SAML 2.0 Plugin

Bug Reference

Version 1: https://issues.apache.org/jira/browse/CLOUDSTACK-7083
Version 2: https://issues.apache.org/jira/browse/CLOUDSTACK-8457

Branch

Version 1: saml2

Version 2: saml2-production-grade

Introduction

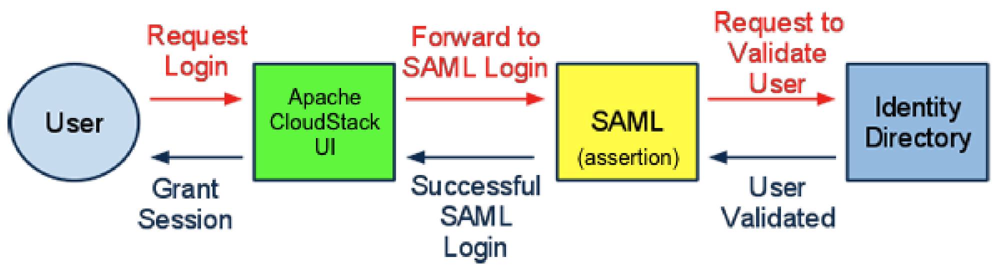
Purpose

Currently CloudStack has its own authentication mechanism, the two of which methods include username/password auth which is cookie/session-key based and another which is HMAC signature based that uses api key and secret key.

Many organization would want to use their existing authentication mechanism and have Single Sign On (SSO) and Single Log Out (SLO) to work on CloudStack UI and clients. SAML (Security Assertion Markup Language) 2.0 is an old, stable and widely used XML based authentication and authorization protocol supported by Salesforce, Google Apps and other public and private companies and the aim is to integrate the SSO SAML support in CloudStack.

This feature will be useful for users who may want to re-use their existing SAML 2.0 IdP (Identity Provider) service which holds the responsibility of users management, authentication & authorization assertions. In SAML terminology, CloudStack is a Service Provider (SP) that means it's the service that users would want to access and the organization would have their own Identity Provider (IdP) which is an authentication and authorization service backed by a user listing/store such as LDAP, AD etc.

For the scope of implementation and integration of the SAML plugin, we will skip the details related to IdP implementation.



Implementation Consideration

I evaluated many opensource libraries which implement the SAML protocol and the most widely used stable library was OpenSAML. It will be implemented as a plugin using OpenSAML.

The version 1 implementation shipped with ACS 4.5.0/4.5.1 is not production grade, only supports HTTP-redirect binding and has several issues with respect to security, compatibility, support and interoperability.

The version 2 implementation aims to produce a production grade SAML2 auth plugin has honours compatibility and interoperability, and supports more endpoint bindings with several widely used IdP servers such as Shibboleth.

Testing Consideration

For testing the plugin with CloudStack, we only need an IdP which can be setup locally using Shibboleth IdP software (https://www.testshib.org), or simply using one of the following free to use public IdP (discovery) service:

- Feide SAML SSO Demo service: https://openidp.feide.no
- SSO Circle: https://idp.ssocircle.com/sso/UI/Login

References

SAML for dummies: https://blog.surfnet.nl/?p=1417

SAML 2.0 on wikipedia: http://en.wikipedia.org/wiki/SAML_2.0

OpenSAML homepage: https://wiki.shibboleth.net/confluence/display/OpenSAML/Home

CloudStack SSO talk by John Burwell: youtube.com/watch?v=kCR0TzrfCOM

Implementing secure SSO with OpenSAML: youtube.com/watch?v=KroIZa1co6g

Document History

Version	Author / Reviewer	Date
1.0	Rohit Yadav / ACS-dev community	14/July/2014
2.0	Rohit Yadav	12/May/2015

Glossary

Term	Definition
Assertion	A part of SAML message (an XML document) which provides facts about subject of the assertion (typically about the authenticated user). Assertions can contain information about authentication, associated attributes or authorization decisions.
Artifact	Identifier which can be used to retrieve a complete SAML message from identity or service provider using a back-channel binding.
Binding	Mechanism used to deliver SAML message. Bindings are divided to front-channel bindings which use web-browser of the user for message delivery (e.g. HTTP-POST or HTTP-Redirect) and back-channel bindings where identity provider and service provider communicate directly (e.g. using SOAP calls in Artifact binding).
Discovery	Mechanism used to determine which identity provider should be used to authenticate user currently interacting with the service provider.
Metadata	Document describing one or multiple identity and service providers. Metadata typically includes entity identifier, public keys, endopoint URLs, supported bindings and profiles, and other capabilities or requirements. Exchange of metadata between identity and service providers is typically the first step for establishment of federation.
Profile	Standardized combination of protocols, assertions, bindings and processing instructions used to achieve a particular use-case such as single sign-on, single logout, discovery, artifact resolution.
Protocol	Definition of format (schema) for SAML messages used to achieve particular functionality such as requesting authentication from IDP, performing single logout or requesting attributes from IDP.
Identity provider (IDP)	Entity which knows how to authenticate users and provides information about their identity to service providers/relaying parties using federation protocols.
Service provider (SP)	Your application which communicates with the identity provider in order to obtain information about the user it interacts with. User information such as authentication state and user attributes is provided in form of security assertions.
Single Sign-On (SSO)	Process enabling access to multiple web sites without need to repeatedly present credentials necessary for authentication. Various federation protocols such as SAML, WS-Federation, OpenID or OAuth can be used to achieve SSO use-cases. Information such as means of authentication, user attributes, authorization decisions or security tokens are typically provided to the service provider as part of single sign-on.
Single Logout (SLO)	Process terminating authenticated sessions at all resources which were accessed using single sign-on. Techniques such as redirecting user to each of the SSO participants or sending a logout SOAP messages are typically used.

Feature Specifications

- With this feature a user in an organization having a SAML 2.0 compliant IdP would be able to do:

        - Single Sign On initiated by SP (CloudStack)
        - Single Local Logout (Logging out CloudStack only)
        - Single Global Logout (Logging out CloudStack and other SPs)
- **Scope:**
    - **Transport**: We will only support SAML transport over HTTP, which means supporting:
        - HTTP Redirects
    - **Clients:**
        - CloudStack UI will be fully supported
        - cloudmonkey and other command line clients may not handle HTTP redirects to IdP website and get assertions, therefore users would be required to use the HMAC auth method which uses API key and Secret key where SSO is enforced and username /password based auth is disabled
- **quality risks (test guidelines)**
    - functional
    - non functional: performance, scalability, stability, overload scenarios, etc
    - corner cases and boundary conditions
    - negative usage scenarios
- **Attributes:** The SAML assertion can have attributes set by IdP on them which can be used to configure user ACL inside CloudStack. The attributed are key value pairs in the XML. The following attributes are expected, on failure the (global) configuration params/values will be used:
    - unique id (email or uuid) - this is at least required
    - first name
    - last name
    - role (admin, resource admin, user etc.)
    - domain
    - misc: based on one's requirement, project etc.
- **Configuration:** The following config param/values are used to govern behaviour of the SAML auth plugin for CloudStack. In cases where SAML assertion does not send all the required attributes, the config params will be used:
    - IdP URL, metadata XML or discovery
    - IdP Entity Unique Name
    - SP Login and Logout Url
    - SP Unique Entity ID (IdP requires unique SAML federation/SP)
    - SP metadata and XML configuration
    - CloudStack Logout behaviour: SAML SLO will trigger global logout
    - Default SAML user role, user domain
    - Future: Cryptographic key management: SP keychain (signing key and encryption key) and IdP public key
- **Compatibility requirements:**
    - OpenSAML and other compliant SAML 2.0 IdP
- **Security specification**
    - TLS/SSL is recommended to do SAML assertion exchanges between CloudStack and IdP
- **Feature User:** CloudStack Admin, CloudStack UI users

Use cases

An organization that has SAML 2.0 compliant IdP service and wants to integrate SSO with CloudStack can use this feature.

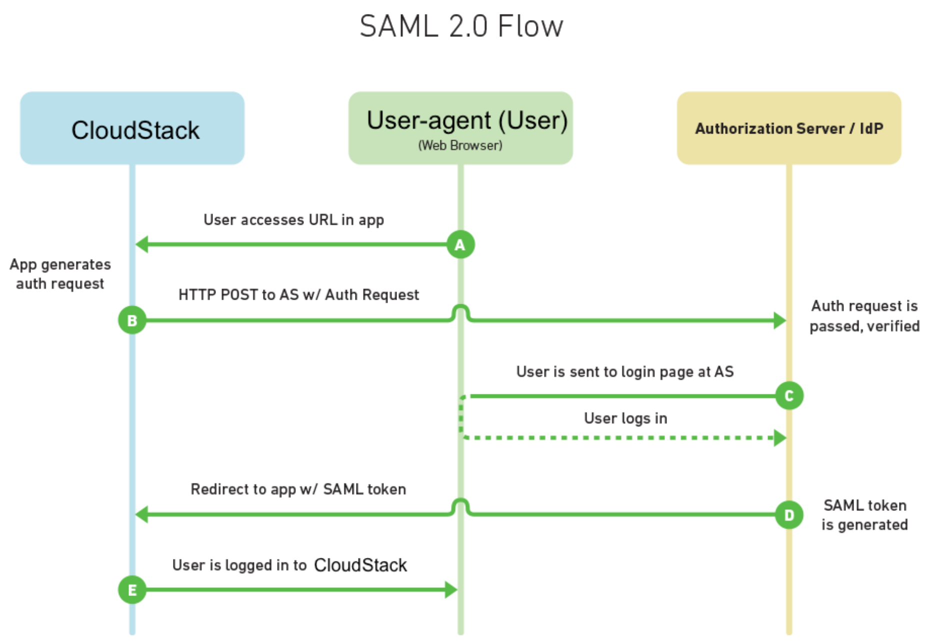
For Admins: Their administrator would create a unique entity ID and put in CloudStack SAML plugin. The CloudStack instance will have a base URL (http://domain/client etc.) which is configured in the SAML plugin. The admin will set IdP public key, IdP URL to the plugin using an API or web UI or as file where CloudStack is installed. The admin will generate X509 certificates (public and private keys) and store them with SAML plugin's java keystore as file, the public key and SAML SP metadata will be store at IdP server. Admin will configure whether logging out from CloudStack will locally logout users or globally. The admin will also configure default SAML authenticated user role which can only be changed by admin later once user is created /logs in the first time.

For Users: The users will open CloudStack UI URL in their browser and will be redirected to IdP URL. Once authenticated by IdP, the IdP will redirect users back to CloudStack UI which on receiving valid assertion from IdP will allow users in. If this is the first time users access CloudStack UI, CloudStack management server will create the user (with no password within CloudStack) using SAML assertion information (email, username etc).

Design and Architecture

A SAML based SSO infrastructure consists of three entities - user-agent (UA), service provider (SP) and identity provider (IdP). The UA is the user /browser, the SP is the application that the UA is accessing (i.e. Apache CloudStack UI) and the IdP is the identity service and does authentication and authorization, management of users among other things. IdP could be backed by LDAP, AD etc. For the scope of this feature, we only need to implement SAML SP plugin in CloudStack and use any free SAML 2.0 compliant IdP server for testing.

Note: After first round of discussing with community I'll add more design and architectural details.


SAML 2.0 Flow
CloudStack
User-agent (User)
(Web Browser)
Authorization Server / IdP
User accesses URL in app
A
App generates auth request
HTTP POST to AS w/ Auth Request
B
Auth request is passed, verified
User is sent to login page at AS
C
User logs in
Redirect to app w/ SAML token
D
SAML token is generated
User is logged in to CloudStack
E

1. User first goes to CloudStack UI and clicks a custom SAML SSO button OR s/he can directly go to the SAML SSO URL: /client/api?command=samlsso
2. User is redirected to the pre-setup IdP where s/he enters their username and password
3. IdP checks it and sends an assertion (XML response) back to CloudStack SP
4. The SP implementation in CloudStack checks the XML response, checks the signature and if it's alright it proceeds, else fails with error
5. Next it finds a NameID (identifier) which should be either persistent or emailAddress type, if it's transient etc. we try to finds it's UID (user ID) as per the LDAP standard attribute (RFC) which is "uid"
6. This NameID or uid is treated as UUID in CloudStack's terms, next it will find a user or create one using default role, domain from Global Config
7. Finally it lets the user log inside CloudStack

Open ended question

- RSA/X509 signature along with
- Create/manage JKS Keystore, X509 keys
- Handle expiry time (as sent by SAML assertion for valid user), on expiry how auth/reauth should work

Implementation Details

Version 1: Done

- Implemented as a separate SAML auth plugin
- The authentication layer was refactored as a framework to support pluggable authentication mechanisms such as SAML
- SSO/SLO works and is tested against https://openidp.feide.no which is also the default pre-configured ACS IDP
- SP XML metadata is listed using getSPMetaData API
- Global configs are created with saml.* namespace
- IdP MetaData is loaded when plugin starts using a provided url from Config
- Cryptographics signature verification is done on Response object when doing SSO
- Fixes CloudStack UI to auto unbox cookie values
- If users are removed from IdP, SSO will not work but recommendation is that admins disable the user account which is removed

Version 2: In Progress

- Support multiple IdP servers if a federated metadata is provided: https://issues.apache.org/jira/browse/CLOUDSTACK-8458
- Support multiple endpoint bindings such as HTTP-Post: https://issues.apache.org/jira/browse/CLOUDSTACK-8459
- Survey and support standard IdP servers such as Shibboleth 2.4+: https://issues.apache.org/jira/browse/CLOUDSTACK-8460
- Ensure compability as per the saml2int.org standards: https://issues.apache.org/jira/browse/CLOUDSTACK-8461

- Don't depend on NameId for getting unique ID (it could be a non-static one like urn:oasis:names:tc:SAML:2.0:nameid-format:transient), instead depend on an admin specified attribute: https://issues.apache.org/jira/browse/CLOUDSTACK-8463
- Let admin add/import and configure the user, the plugin should only check authorization. In case of multiple allowed SAML users with same username, authenticated user should be allowed to use any of the available account in any domain: https://issues.apache.org/jira/browse/CLOUDSTACK-8462
- Allow admins to save IdP metadata on filesystem or in DB: https://issues.apache.org/jira/browse/CLOUDSTACK-8464
- Improve UI for SAML authentication, currently a button is visible if saml plugin is enabled. We need to have a dropdown list of IdPs in case of multiple IdPs.
- Document the improved plugin usage

IP Clearance

- **Dependencies:**
  - OpenSAML: Apache 2.0 license

Dev-Testing

For testing the plugin, you may use one of publicly available IdP servers such as SSOCircle etc. or download the IDP ova appliance:

http://home.apache.org/~rohit/cloudstack/saml/

The IDP appliance has a pre-configured Shibbotleth 2.4.0 server and OpenLDAP.

Login credentials:
username = root
password = password

LDAP admin: admin
LDAP password: password
Hostname: idp.bhaisaab.org
IP: 172.16.154.200

The hostname idp.bhaisaab.org is A record to IP 172.16.154.200, if you need to change the IDPServer appliance IP (say in KVM, VMWare Fusion, VirtualBox), add an entry in your hosts files for idp.bhaisaab.org domain.

Note: After starting the IDP VM, run "sudo ntpdate pool.ntp.org" to update its date/time and make sure that the management server host has same effective time as the IDP server. In case time/date mismatches, the IDP server on single-sign-on will state security errors and fail.

LDAP interface: idp.bhaisaab.org/phpldapadmin
Shibboleth IdP Metadata: idp.bhaisaab.org/idp/shibboleth

To test, build CloudStack, deploydb and deploydb-saml. The SAML plugin is located in the source directory at this location: "plugins/user-authenticators/saml2/"

The deploydb-saml will automatically configure ldap and saml auth plugin in CloudStack to use with this appliance.

- Build CloudStack:
  mvn clean install -P developer
- Deploy Database:
  mvn -q -Pdeveloper -pl developer -Ddeploydb
  mvn -q -Pdeveloper -pl developer -Ddeploydb-saml
- Start management server:
  mvn -pl client jetty:run -Djava.net.preferIPv4Stack=true

Log in to CloudStack and check/update global settings by searching for all config settings starting with 'saml'. If you need to change them, restart the management server.

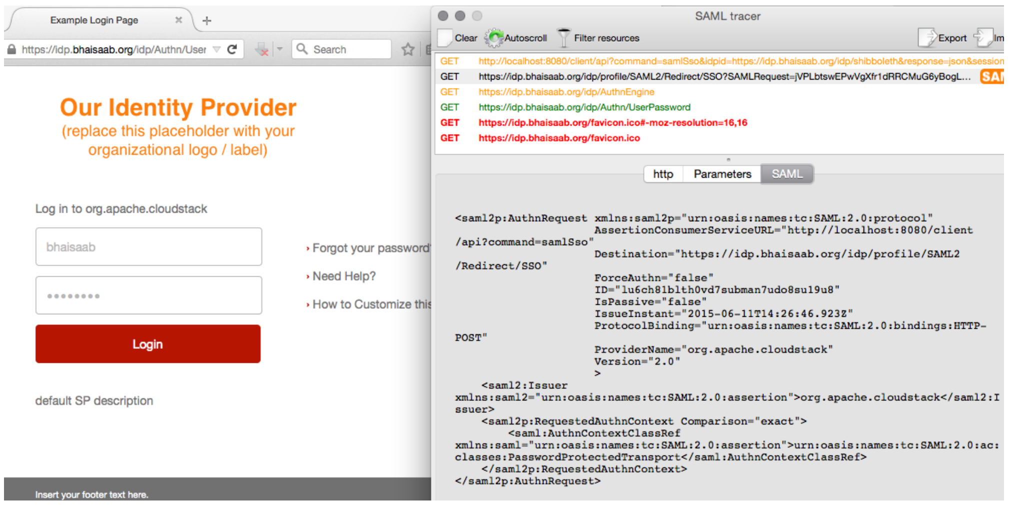
Import users from LDAP or add them manually. Next authorize the user(s) to use SAML SSO against a IdP server by choosing the correct entity ID.

Log out and select an appropriate IdP server from the list of dropdown (in the default case it will be only one, pre-selected) and enter the domain where your account is, the default domain is the ROOT or / domain. Click on SAML SSO button which will redirect you to the IdP log in page, where upon successful authentication you'll be redirected to CloudStack UI with your user account logged in.

As a compatibility requirement, the SAML SP implementation in CloudStack need to adhere to the SAML profile: http://saml2int.org/profile/current/

There is a strict policy on timestamps and cryptographic token checking using IdP server's public key and SP (CloudStack's) private key, so sometimes upon successful authentication the UI may not get logged in - in this case simply re-login using SAML SSO. By default when the management server starts for the first time, SAML certificates are created and stored in cloud.keystore table.

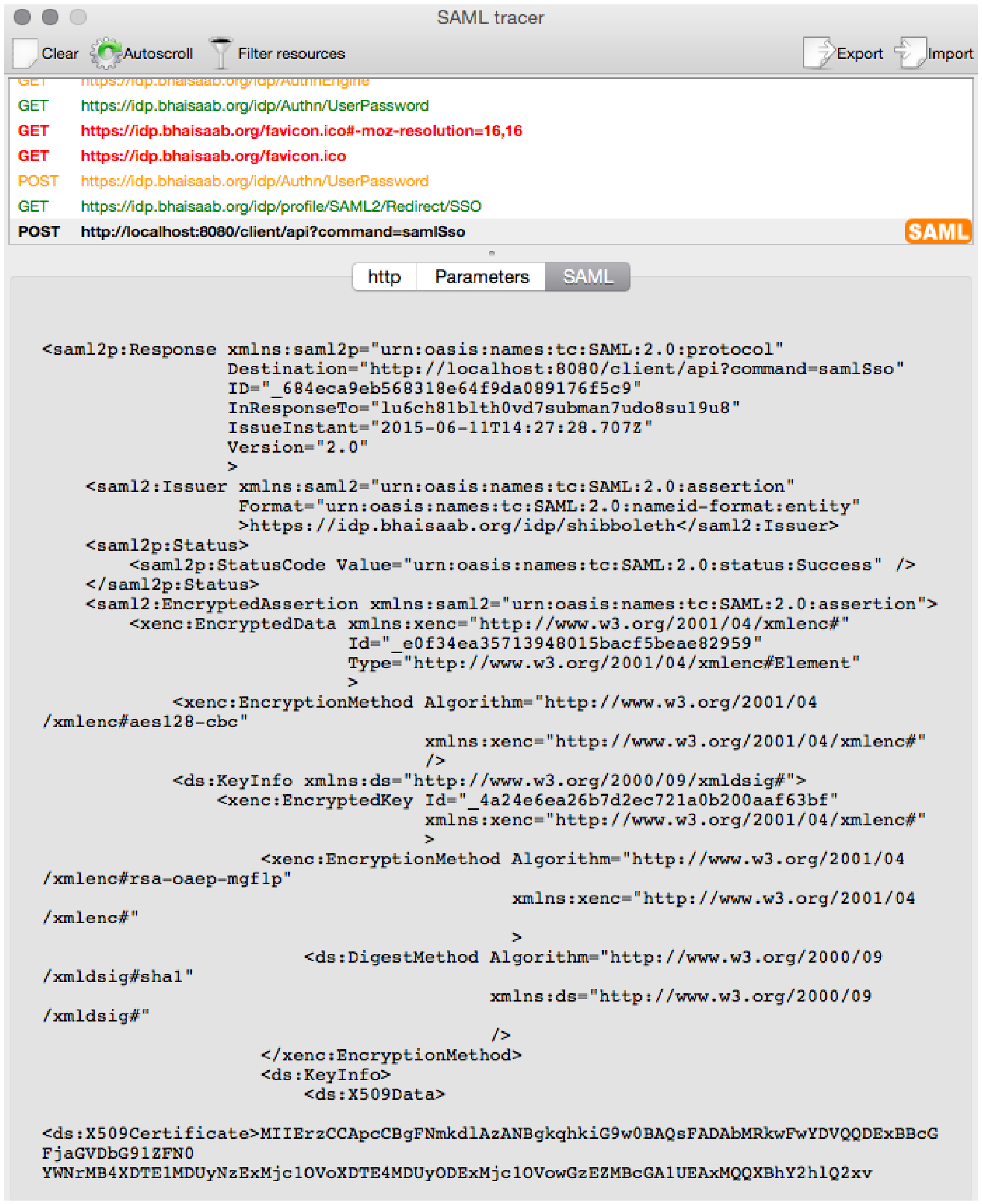
On every SAML SSO attempt, an entry is recorded in CloudStack's cloud.saml_token table to protect against spoofed log-in attempts or an IdP initiated log in where CloudStack won't know in specific domain the user wishes to log in. On Firefox, SAML tracer add-on can be used to view the SAML tokens that get exchanged. Every SAML Request (AuthnRequest) is a XML encoded HTTP-POST request as per the saml2int.org profile, the XML consists of the issuer information (SP information, name, entity ID), a unique ID (securely generated random string) and some security enforcement on how to authenticate the user. In cloud.saml_token table, we store the IdP we will redirect the user to, along with the unique ID used in the XML and the domain name to be later referenced.



```
GET   http://localhost:8080/client/api?command=samlSso&idpid=https://idp.bhaisaab.org/idp/shibboleth&response=json&session
GET   https://idp.bhaisaab.org/idp/profile/SAML2/Redirect/SSO?SAMLRequest=jVPLbtswEPwVgXfr1dRRCMuG6yBogL...
GET   https://idp.bhaisaab.org/idp/AuthnEngine
GET   https://idp.bhaisaab.org/idp/Authn/UserPassword
GET   https://idp.bhaisaab.org/favicon.ico#-moz-resolution=16,16
GET   https://idp.bhaisaab.org/favicon.ico
```

```
<saml2p:AuthnRequest xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
                     AssertionConsumerServiceURL="http://localhost:8080/client
/api?command=samlSso"
                     Destination="https://idp.bhaisaab.org/idp/profile/SAML2
/Redirect/SSO"
                     ForceAuthn="false"
                     ID="lu6ch81blth0vd7subman7udo8su19u8"
                     IsPassive="false"
                     IssueInstant="2015-06-11T14:26:46.923Z"
                     ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
POST"
                     ProviderName="org.apache.cloudstack"
                     Version="2.0"
                     >
    <saml2:Issuer
xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">org.apache.cloudstack</saml2:I
ssuer>
    <saml2p:RequestedAuthnContext Comparison="exact">
        <saml:AuthnContextClassRef
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">urn:oasis:names:tc:SAML:2.0:ac:
classes:PasswordProtectedTransport</saml:AuthnContextClassRef>
    </saml2p:RequestedAuthnContext>
</saml2p:AuthnRequest>
```

After successfully authenticated from the IdP/federation login system, a SAML Response is sent to the CloudStack SAML Plugin either as a HTTP-Redirect (GET) or HTTP-POST (POST) request that consists of an XML that has issuer information (IdP entity name, timestamp etc), response information (saml token ID, response to ID), authentication status (success, failure etc) and encrypted and unencrypted attributes.



```
GET    https://idp.bhaisaab.org/idp/AuthnEngine
GET    https://idp.bhaisaab.org/idp/Authn/UserPassword
GET    https://idp.bhaisaab.org/favicon.ico#-moz-resolution=16,16
GET    https://idp.bhaisaab.org/favicon.ico
POST   https://idp.bhaisaab.org/idp/Authn/UserPassword
GET    https://idp.bhaisaab.org/idp/profile/SAML2/Redirect/SSO
POST   http://localhost:8080/client/api?command=samlSso
```

```
<saml2p:Response xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
                 Destination="http://localhost:8080/client/api?command=samlSso"
                 ID="_684eca9eb568318e64f9da089176f5c9"
                 InResponseTo="1u6ch81b1th0vd7subman7udo8su19u8"
                 IssueInstant="2015-06-11T14:27:28.707Z"
                 Version="2.0"
                 >
    <saml2:Issuer xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
                  Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity"
                  >https://idp.bhaisaab.org/idp/shibboleth</saml2:Issuer>
    <saml2p:Status>
        <saml2p:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success" />
    </saml2p:Status>
    <saml2:EncryptedAssertion xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
        <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
                            Id="_e0f34ea35713948015bacf5beae82959"
                            Type="http://www.w3.org/2001/04/xmlenc#Element"
                            >
            <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04
/xmlenc#aes128-cbc"
                                   xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
                                   />
            <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <xenc:EncryptedKey Id="_4a24e6ea26b7d2ec721a0b200aaf63bf"
                                   xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
                                   >
                    <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04
/xmlenc#rsa-oaep-mgf1p"
                                           xmlns:xenc="http://www.w3.org/2001/04
/xmlenc#"
                                           >
                        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09
/xmldsig#sha1"
                                         xmlns:ds="http://www.w3.org/2000/09
/xmldsig#"
                                         />
                    </xenc:EncryptionMethod>
                    <ds:KeyInfo>
                        <ds:X509Data>

<ds:X509Certificate>MIIErzCCApcCBgFNmkdlAzANBgkqhkiG9w0BAQsFADAbMRkwFwYDVQQDExBBcG
FjaGVDbG91ZFN0
YWNrMB4XDTE1MDUyNzExMjcl0VoXDTE4MDUyODExMjcl0VowGzEZMBcGA1UEAxMQQXBhY2hlQ2xv
```

The response ID in the SAML Response is same as the unique ID we generated for the SAML request, this ID can be used to verify from cloud. saml_token table if we generated any authentication response or not, along with the IdP and domain selected if we did request it. Using this information and user attributes parsed from the XML's encrypted or unencrypted attribute nodes, we get the user details (username, domain and IDP name) and if the user is authorized we log in the user by setting appropriate cookies and redirect the browser to CloudStack's UI which checks these cookies and logs in the user. We return error, in case the user is found to be not authorized i.e. either the user was not SAML enabled or not enabled for the specific IdP or just does not exist in the chosen domain.

it consists of   Using saml_token table, we can know if a user wants to access a specific domain upon successful log in, this is useful in case a user has multiple user account with same "username" across several domains. The saml_token table can be flushed after a timeout period.

The SAML IDP metadata of the IDP/Shibbotleth server can be accessed from: https://idp.bhaisaab.org/idp/shibboleth

The SAML SP metadata can be accessed from CloudStack: http://localhost:8080/client/api?command=getSpMetadata