Cross-Enterprise User Assertion (XUA)
Analysis document

Version 1.0

By ViCarePlus Team

Prepared by

ViSolve Inc.,
Contact: 408.666.4320
EMail: vicareplus_engg@visolve.com
www.visolve.com

23rd April 2010

Revision History

Version	Date	Author	Reviewed By
1.0	19/04/10	ViCarePlus Team	Team
2.0	23/04/10	ViCarePlus Team	Team

1. XUA

What is XUA?

Cross-Enterprise User Assertion (XUA) profile is used to provide the user identity in transactions that cross enterprise boundaries.

Eg: user (on the EHR side) participating in the HIE activities (eg:for document sharing), need to provide their identity to the HIE. XUA profile specifies the standards to accomplish this.

Enterprises may choose to have their own user directory and their own unique method of authenticating. To provide accountability in these cross enterprise transactions there is a need to identify the requesting user in a way that the receiver can make access decisions and proper audit entries.

Actors and transactions

The XUA profile has three actors participating in three transactions. The following figure shows the actors directly involved in the XUA Profile and the transactions between them.

Other actors and transactions that may be indirectly involved due to their participation in other grouped profiles are shown in italics. The “Authenticate User” transaction is outside the scope of this profile and may be filled through the use of EUA or some other enterprise class authentication. The “Request any service” transaction is outside the scope of this profile and represents an existing transaction that needs to convey user authentication information (i.e. XUA Assertion).

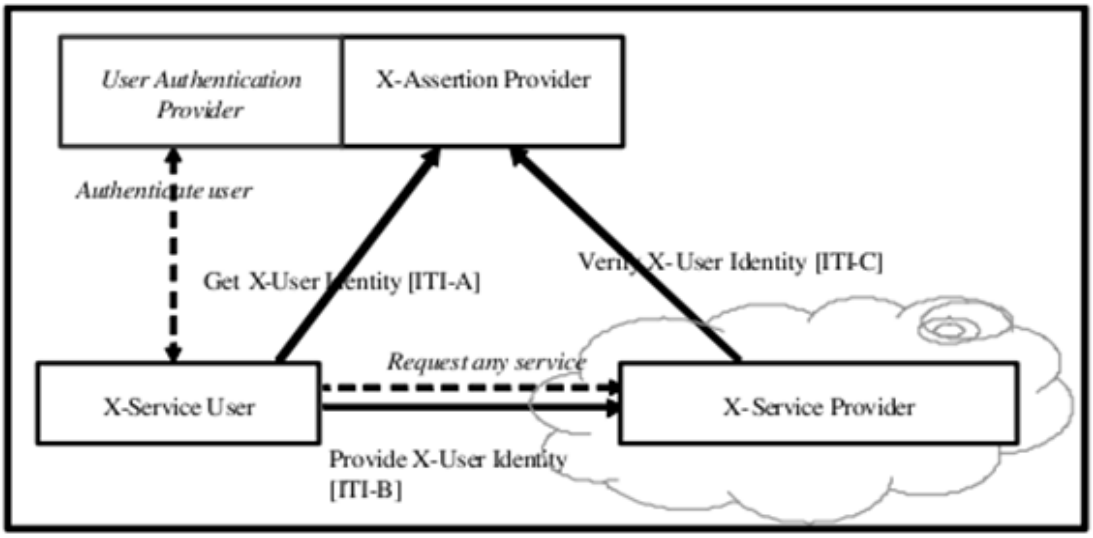


Figure: Actors and transactions in XUA profile

X-Service User: requests and obtains an assertion from an authority (the X-Assertion Provider) and sends it to the X-Service Provider

X-Assertion Provider: Provider the SAML assertion (which describes the user) to the X-Service User upon request. It contacts the Authentication provider to validate the user and collects the user attributes

X-Service Provider: receives the assertion from the X-Service User, parses and validates the assertion with a trusted authority (the X-Assertion Provider) and acts accordingly

2. SAML

Security Assertion Markup Language (**SAML**) is an XML -based standard for exchanging authentication and authorization data between security domains, that is, between an *identity provider* (a producer of assertions) and a *service provider* (a consumer of assertions).

SAML consists of building-block components that, permits transfer of identity, authentication, attribute, and authorization information to be exchanged between autonomous organizations. The “core” SAML specification defines the structure and content of Assertions – which carry statements about a Principal (user, application, system...) as asserted by an Asserting Party. These are defined by an XML schema.

SAML Components

[Assertion, profile, protocols, binding – intro]

The SAML components and their individual parts are as follows:

2.1 Assertions

SAML allows for one party to assert characteristics and attributes of a subject. For instance, a SAML assertion could state that the subject is “John Doe”, has “Gold” status, has an email address of john.doe@example.com, and is a member of the “engineering” group. SAML assertions are encoded in an XML schema. SAML defines three kinds of statements that can be carried within an assertion:

• Authentication statements: These are issued by the party that successfully authenticated the user. They describe who issued the assertion, the authenticated subject, and the validity period, plus other authentication-related information.

• Attribute statements: These contain specific details about the subject (for example, that user “John Doe” has “Gold” status).

• Authorization decision statements: These define something the subject is entitled to do (for example, whether “John Doe” is permitted to buy a specified item).

2. 2 Protocols

SAML defines a number of request/response protocols, which are encoded in an XML schema as a set of request-response pairs. The protocols defined are:

• Assertion Query and Request Protocol: Defines a set of queries by which existing SAML assertions may be obtained. The query can be on the basis of a SAML message reference, the subject, or the statement type.

• Authentication Request Protocol: Defines a protocol by which a service provider or principal can request assertions from an identity provider tailored to the requirements of a particular SAML profile, for example the Web Browser SSO Profile.

• Artifact Resolution Protocol: Provides a mechanism by which protocol messages may be passed by reference using a small, fixed-length value called an artifact. The artifact receiver uses the Artifact Protocol to dereference the actual protocol message.

• Name Identifier Management Protocol: Provides mechanisms to change the value of the principal's name identifier. The issuer of the request can be either the service provider or the

identity provider. The protocol also provides a mechanism to terminate an association of a name between an identity provider and service provider.

• Single Logout Protocol: Defines a request that allows near-simultaneous logout of all sessions associated by a principal. The logout can be directly initiated by the principal, due to a session timeout or because a user access rights have been revoked. Logout can also be initiated by a provider site.

• Name Identifier Mapping Protocol: Provides a mechanism to programmatically map one SAML name identifier into another, subject to appropriate policy controls. In addition to request and response messages comprising the protocols listed here, SAML provides for a special representation of any type of protocol message, called an artifact, which can be dereferenced to obtain the full message. An artifact takes the form of a base-64 encoded string.

2.3 Bindings

These detail exactly how the SAML protocols map onto underlying transport protocols. For instance, the SAML specification provides a binding of how SAML requests and responses are carried with SOAP exchange messages. The bindings defined are:

• SAML SOAP Binding: Defines how SAML protocol messages are transported within SOAP 1.1 messages, with details about using SOAP over HTTP.

• Reverse SOAP (PAOS) Binding: Defines a multi-stage SOAP/HTTP message exchange that permits an HTTP client to be a SOAP responder. Used in the Enhanced Client and Proxy Profile and particularly designed to support WAP gateways.

• HTTP Redirect Binding: Defines how SAML protocol messages can be transported using HTTP redirect messages (302 status code responses).

• HTTP POST Binding: Defines how SAML protocol messages can be transported within the base64-encoded content of an HTML form control.

• HTTP Artifact Binding: Any SAML protocol message can be represented by an artifact, which has a compact base-64 format and allows for the real message to be “pulled” (dereferenced). This binding defines how an artifact is transported by HTTP using one of two mechanisms: either an HTML form control or a query string in the URL.

• SAML URI Binding: Defines a means for retrieving a SAML assertion by resolving a URI (uniform resource identifier).

2.4 Profiles

Profiles define how the SAML assertions, protocols, and bindings are combined for interoperability in particular usage scenarios. Some of these are described in detail later on in the document. In summary they are:

• Web Browser SSO Profile: Defines a mechanism for single sign-on by unmodified web browsers to multiple service providers using the Authentication Request protocol in combination with the HTTP Redirect, HTTP POST, and Artifact bindings.

• Enhanced Client and Proxy (ECP) Profile: Defines a profile of the Authentication Request protocol in conjunction with the Reverse-SOAP and SOAP bindings suited to clients or gateway devices with knowledge of one or more identity providers.

• Identity Provider Discovery Profile: Defines one possible mechanism for a set of cooperating Identity and service providers to obtain the identity providers used by a principal.

• Single Logout Profile: A profile of the SAML Single Logout protocol is defined. Defines how SOAP, HTTP Redirect, HTTP POST, and HTTP Artifact bindings may be used.

• Name Identifier Management Profile: Defines how the Name Identifier Management protocol may be used with SOAP, HTTP Redirect, HTTP POST, and HTTP Artifact bindings.

• Artifact Resolution Profile: Defines how the Artifact Resolution protocol uses a synchronous binding, for example the SOAP binding.

• Assertion Query/Request Profile: Defines how the SAML query protocols (used for obtaining SAML assertions) use a synchronous binding such as the SOAP binding.

• Name Identifier Mapping Profile: Defines how the Name Identifier Mapping protocol uses a synchronous binding such as the SOAP binding.

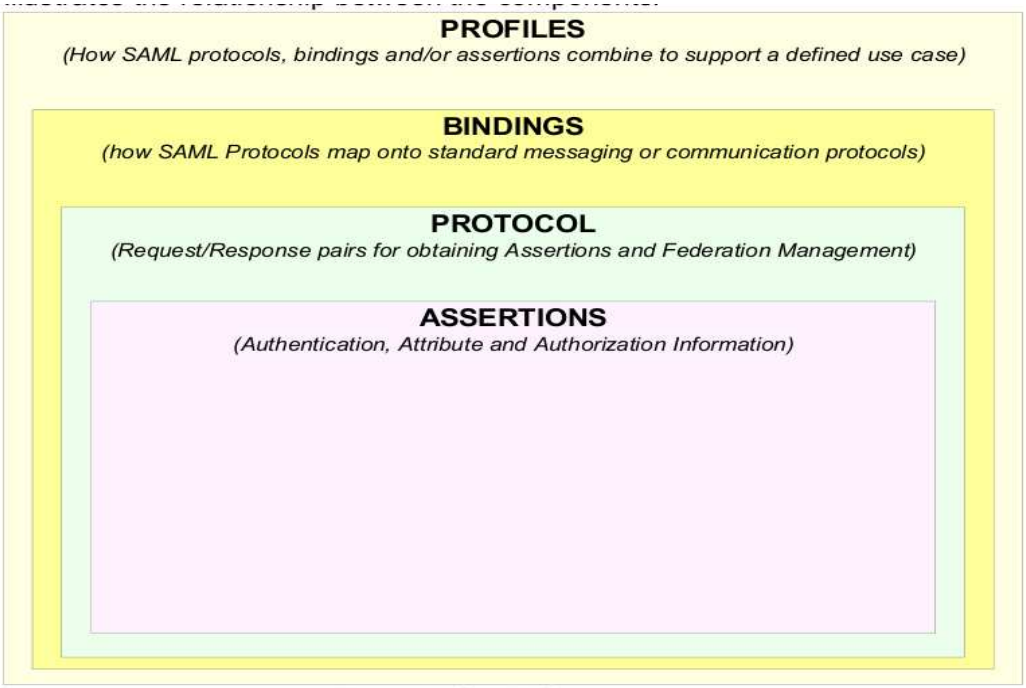


Figure: SAML Components

What assertion, profile, protocol - we are going to follow for XUA?

Assertions: Authentication statements

Protocol : Assertion Query and Request Protocol

Binding : SAML SOAP Binding

Profile : Web Browser SSO Profile

2.5 Types of binding suited for XUA

1. SP initiated: POST->POST binding

In this use case the user attempts to access a resource on www.abc.com. However they do not have current an existing session context on this site (e.g. logged on) and their identity is managed by www.xyz.com. A SAML <AuthnRequest> is sent to their identity provider so that the identity provider can provide back a SAML assertion concerning the user. HTTP POST

messages are used to deliver the SAML <AuthnRequest> to the identity provider as well as receive back the SAML response.

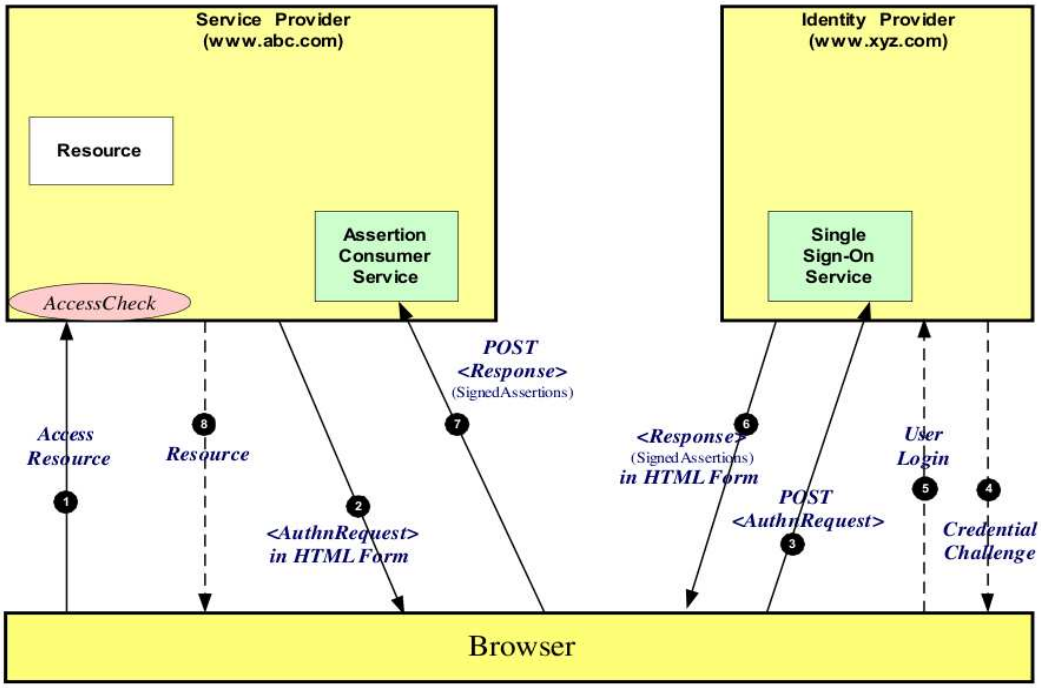


Figure: SP initiated POST → POST binding

The processing is as follows:

1. The user attempts to access a resource on www.abc.com. The user does not have any current logon session (i.e. security context) on this site, and is unknown to it.

2. The SP sends a HTML form back to the browser. The HTML FORM contains a SAML <AuthnRequest> defining the user for which authentication and authorization information is required. Typically the HTML FORM will contain an input or submit action that will result in an HTTP POST.

3. The browser, either due to a user action or via an “auto-submit”, issues an HTTP POST containing the SAML <AuthnRequest> to the identity provider's Single Sign-On service.

4. If the user does not have an existing security context on the identity provider, or the policy define that authentication is required, the user will be challenged to provide valid credentials.

5. The user provides valid credentials and a security context is created for the user.

6. The Single Sign-On Service sends a HTML form back to the browser. The HTML FORM contains a SAML response, within which is a SAML assertion. The SAML specification mandates that the response must be digitally signed. Typically the HTML FORM will contain an input or submit action that will result in an HTTP POST.

7. The browser, either due to a user action or via an “auto-submit”, issues a HTTP POST containing the SAML response to be sent to the service provider's Assertion Consumer service.

8. The service provider's Assertion Consumer service validates the digital signature on the SAML Response. If this, and the Assertion validate correctly, it sends an HTTP redirect to the browser causing it to access the TARGET resource, with a cookie that identifies the local session (use of a cookie is implementation specific, other techniques to maintain the security context at

the SP can be used). An access check is then made to establish whether the user has the correct authorization to access the www.abc.com web site and the TARGET resource. If the access check passes, the TARGET resource is then returned to the browser.

2. IdP initiated: POST binding

In this use case the user has a security context on the identity provider (www.xyz.com) and wishes to access a resource at a service provider (www.abc.com). The SAML assertion is transported to the service provider using the HTTP POST binding.

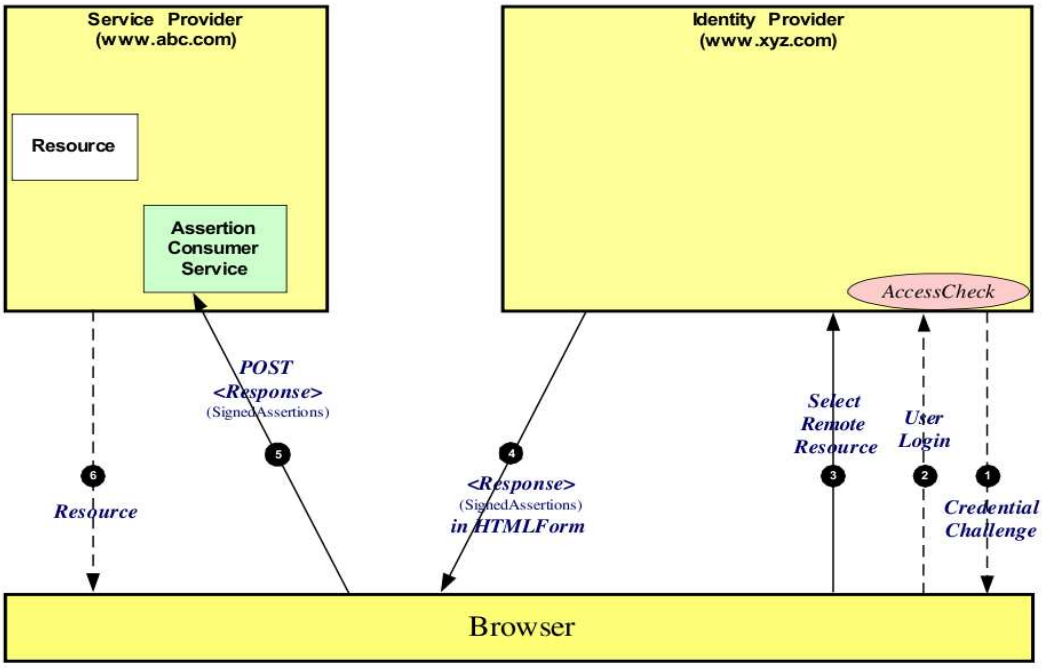


Figure: IdP initiated Post binding

The processing is as follows:

1. At some point the user will have been challenged to supply their credentials to the site www.xyz.com.

2. The user successfully provides their credentials and has a security context with the identity provider.

3. The user selects a menu option (or function) on the displayed screen that means the user wants to access a resource or application on another web site www.xyz.com.

4. The SP sends a HTML form back to the browser. The HTML FORM contains a SAML response, within which is a SAML assertion. The SAML specification mandates that the response must be digitally signed. Typically the HTML FORM will contain an input or submit action that will result in an HTTP POST.

5. The browser, either due to a user action or via an “auto-submit”, issues an HTTP POST containing the SAML response to be sent to the Service provider's Assertion Consumer service.

6. The service provider's Assertion Consumer service validates the digital signature on the SAML Response. If this, and the Assertion validate correctly, it sends an HTTP redirect to the browser causing it to access the TARGET resource, withing with a cookie that identifies the local session (use of a cookie is implementation specific, other techniques to maintain the security

context at the SP can be used). An access check is then made to establish whether the user has the correct authorization to access the www.abc.com web site and the TARGET resource. If the access check passes, the TARGET resource is then returned to the browser.

2.6 SAML Sample Request format

```
<samlp:AuthnRequest
  xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  ID="aaf23196-1773-2113-474a-fe114412ab72"
  Version="2.0"
  IssueInstant="2004-12-05T09:21:59Z"
  AssertionConsumerServiceIndex="0"
  AttributeConsumingServiceIndex="0">
  <saml:Issuer>https://sp.example.com/SAML2</saml:Issuer>
  <samlp:NameIDPolicy
    AllowCreate="true"
    Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient"/>
 </samlp:AuthnRequest>
```

2.7 SAML Sample Response format

```
<saml:Assertion
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  ID="b07b804c-7c29-ea16-7300-4f3d6f7928ac"
  Version="2.0"
  IssueInstant="2004-12-05T09:22:05Z">
  <saml:Issuer>https://idp.example.org/SAML2</saml:Issuer>
  <ds:Signature
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#">...</ds:Signature>
  <saml:Subject>
    <saml:NameID
      Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient">
      3f7b3dcf-1674-4ecd-92c8-1544f346baf8
    </saml:NameID>
    <saml:SubjectConfirmation
      Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
      <saml:SubjectConfirmationData
        InResponseTo="aaf23196-1773-2113-474a-fe114412ab72"
        Recipient="https://sp.example.com/SAML2/SSO/POST"
        NotOnOrAfter="2004-12-05T09:27:05Z"/>
    </saml:SubjectConfirmation>
  </saml:Subject>
  <saml:Conditions
    NotBefore="2004-12-05T09:17:05Z"
    NotOnOrAfter="2004-12-05T09:27:05Z">
    <saml:AudienceRestriction>
      <saml:Audience>https://sp.example.com/SAML2</saml:Audience>
    </saml:AudienceRestriction>
  </saml:Conditions>
  <saml:AuthnStatement
```

```
    AuthnInstant="2004-12-05T09:22:00Z"
    SessionIndex="b07b804c-7c29-ea16-7300-4f3d6f7928ac">
    <saml:AuthnContext>
      <saml:AuthnContextClassRef>
        urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
     </saml:AuthnContextClassRef>
    </saml:AuthnContext>
  </saml:AuthnStatement>
  <saml:AttributeStatement>
    <saml:Attribute
      xmlns:x500="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
      x500:Encoding="LDAP"
      NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
      Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
      FriendlyName="eduPersonAffiliation">
      <saml:AttributeValue
        xsi:type="xs:string">member</saml:AttributeValue>
      <saml:AttributeValue
        xsi:type="xs:string">staff</saml:AttributeValue>
    </saml:Attribute>
  </saml:AttributeStatement>
</saml:Assertion>
```

3. HITSP/C19 Entity Identity Assertion

The Entity Identity Assertion Component provides the mechanisms to ensure that an entity is the person or application that claims the identity provided. An example of this Component is the validation and assertion of a consumer logging on to a Personal Health Record (PHR) system

The following are the requirements derived from the AHIC Use Cases for the authentication and assertion of users:

1. Entities are asserted to assure that the entity is the person or application that claims the identity.

In addition, the Entity Identity Assertion Component is meant to apply to the following scenarios as defined in the HITSP Interoperability Specifications:

2. User using a Document Registry or Document Repository where the Service Provider wants a user identity for additional detail in their audit log.

3. User using a Document Registry or Document Repository where the Service Provider wants to be assured that the user has been authenticated to a specific assurance level.

4. User using a Document Registry or Document Repository where the Service Provider wants to impose additional access controls.

5. User using a Document Registry or Document Repository is the consumer. The consumer is using an authorized PHR service which is handling the Document Consumer responsibilities. The Service Provider wants to restrict the information returned to those that have been released for consumer consumption (for example a lab result that regulations require the provider to discuss in person before releasing the information).

Pre-condition

Necessary pre-conditions that must be in place prior to the onset of the workings of the Component.

Entities must have been identified and provisioned (credentials issued, privileges assigned)
Audit services are initialized as outlined in the HITSP/T15 Collect and Communicate Security Audit Trail Transaction
Secure channels are initialized in accordance with HITSP/T17 Secured Communication Channel Transaction
All interface(s)s are synchronized to a consistent time base by the HITSP/T16 Consistent Time Transaction

Post Condition
This section provides an overview of the post-conditions or results that must occur at the end of the Component in order for the Component to be deemed successfully completed.

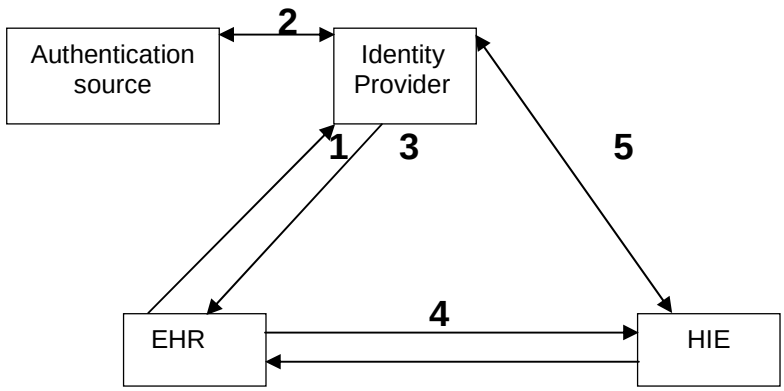
Entity has authenticated
An error condition occurs. This can include errors in the verification step – malformed assertion; assertion from a distrusted identity provider; assertion from individual without enough information to perform verification; or identity provider is unknown
Entity identity assertion is verified

Required Output

Results of the assertion are made available to the assertion provider	SAML assertion
A security audit event is generated	a specific audit event will be generated specific to the vocabulary required to generate that event
Authentication information that was verified is available	Standards for minimum core set of required data are specific to a domain or organization

4. Proposed architecture

4.1 Diagram of the proposed architecture



4.2 Explanation

Steps:

1. The EHR user requesst for an assertion from an authority (the Identity Provider).
2. The Identity provider authenticates the user.
3. When authentication is successful, the identity provider responses back to user with user assertion.
4. After obtaining assertion from the identity provider, the user communicates the service request (along with the assertion), to the service provider (HIE).
5. HIE verifies the assertion by communicating with the identity provider.
6. On successful verification, HIE serves the user for the service requested.

5. XUA Implementation Details

The types of tasks that requires at any EHR side are:

1. Identity provider implementation
2. Changes at the EHR side for the incorporating SAML HTTP POST Binding

5.1 Identity Provider

- Simplesamlphp or Opensaml can be used as identity provider
- or Identity provider can be developed from scrap

5.2 EHR changes

HTTP POST Binding

Pre-generated assertions thus Idp initiated -> POST binding

In the following example, both the service provider and the identity provider use an HTTP POST Binding. Initially, the service provider responds to a request from the user agent with a document containing an XHTML form:

```
<form method="post" action="https://idp.example.org/SAML2/SSO/POST" ...>
  <input type="hidden" name="SAMLRequest" value="request" />
  ...
  <input type="submit" value="Submit" />
</form>
```

The value of the SAMLRequest parameter is the base64 encoding of a <samlp:AuthnRequest> element, which is transmitted to the identity provider via the browser. The SSO service at the identity provider validates the request and responds with a document containing another XHTML form:

```
<form method="post" action="https://sp.example.com/SAML2/SSO/POST" ...>
  <input type="hidden" name="SAMLResponse" value="response" />
  ...
```

```
  <input type="submit" value="Submit" />
</form>
```

The value of the SAMLResponse parameter is the base64 encoding of a <samlp:Response> element, which likewise is transmitted to the service provider via the browser.

To automate the submission of the form, the following line of JavaScript may appear anywhere on the XHTML page:

```
window.onload = function () { document.forms[0].submit(); }
```

This assumes of course that the page contains a single form element (forms[0]).

Here the Service user has a security context on the identity provider and wishes to access a resource at a service provider. The SAML assertion is transported to the service provider using the HTTP POST binding.

5.3 Assumptions made

- We can use simpleSAMLphp as an Identity provider
- Pre-generated assertions thus Idp initiated -> POST binding

6. About simplesamlphp

SimpleSAMLphp as Identity Provider

SimpleSAMLphp is a simple application written in native PHP that deals with authentication. SimpleSAMLphp supports several federation protocols, authentication mechanisms and can be used both for local authentication, as a service provider or as an identity provider.

If you have a storage of users, a database, a LDAP or a radius interface, you can setup a installation of simpleSAMLphp to have your own federated Single Sign-On environment.

If you run SimpleSAMLphp as an Identity Provider both Shibboleth and SAML 2.0 services may connect to you.

You may use one of the following included authentication modules or you can very simply make your own:

- Simple LDAP
- Multiple LDAP - Select organization, and connect to different LDAP for each organization.
- CAS remote authentication lets you connect authentication to your existing CAS service, and subsequently retrieve attributes from LDAP.
- Radius authentication lets to check the credentials against a Radius server
- MySQL authentication implemented but not part of the stable release.

In SimpleSAMLphp

to obtain assert token (service user request for assert token to identity provider)

$assertToken = $serviceUser->getRequestToken ($serviceProviderURL);

For assert token verification (service provider verfies with identity provider regarding obtained assert token)

$userdata = $serviceUser->getUserInfo ($serviceProviderURL, $assertToken);

note: include the files '/lib/_autoload.php' and '/libextinc/OAuth.php'

Or identity provider should be developed from scrap

License – Create Common GNU Lesser General Public License

Challenges and constraints with simpleSAMLphp

Integration of openemr with simpleSAMLphp and their communication

Identifies the function to integrate with simpleSAMLphp by the openemr

Identifies the function to integrate with simpleSAMLphp by the Service Provider

7. HIMSS & XUA standard

HIMSS has examined the MUO requirements and commented the following for XUA

- There is not sufficient infrastructure to support this requirement today
- It suggest that for 2011, requirement should be only mutual authentication of endpoints of a network communication to assure that systems only talk to authorized systems that have proven operationally that they can enforce necessary access controls and audit logs.
- Authentication should be system – system based instead of user based authentication
- clarification of whether “receiver” is only external or also includes internal is recommended

8. References

http://www.hitsp.org/ConstructSet_Details.aspx?&PrefixAlpha=4&PrefixNumeric=19

ftp://ftp.ihe.net/IT_Infrastructure/iheitiyr5-2007-2008/Technical_Cmte/Profile_Work/XUA/IHE_ITI_TF_Supplement_XUA_TI_Approved_2007-07-26.doc

http://www2.audaxis.com/roller/eric/entry/sso_integration_with_simplesamlphp_and

http://www.sugarcrm.com/forums/showthread.php?t=42879

http://www.himss.org/content/files/HIMSSresponseONC_IFR.pdf

https://iheprofiles.projects.openhealthtools.org/profiles/xua.html

http://wiki.ihe.net/index.php?title=Cross-Enterprise_User_Assertion

SimpleSAMLphp

a. http://rnd.feide.no/simplesamlphp/
b. http://simplesamlphp.org/