SAML & SSO

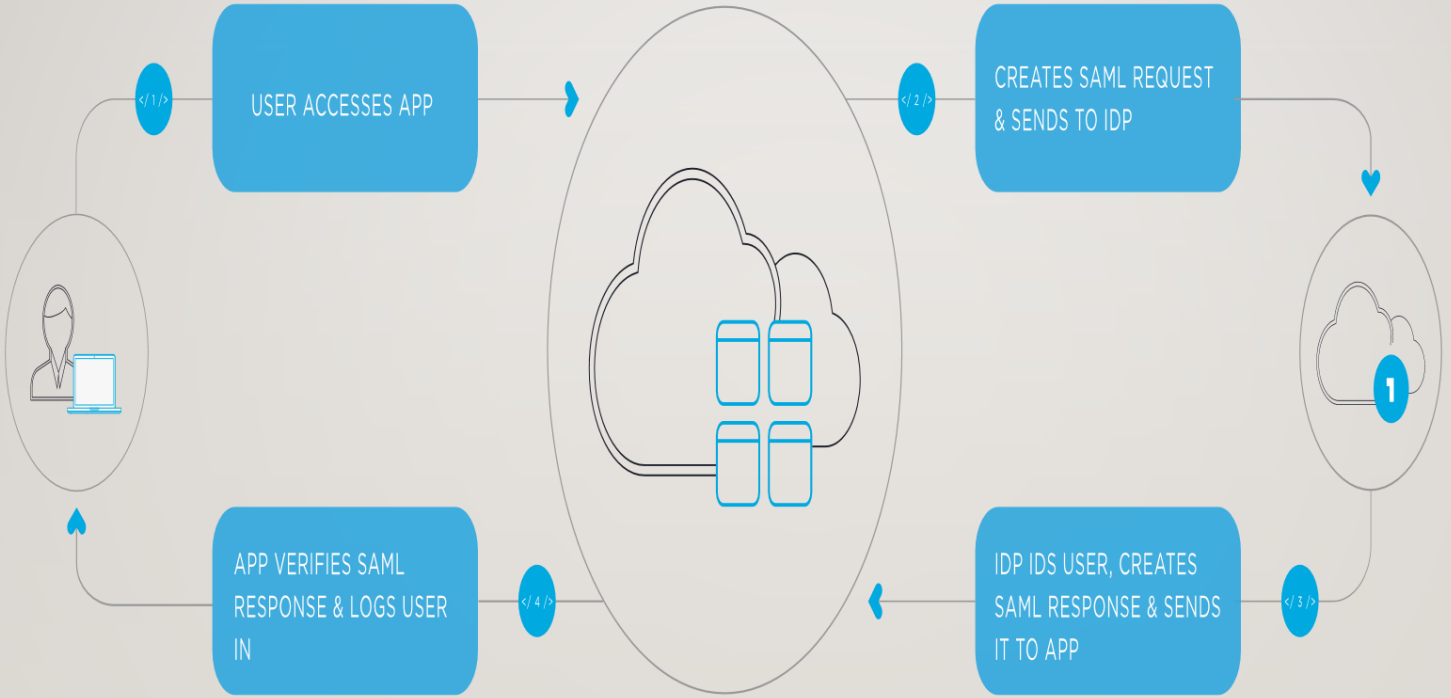
SAML, SSO AND HOW TO ENABLE SINGLE SIGN ON USING SAML IN A FLASK APP

WHAT IS SAML ?

- Security assertion markup language (SAML) is an xml-based framework for authentication and authorization between two entities: a service provider and an identity provider. The service provider agrees to trust the identity provider to authenticate users. In return, the identity provider generates an *authentication assertion*, which indicates that a user has been authenticated.

Benefits of SAML Authentication

- **Standardization**
- **Improved User Experience**
- **Increased Security**
- **Loose Coupling of Directories**
- **Reduced Costs for Service Providers**

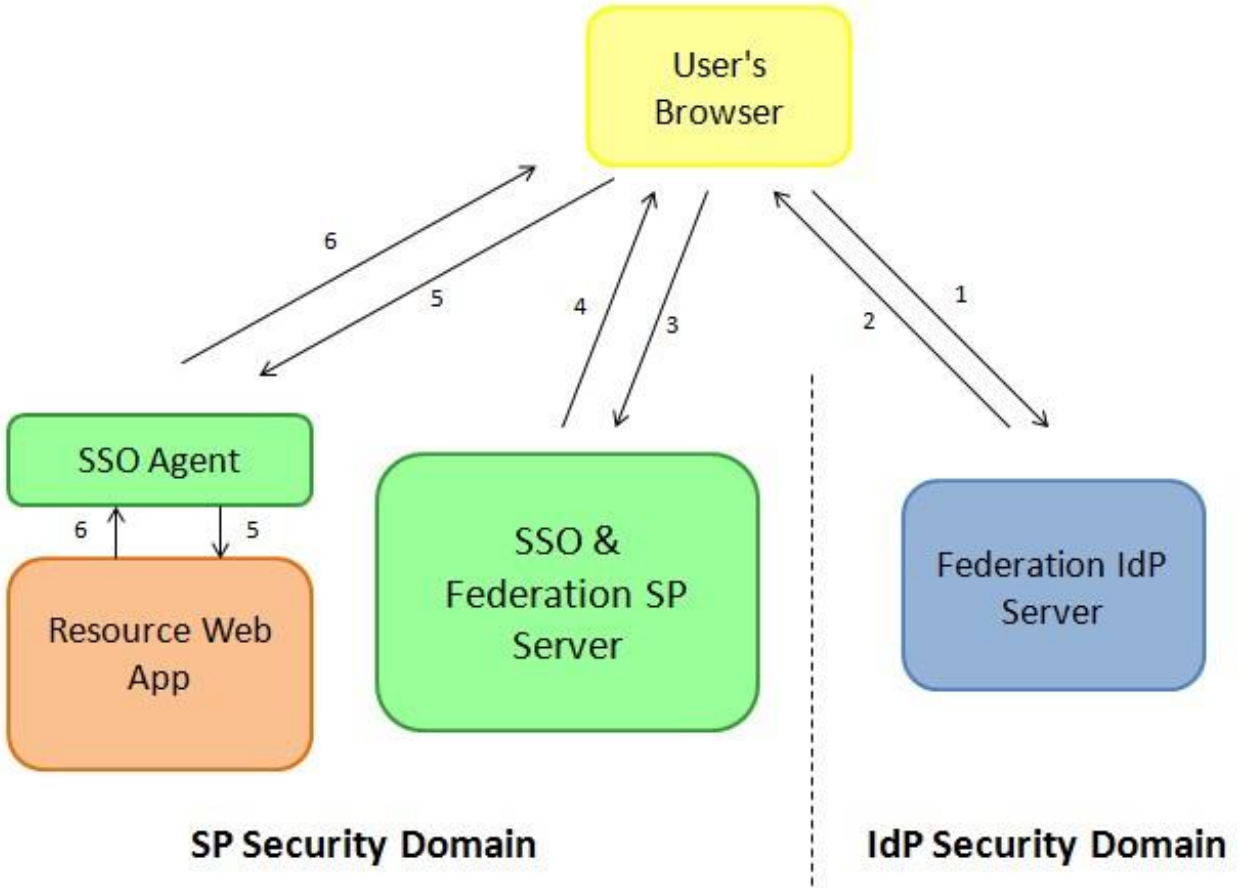

USER ACCESSES APP
CREATES SAML REQUEST & SENDS TO IDP
IDP IDS USER, CREATES SAML RESPONSE & SENDS IT TO APP
APP VERIFIES SAML RESPONSE & LOGS USER IN
1

HOW SAML WORKS

- The user accesses the remote application using a link on an intranet, a bookmark, or similar and the application loads.
- The application identifies the user's origin (by application subdomain, user IP address, or similar) and redirects the user back to the identity provider, asking for authentication. This is the authentication request.
- The user either has an existing active browser session with the identity provider or establishes one by logging into the identity provider.
- The identity provider builds the authentication response in the form of an XML-document – SAML assertion containing the user's username or email address, signs it using an X.509 certificate, and posts this information to the service provider.
- The service provider, which already knows the identity provider and has a certificate fingerprint, retrieves the authentication response and validates it using the certificate fingerprint.
- The identity of the user is established and the user is provided with app access.

IDP INITIATED SSO

- The user's browser accesses the IdP to start a Federation SSO flow by specifying
  - The SP to be used
  - Optionally the URL where the user's browser should be redirected after the Federation SSO is complete
- After having identified the user, the IdP creates an SSO Response with a SAML 2.0 Assertion containing user information as well as authentication data, and redirects the user's browser to the SP with the message and the RelayState parameter
- The user's browser presents the SSO response to the SP server
- The SP validates the SAML 2.0 Assertion and creates an SSO session for the user. The SSO server will then redirect the user's browser back to the resource originally requested
- The user's browser requests access to the resource. This time the SSO Web Agent grants access to the resource
- The Web Application returns a response to the user's browser


User's Browser
6
5
4
3
2
1
SSO Agent
6
5
Resource Web App
SSO & Federation SP Server
Federation IdP Server
SP Security Domain
IdP Security Domain

ONELOGIN'S SAML TOOLKIT

- Onelogin's SAML python toolkit (python3-saml) lets you turn your Python application into a SP (Service Provider) that can be connected to an IdP (Identity Provider).
- Single Sign on and Single Logout Services
- Assertion and NameID encryption
- Publish Service Provide Metadata

- Key features:
- **saml2int** - Implements the SAML 2.0 Web Browser SSO Profile.
- **Session-less** - Forget those common conflicts between the SP and the final app, the toolkit delegate session in the final app.
- **Easy to use** - Programmer will be allowed to code high-level and low-level programming, 2 easy to use APIs are available.
- **Tested** - Thoroughly tested.
- **Popular** - OneLogin's customers use it. Add easy support to your django/flask web projects.

HOW TO INTEGRATE WITH A FLASK APP

- SAML folder needs these two:
- Use a settings.json file that we should locate in any folder, but indicates its path with the 'custom_base_path' parameter and a advanced_settings.json files. Those files contain the settings for the saml toolkit. Copy them in your project and set the correct values.
- Certs – This can be a self signed certificate
  - Sp.cert
  - Sp.key
- Also we can provide those data in the setting file at the 'x509cert' and the privateKey' json parameters of the 'sp' element.
- If you want to create self-signed certs, you can do it at the https://www.samltool.com/self_signed_certs.php service, or using the command:

SOME EXAMPLES AND PSEUDO CODE

- Auth/views.py
- Prepare_request
- Idp_initiated_sso
- Saml/settings.json
- Saml/certs

ATTRIBUTE CONSUMER SERVICE (ACS)

- This code handles the SAML response that the IdP forwards to the SP through the user's client.
- The SAML response is processed and then checked that there are no errors. It also verifies that the user is authenticated and stored the userdata in session.
- At that point there are 2 possible alternatives:
- If no RelayState is provided, we could show the user data in this view or however we wanted.
- If RelayState is provided, a redirection takes place.
- Notice that we saved the user data in the session before the redirection to have the user data available at the RelayState view.
- In order to retrieve attributes we use:
- Attributes = auth.get_attributes()

REPLAY ATTACKS

- In order to avoid replay attacks, you can store the ID of the SAML messages already processed, to avoid processing them twice. Since the Messages expires and will be invalidated due that fact, you don't need to store those IDs longer than the time frame that you currently accepting.
- Get the ID of the last processed message/assertion with the get_last_message_id/get_last_assertion_id method of the Auth object.