python-saml Documentation

Release 0.9.0

Ryan Leckey

June 19, 2015

Contents

A python interface to produce and consume Security Assertion Markup Language (SAML) v2.0 messages.

See: https://www.oasis-open.org/standards#samlv2.0

Contents:

CHAPTER 1

Installation

1.1 Supported platforms

- Python 2.7
- Python 3.3
- Python 3.4

1.2 Dependencies

In order to sign and verify signatures, *libxml2* and *libxmlsec* are required.

Linux

```
apt-get install libxml2-dev libxmlsec1-dev
```

Mac

```
brew install libxml2 libxmlsec1
```

1.3 Installing an official release

The most recent release is available from PyPI

```
pip install saml
```

1.4 Installing the development version

1. Clone the **python-saml** repository

```
git clone git://github.com/mehcode/python-saml.git
```

2. Change into the project directory

```
cd python-saml
```

3. Install the project and all its dependencies using *pip*

```
pip install .
```

CHAPTER 2

Creating Documents

Create XML documents in accordance with the SAML 2.0 specification

2.1 AuthnRequest

class saml.schema.**AuthenticationRequest**(*text=None*, ***kwargs*)

Create a SAML AuthnRequest

```
from saml import schema
from datetime import datetime

document = schema.AuthenticationRequest()
document.id = '11111111-2222-3333-4444-555555555555'
document.issue_instant = datetime(2000, 1, 1)
document.assertion_consumer_service_index = 0
document.attribute_consuming_service_index = 0
document.issuer = 'https://sp.example.com/SAML2'

policy = schema.NameIDPolicy()
policy.allow_create = True
policy.format = schema.NameID.Format.TRANSIENT
document.policy = policy

print document.tostring()
```

Produces the following XML document:

```
<samlp:AuthnRequest
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    Version="2.0"
    ID="11111111-2222-3333-4444-555555555555"
    IssueInstant="2000-01-01T00:00:00Z"
    AssertionConsumerServiceIndex="0"
    AttributeConsumingServiceIndex="0">
    <saml:Issuer>https://sp.example.com/SAML2</saml:Issuer>
    <samlp:NameIDPolicy
        Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient"
        AllowCreate="true"/>
</samlp:AuthnRequest>
```

2.2 Response

class saml.schema.**Response**(*text=None*, ***kwargs*)
Create a SAML Response

```
from saml import schema
from datetime import datetime

document = schema.Response()
document.id = '11111111-1111-1111-1111-111111111111'
document.in_response_to = '22222222-2222-2222-2222-222222222222'
document.issue_instant = datetime(2000, 1, 1, 1)
document.issuer = 'https://idp.example.org/SAML2'
document.destination = 'https://sp.example.com/SAML2/SSO/POST'
document.status.code.value = schema.StatusCode.SUCCESS

# Create an assertion for the response.
document.assertions = assertion = schema.Assertion()
assertion.id = '33333333-3333-3333-3333-333333333333'
assertion.issue_instant = datetime(2000, 1, 1, 2)
assertion.issuer = 'https://idp.example.org/SAML2'

# Create a subject.
assertion.subject = schema.Subject()
assertion.subject.principal = '44444444-4444-4444-4444-444444444444'
assertion.subject.principal.format = schema.NameID.Format.TRANSIENT
data = schema.SubjectConfirmationData()
data.in_response_to = '22222222-2222-2222-2222-222222222222'
data.not_on_or_after = datetime(2000, 1, 1, 1, 10)
data.recipient = 'https://sp.example.com/SAML2/SSO/POST'
confirmation = schema.SubjectConfirmation()
confirmation.data = data
assertion.subject.confirmation = confirmation

# Create an authentication statement.
statement = schema.AuthenticationStatement()
assertion.statements.append(statement)
statement.authn_instant = datetime(2000, 1, 1, 1, 3)
statement.session_index = '33333333-3333-3333-3333-333333333333'
reference = schema.AuthenticationContextReference
statement.context.reference = reference.PASSWORD_PROTECTED_TRANSPORT

# Create a authentication condition.
assertion.conditions = conditions = schema.Conditions()
conditions.not_before = datetime(2000, 1, 1, 1, 3)
conditions.not_on_or_after = datetime(2000, 1, 1, 1, 9)
condition = schema.AudienceRestriction()
condition.audiences = 'https://sp.example.com/SAML2'
conditions.condition = condition

print document.tostring()
```

Produces the following XML document:

```
<samlp:Response
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    Version="2.0"
```

```
        ID="11111111-1111-1111-1111-111111111111"
        IssueInstant="2000-01-01T01:00:00Z"
        Destination="https://sp.example.com/SAML2/SSO/POST"
        InResponseTo="22222222-2222-2222-2222-222222222222">
        <saml:Issuer>https://idp.example.org/SAML2</saml:Issuer>
        <samlp:Status>
            <samlp:StatusCode
                Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
        </samlp:Status>
        <saml:Assertion
            Version="2.0"
            ID="33333333-3333-3333-3333-333333333333"
            IssueInstant="2000-01-01T02:00:00Z">
            <saml:Issuer>https://idp.example.org/SAML2</saml:Issuer>
            <saml:Subject>
                <saml:NameID
                    Format=
                    "urn:oasis:names:tc:SAML:2.0:nameid-format:transient">
                    44444444-4444-4444-4444-444444444444
                </saml:NameID>
                <saml:SubjectConfirmation
                    Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
                    <saml:SubjectConfirmationData
                        NotOnOrAfter="2000-01-01T01:10:00Z"
                        Recipient="https://sp.example.com/SAML2/SSO/POST"
                        InResponseTo=
                        "22222222-2222-2222-2222-222222222222"/>
                </saml:SubjectConfirmation>
            </saml:Subject>
            <saml:Conditions
                NotBefore="2000-01-01T01:03:00Z"
                NotOnOrAfter="2000-01-01T01:09:00Z">
                <saml:AudienceRestriction>
                    <saml:Audience>
                        https://sp.example.com/SAML2
                    </saml:Audience>
                </saml:AudienceRestriction>
            </saml:Conditions>
            <saml:AuthnStatement
                AuthnInstant="2000-01-01T01:03:00Z"
                SessionIndex="33333333-3333-3333-3333-333333333333">
                <saml:AuthnContext>
                    <saml:AuthnContextClassRef>
                        urn:oasis:names:tc:SAML:2.0:ac:classes:
                        PasswordProtectedTransport
                    </saml:AuthnContextClassRef>
                </saml:AuthnContext>
            </saml:AuthnStatement>
        </saml:Assertion>
    </samlp:Response>
```

2.3 LogoutRequest

class saml.schema.**LogoutRequest**(*text=None*, ***kwargs*)
Create a SAML LogoutRequest

```
from saml import schema
from datetime import datetime

document = schema.LogoutRequest()
document.id = '11111111-1111-1111-1111-111111111111'
document.issue_instant = datetime(2000, 1, 1)
document.issuer = 'https://idp.example.org/SAML2'
document.destination = 'https://sp.example.org/SAML2/logout'
document.principal = 'myemail@mydomain.com'
document.principal.format = schema.NameID.Format.EMAIL
document.principal.name_qualifier = 'https://idp.example.org/SAML2'
document.session_index = 'SESSION-22222222-2222-2222-2222-222222222222'

print document.tostring()
```

Produces the following XML document:

```
<samlp:LogoutRequest
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    Version="2.0"
    ID="11111111-1111-1111-1111-111111111111"
    IssueInstant="2000-01-01T00:00:00Z"
    Destination="https://idphost/adfs/ls/">
    <saml:Issuer>https://idp.example.org/SAML2</saml:Issuer>
    <saml:NameID
        NameQualifier="https://idp.example.org/SAML2"
        Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
        myemail@mydomain.com
    </saml:NameID>
    <samlp:SessionIndex>
        SESSION-22222222-2222-2222-2222-222222222222
    </samlp:SessionIndex>
</samlp:LogoutRequest>
```

2.4 LogoutResponse

class saml.schema.**LogoutResponse**(*text=None*, ***kwargs*)
Create a SAML LogoutResponse

```
from saml import schema
from datetime import datetime

document = schema.LogoutResponse()
document.id = '22222222-2222-2222-2222-222222222222'
document.in_response_to = '11111111-1111-1111-1111-111111111111'
document.issue_instant = datetime(2000, 1, 1)
document.issuer = 'https://idp.example.org/SAML2'
document.destination = 'https://sp.example.com/SAML2/SLO/POST'
document.status.code.value = schema.StatusCode.SUCCESS

print document.tostring()
```

Produces the following XML document:

```
<samlp:LogoutResponse
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    Version="2.0"
    ID="22222222-2222-2222-2222-222222222222"
    IssueInstant="2000-01-01T00:00:00Z"
    Destination="https://sp.example.com/SAML2/SLO/POST"
    InResponseTo="11111111-1111-1111-1111-111111111111">
    <saml:Issuer>https://idp.example.org/SAML2</saml:Issuer>
    <samlp:Status>
        <samlp:StatusCode
            Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
    </samlp:Status>
</samlp:LogoutResponse>
```

CHAPTER 3

Signing Documents

Sign and verify signatures using the *python-xmlsec* library.

saml.signature.**sign**(*xml*, *stream*, *password=None*)

Sign an XML document with the given private key file. This will add a <Signature> element to the document.

Parameters

- **xml** (*lxml.etree._Element*) – The document to sign
- **stream** (*file*) – The private key to sign the document with
- **password** (*str*) – The password used to access the private key

Return type None

Example usage:

```
from saml import schema
from lxml import etree

document = schema.AuthenticationRequest()
xml_document = document.serialize()
with open('my_key_file.pem', 'r+') as stream:
    sign(xml_document, stream)

print etree.tostring(xml_document)
```

Produces the following XML document:

```
<samlp:AuthnRequest
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    Version="2.0" ID="_6087de0b111b44349a70ff40191a4c0c"
    IssueInstant="2015-03-16T21:06:39Z">
    <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
        <SignedInfo>
            <CanonicalizationMethod
                Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                <SignatureMethod
                    Algorithm="http://www.w3.org/2000/
                    09/xmldsig#rsa-sha1"/>
                    <Reference>
                        <Transforms>
                            <Transform
                                Algorithm="http://www.w3.org/2000/
                                09/xmldsig#enveloped-signature"/>
```

```
                            </Transforms>
                            <DigestMethod
                                Algorithm="http://www.w3.org/2000/
                                09/xmldsig#sha1"/>
                                <DigestValue>
                                    94O1FOjRE4JQYVDqStkYzne9StQ=
                                </DigestValue>
                        </Reference>
            </SignedInfo>
            <SignatureValue>
                aFYRRjtB3bDyLLJzLZmsn0K4SXmOpFYJ+8R8D31VojgiF37FOElbE56UFbm8BAjn
                l2AixrUGXP4djxoxxnfBD/reYw5yVuIVXlMxKec784nF2V4GyrfwJOKaNmlVPkq5
                c8SI+EkKJ02mwiail0Zvjb9FzwvlYD+osMSXvJXVqnGHQDVFlhwbBRRVB6t44/M3
                TzC4mLSVhuvcpsm4GTQSpGkHP7HvweKN/OTc0aTy8Kh/YUrImwnUCii+J0EW4nGg
                71eZyq/IiSPnTD09WDHsWe3g29kpicZXqrQCWeLE2zfVKtyxxs7PyEmodH19jXyz
                wh9hQ8t6PFO47Ros5aV0bw==
            </SignatureValue>
        </Signature>
    </samlp:AuthnRequest>
```

saml.signature.**verify**(*xml*, *stream*)

Verify the signaure of an XML document with the given certificate. Returns *True* if the document is signed with a valid signature. Returns *False* if the document is not signed or if the signature is invalid.

Parameters

- **xml** (*lxml.etree._Element*) – The document to sign
- **stream** (*file*) – The private key to sign the document with

Return type Boolean

CHAPTER 4

Contributing

4.1 Setting up your environment

1. Fork the repository
2. Clone your fork
3. Create a virtual environment.

4. Install **python-saml** in development mode with testing enabled. This will download all dependencies required for running the unit tests.

```
pip install -e ".[test]"
```

5. Make changes with tests and documentation
6. Open a pull request

4.2 Running the tests

Tests are run with *py.test*.

```
py.test --pep8 --flakes --cov saml
```

4.3 Testing documentation changes

Documentation is handled with Sphinx. Use the *make html* command in the *docs* directory to build an HTML preview of the documentation.

```
cd docs
make html
```

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

S

A

L

R

S

V