SpPythonTutorial

Simplest service provider AssertionConsumer ever as a WSGI application

The prerequisite for this example is:

- you must set the IdP signature and encryption keys in the idp_metata_xml string containing the IdP metadata file, you can replace it completely by the IdP metadata file if you have one,
- the AuthnResponse is transmitted using the HTTP-Post binding.

```
import sys
import lasso
from wsgiref.simple_server import make_server
import logging
import urlparse

logging.basicConfig(level=logging.DEBUG)

sp_metadata_xml = '''<?xml version="1.0"?>
<EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
      xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
      xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
      entityID="http://localhost:8081/metadata">
  <SPSSODescriptor
      AuthnRequestsSigned="true"
      protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">

    <AssertionConsumerService isDefault="true" index="0"
      Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
      Location="http://localhost:8081/singleSignOnPost" />
    <NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</NameIDFormat>
  </SPSSODescriptor>
  <Organization>
     <OrganizationName xml:lang="en">Example SAML 2.0 metadatas</OrganizationName>
  </Organization>
</EntityDescriptor>'''

idp_metadata_xml = '''<?xml version="1.0"?>
<EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    entityID="http://localhost:3001/saml/metadata">

  <IDPSSODescriptor
      WantAuthnRequestsSigned="true"
      protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
    <KeyDescriptor use="signing">
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <KeyValue  xmlns="http://www.w3.org/2000/09/xmldsig#">
    <RSAKeyValue>
        <Modulus>4yalpsp9Sxlsj07PEI8jJxhSJdo4F0iW0H8u1dhwmsW5YQvRUw/yPlmC09q4WjImmnFVNCJarAOYeFgQC
xfIoBasKNnUeBQpogo8W0Q/3mCuKl6lNSr/PIuxMVVNPDWmWkhHXJx/MVar2IREKa1P4jHL0Uxl69/idLwc7TtK1h8=</Modul
us>
        <Exponent>AQAB</Exponent>
    </RSAKeyValue>
</KeyValue>
      </ds:KeyInfo>
    </KeyDescriptor>
    <KeyDescriptor use="encryption">
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <KeyValue  xmlns="http://www.w3.org/2000/09/xmldsig#">
    <RSAKeyValue>
        <Modulus>wLu5SdmwyS4o1On/aw4nElLGERFG931exvkzu0ewaM1/oUyD3dO7UC5xMGnPfc6IaH5BcJc3fLr6PJhX5
```

```
5ZrMR98ToPwoUFwuLKK43exwYBEBOOMe1CrCB/Bq+EH6/2sKNXKfgJqj06/3yzafLRiWpMxy2isllxMAvaZXrkpm4c=</Modul
us>
        <Exponent>AQAB</Exponent>
    </RSAKeyValue>
</KeyValue>
      </ds:KeyInfo>
    </KeyDescriptor>
  </IDPSSODescriptor>

</EntityDescriptor>
'''

def app(environ, start_response):
    server = lasso.Server.newFromBuffers(sp_metadata_xml)
    server.addProviderFromBuffer(lasso.PROVIDER_ROLE_IDP, idp_metadata_xml)
    login = lasso.Login(server)
    try:
        data = environ['wsgi.input'].read(int(environ['CONTENT_LENGTH']))
        qs = urlparse.parse_qs(data)
        try:
            login.processAuthnResponseMsg(qs['SAMLResponse'][0])
        except (lasso.DsError, lasso.ProfileCannotVerifySignatureError):
            raise Exception('Invalid signature')
        except lasso.Error:
            raise Exception('Misc error')
        try:
            login.acceptSso()
        except lasso.Error:
            raise Exception('Invalid assertion')
    except Exception, e:
        start_response('500 Internal Error', [('content-type', 'text/plain')],
            sys.exc_info())
        return ['Erreur: ', str(e)]
    else:
        start_response('200 Ok', [('content-type', 'text/plain')], sys.exc_info())
        return ['You are identified as ', login.assertion.subject.nameId.content]

s = make_server('0.0.0.0', 8081, app)
s.serve_forever()
```