_text()

    def get_private_key(self):
        return Path(f'/keys/{self.issuer}-private.pem').read_text()
```

Creating token objects from existing token strings, e.g. directly from http headers or similar.

```
# You can pass a full bearer token value directly from the request header
# no need to strip out the Bearer part first
token = MySWT(http_header_value)

if token.is_valid:
    # Token has both a valid signature and is not expired

if token.is_signed:
    # Token has a valid signature

if token.is_expired:
    # Token is signed and expired, or token is not signed
    # We only trust data in the token if it is signed
```

Creating and signing new tokens

```
# Create token
token = MySWT()

# Set issuer, you must have a key locator that can find the private key based
# upon the issuer
```

(continues on next page)

(continued from previous page)

```
token.issuer = 'my-issuer'

# Set time to live
token.ttl = 3600

# Set claims
token.set_claim('sub', 42)
token.set_claim('foo', 'bar')

# Sign token with private key, and get serialized token back
token_str = token.sign()

# You can also get the serialized token from the token objectt by accessing
# the token_str property
token_str = token.token_str
```

As a convenience, if you use the token object in string or boolean context it will do *the right thing*™.

```
# Token in string context gives you the serialized token
token_str = str(token)

# Token in boolean context gives you the validity of the token
is_valid = bool(token)

# Parse token from string, and do stuff with it, if it is valid
token = MySWT(http_header_value)
if token:
    # Do stuff with the valid token
```

CHAPTER 2

API Reference

class swt.**SWT**(*token_str: Optional[str] = None*)

Simple Web Token base class

To use this library, you must choose which algorithm you want to use, and extend the algorithm specific sub class. Currently only RSA SHA256 is implemented.

algorithm

The algorithm used for the SWT

get_private_key() → str

Implement this in your own subclass to find and load the private key by issuer

get_public_key() → str

Implement this in your own subclass to find and load the public key by issuer

is_expired

Check if the SWT is expired

Returns is expired

Return type bool

is_signed

Algorithm specific is_signed() property

Returns signed status

Return type bool

is_valid

Check if the SWT is both sign and not expired

Returns token validity

Return type bool

issuer

Issuer of token

sign() → str
Algorithm specific sign() method

Returns signed token

Return type str

token_str
Token as serialized string

ttl
Time to live in seconds

class swt.**SWT_HMAC_SHA256**(*token_str: Optional[str] = None*)
Not yet implemented

algorithm
The algorithm used for the SWT

is_signed
Algorithm specific is_signed() property

Returns signed status

Return type bool

sign()
Algorithm specific sign() method

Returns signed token

Return type str

class swt.**SWT_RSA_SHA256**(*token_str: Optional[str] = None*)
SWT using RSA and SHA256

Extend this class and implement the key locater methods

algorithm
The algorithm used for the SWT

is_signed
Algorithm specific is_signed() property

Returns signed status

Return type bool

sign()
Algorithm specific sign() method

Returns signed token

Return type str

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

swt, 5

Index

A

`algorithm` (*swt.SWT attribute*), 5
`algorithm` (*swt.SWT_HMAC_SHA256 attribute*), 6
`algorithm` (*swt.SWT_RSA_SHA256 attribute*), 6

G

`get_private_key()` (*swt.SWT method*), 5
`get_public_key()` (*swt.SWT method*), 5

I

`is_expired` (*swt.SWT attribute*), 5
`is_signed` (*swt.SWT attribute*), 5
`is_signed` (*swt.SWT_HMAC_SHA256 attribute*), 6
`is_signed` (*swt.SWT_RSA_SHA256 attribute*), 6
`is_valid` (*swt.SWT attribute*), 5
`issuer` (*swt.SWT attribute*), 5

S

`sign()` (*swt.SWT method*), 5
`sign()` (*swt.SWT_HMAC_SHA256 method*), 6
`sign()` (*swt.SWT_RSA_SHA256 method*), 6
`SWT` (*class in swt*), 5
`swt` (*module*), 5
`SWT_HMAC_SHA256` (*class in swt*), 6
`SWT_RSA_SHA256` (*class in swt*), 6

T

`token_str` (*swt.SWT attribute*), 6
`ttl` (*swt.SWT attribute*), 6