OASIS SSTC Bindings Model

Prateek Mishra, Netegrity
Bob Blakley, Tivoli
Scott Cantor, Ohio State University
Marlena Erdos, Tivoli
Chris Ferris, SUN Microsystems
Simon Godik, Crosslogix
Jeff Hodges, Oblix
Tim Moses, Entrust
Bob Morgan, University of Washington
Evan Prodromou, Securant
Krishna Sankar, Cisco

draft-sstc-bindings-model-06.doc

9 November 2001

79 # 1 Revision History

Revision	Date	Author	1.1.1.1.1.1 Title
0.5	18 August 2001	Prateek Mishra	Bindings model draft
0.6	8 November 2001	Prateek Mishra	Removed SAML HTTP binding, removed artifact PUSH case, updated SOAP profile based on Blakley note

80

81

82

83

84

85

2 Introduction

2.1 Scope

Other Oasis Security Services TC subcommittees (e.g. Core Assertions and Protocol) are producing a specification of SAML security assertions and one or more SAML request-response message exchanges.

The high-level goal of this document is to specify how:

(1) SAML request-response message exchanges are mapped into standard messaging or communication protocols. Such mappings are called SAML *protocol bindings.* An instance of mapping SAML request-response message exchanges into a specific protocol <FOO> is termed a *SAML <FOO> binding*.

Example: A SAML HTTP binding describes how SAML Query and Response message exchanges are mapped into HTTP message exchanges. A SAML SOAP binding describes how SAML Query and Response message exchanges are mapped into SOAP message exchanges.

(2) SAML security assertions are embedded in or combined with other objects (e.g. files of various types, protocol data units of communication protocols) by an originating party, communicated from the originating site to a destination, and subsequently processed at the destination. A set of rules describing how to embed and extract SAML assertions into a framework or protocol is termed a *profile* for SAML. A set of rules for embedding and extracting SAML assertions into a specific class of <FOO> objects is termed a *<FOO> profile* for SAML.

Example: A SOAP profile for SAML describes how SAML assertions may be added to SOAP messages, the interaction between SOAP headers and SAML assertions, description of SAML-related error states at the destination.

(1) and (2) MUST be specified in sufficient detail to yield interoperability when independently implemented.

2.2 Contents

The remainder of this document is in four sections:

- Guidelines for the specification of protocol bindings and profiles. The intent here is to provide a checklist that MUST or SHOULD be filled out when developing a protocol

binding or profile for a specific protocol or framework.

- A process framework for describing and registering proposed and future protocol bindings and profiles.

- Protocol bindings for selected protocols. Bindings MUST be specified in enough detail to satisfy the inter-operability requirement.

- Profiles for selected protocols and frameworks. Profiles MUST be specified in enough detail to satisfy the inter-operability requirement.

2.3 Guidelines for Specifying Protocol Bindings and Profiles

Issues that MUST be identified in each protocol binding and profile:

(1) Each binding or profile must be characterized as set of interactions between parties. Any restriction on applications used by each party and the protocols involved in each interaction must be explicitly called out.

(2) Identification of parties involved in each interaction: how many parties are involved in the interaction? Can intermediaries be involved?

(3) Authentication of parties involved in each interaction: Is authentication required? What types of authentication are acceptable?

(4) Support for message integrity: what mechanisms are used to ensure message integrity?

(5) Support for Confidentiality: can a third party view the contents of SAML messages and assertions? Does the binding or profile require confidentiality? What mechanisms are recommended for securing confidentiality?

(6) Error states: characterization of error states at each participant, especially those that receive and process SAML assertions or messages.

(7) Support for *integrity of assertion attachment*. Many profiles consist of a set of rules for adding assertions to an existing protocol or packaging framework. These rules will be used by an originating party (e.g., user, server) to create a *composite package* consisting of assertions and a business payload for delivery to a destination. When the composite package arrives at the destination, the recipient will require proof (1) the originating party is the subject of the assertions contained within the composite package, (2) neither the assertion nor business payload have been altered.

The term *integrity of assertion attachment* refers to the linkage between the originating party, assertions and business payload, created when an originating party constructs the composite package. Integrity of assertion attachment MUST be verifiable by a recipient. Typically, mechanisms provided to support attachment integrity will be based on some cryptographic techniques (hash or digital signature).

2.4 Process Framework for Describing and Registering Protocol Bindings and Profiles

When a profile or protocol binding is registered, the following information is supplied:

1. Identification: specify a URI that authoritatively identifies this profile or protocol binding.

2. Contact information: specify the postal and electronic contact information for the author of the profile or protocol binding.

3. Description: the description MUST follow the guidelines for profiles and protocol bindings given above.

4. Updates: references to previously registered profiles or bindings that the current entry improves or obsoletes.

*ISSUE:[BINDINGS-01] Where should this registry be maintained? It has been proposed that IANA (*http://www.iana.org*) might provide an appropriate forum. Further investigation is required.*

3 Protocol Bindings

3.1 SOAP

SOAP (Simple Object Access Protocol) 1.1 is a standard proposed by Microsoft, IBM, and other contributors for RPC-like interactions using XML. It defines a mechanism for defining messages in XML, and for sending them over HTTP. Since its introduction, it has attracted much attention, and it is expected to provide the foundation for many future Web-based services.

SOAP 1.1 [SOAP1.1] has three main parts. One is a message format that uses an envelope and body metaphor to wrap XML data for transmission between parties. The second is a restricted definition of XML data for making strict RPC-like calls through SOAP, without using a predefined XML schema. Finally, it provides a binding for SOAP messages to HTTP and extended HTTP.

This document describes how to use SOAP to send and receive SAML messages. An additional section of the SAML specification ("SOAP Profile") defines how to use SAML as an authentication mechanism for SOAP. In other words, the former describes using SAML over SOAP, and the latter describes using SAML for SOAP.

Like SAML, SOAP can be used over multiple underlying transports. This document describes protocol independent aspects of the SAML SOAP binding and calls out the use of HTTP protocol as mandatory-to-implement. It includes recomendations for HTTP specifics, including http headers, error reporting, authentication, message integrity, and confidentiality.

3.1.1 Overview.

3.1.1.1 Referenced Namespaces

SOAP envelope namespace:

SOAP-ENV=http://schemas.xmlsoap.org/soap/envelope

SAML core assertions namespace:

saml=http://www.oasis-open.org/committees/security/docs/sstc-schema-assertion.xsd

SAML protocol namespace:

samlp=http://www.oasis-open.org/committees/secutiry/docs/sstc-schema-protocol.xsd

3.1.1.2 Basic Operation

SOAP messages consist of three elements: an envelope, header data, and a message body. SAML messages (<samlp:Request> and <samlp:Response>) are enclosed within the SOAP message body.

SOAP 1.1 also defines an optional data encoding system. This system is not used within the SAML SOAP binding. This means that SAML messages can be transported using SOAP without re-encoding from the "standard" SAML schema to one based on SOAP encoding.

The system model used for SAML conversations over SOAP is a simple request-response model. A sender transmits a SAML <samlp:Request> within the body of a SOAP message to a receiver. The receiver processes the SAML request and returns a <samlp:Response> within the body of another SOAP message.

3.1.2 SOAP Headers

A sender in a SAML conversation over SOAP MAY add arbitrary headers to the SOAP message.

[Rationale: some SOAP software and libraries may add headers to a SOAP message that are out of the control of the SAML-aware process. Also, some headers may be needed for underlying protocols that require routing of messages.]

A receiver in a SAML conversation MUST NOT require any headers for the SOAP message.

[Rationale: requiring extra headers will cause fragmenting of the standard and will hurt interoperability.]

3.1.3 SAML Requests

A SAML request <samlp:Request> is stored as the (only) child of the <SOAP-ENV:body> element of a SOAP message. The sender MUST NOT include more than one SAML request per SOAP message or include any additional XML elements in the SOAP body.

On receiving a SAML request as a SOAP message, the receiver MUST return either a SAML response <samlp:Response> or a SOAP fault code.

3.1.4 SAML Responses

A SAML response <samlp:Response> is stored as the (only) child of the <SOAP-ENV:body> element of a SOAP message. The SOAP message MUST contain exactly one SAML response element. The receiver MUST NOT include any additional XML elements in the SOAP body.

On receiving a SAML response in a SOAP message, the sender MUST NOT send a fault code or other error messages to the receiver.

[Rationale: The format for the message interchange is a simple request-response. Adding additional error conditions, notifications, etc. would needlessly complicate the protocol.]

3.1.5 Fault Codes

If a receiver cannot, for some reason, process a SAML request, it should return a SOAP fault code. Fault codes MUST NOT be sent for errors within the SAML problem domain, e.g. as a signal that the subject is not authorized to access resource in an authorization query.

Section 4.1 of [SOAP1.1] describes SOAP faults and fault codes.

3.1.6 Authentication

Authentication of both sender and receiver is optional and depends on the environment of use. Authentication protocols available from the underlying substrate protocol MAY be utilized to provide authentication. Section 3.1.9.2 describes authentication in the HTTP environment.

3.1.7 Message Integrity

Message integrity of both requests and responses is optional and depends on the environment of use. Security layer in the underlying substrate protocol MAY be used to ensure message integrity.

3.1.8 Confidentiality

Currently SOAP does not specify standard message-oriented technique for confidentiality. This will only be possible when XML encryption standard becomes available. So for the near future we have to depend on facilities provided by the underlying substrate protocol over which SOAP is layered.

Communicating parties MAY encrypt messages if confidentiality is required by the context of use.

3.1.9 HTTP Specifics

The SAML SOAP binding is mandatory to implement.

The HTTP binding for SOAP is described in Section 6.0 of [SOAP1.1]. It requires the use of a SOAPAction header as part of a SOAP HTTP request. A SAML receiver SHOULD NOT depend on the value of this header. A SAML sender MAY set the value of SOAPAction header to "http://www.oasis-open.org/committees/security".

3.1.9.1 HTTP Headers.

When using HTTP 1.1:

(1) a SAML receiver should not include Cache-Control header field in the response to a POST request unless its value is set to no-store.

(2) Expires response header field should not be included, unless it is disabled by Cache-Control header with the value of no-store.

[Rationale: HTTP proxies should not cache POST request responses carrying SAML assertions]

There are no other restrictions on HTTP headers.

3.1.9.2 Authentication

Following authentication protocols MUST be supported:

1. No client authentication.

2. HTTP basic client authentication [rfc2617] with and without SSL.

3. HTTPS server authentication with server-side certificate.

4. HTTPS client authentication with client-side certificate.

The use of server side certificate is mandatory in HTTPS deployment.

ISSUE:[BINDINGS-02] Do we need to support (a) message digest (b) client authentication based on digital signature?

3.1.9.3 Message Integrity

If message integrity is required, HTTPS with server-side certificate MUST be used.

3.1.9.4 Message Confidentiality

If message confidentiality is required, HTTPS with server-side certificate MUST be used.

3.1.9.5 Security Considerations

Each combination of authentication-message integrity-confidentiality should be analyzed for vulnerability in the context of deployment environment.(See security considerations document [saml-sec-cons] for detailed discussion).

[Rfc2617] provides description of possible attacks in HTTP environment using basic and digest authentication schemes.

3.1.9.6 Error reporting

If the receiver refuses to perform a SAML message exchange with the sender it should return a "403 Forbidden" response. In this case content of the HTTP body is undefined.

As described in [SOAP1.1], in case of a SOAP error while processing SOAP request the SOAP HTTP server MUST return a "500 Internal Server Error" response and include a SOAP message in response containing a SOAP Fault element. This type of error should be returned for SOAP related errors detected before control is passed to the SOAP processor, or when the SOAP processor reports an internal error. Examples include situations when soap namespace is incorrect, SAML schema can not be located, SOAP message signature does not validate, SAML processor runs out of memory, etc.

In case of a SAML processing error the SOAP HTTP server MUST respond with "200 OK" and include SAML specified error description as the only child of the SOAP-ENV:Body element. For complete list of SAML error codes see [SAML-CoreDoc].

3.1.9.7 Example: SAML over SOAP/HTTP

REQUEST:

```
POST /SamlService HTTP/1.1
Host: www.whatever.com
Content-Type: text/xml
Content-Length: nnn
SOAPAction: "SAML-URI"

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
        <SOAP-ENV:Body>
                <samlp:Request xmlns:samlp="..." xmlns:saml="..."
                xmlns:ds="...">
                        <ds:Signature> ... </ds:Signature>
```

```
                        <samlp:AuthenticationQuery>
                                ...
                        </samlp:AuthenticationQuery>
                </samlp:Request>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

RESPONSE:

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: nnnn

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
        <SOAP-ENV:Body>
                <samlp:Response xmlns:samlp="..." xmlns:saml="..."
                xmlns:ds="..." samlp:StatusCode="Success">
                        <ds:Signature> ... </ds:Signature>
                        <saml:AssertionSimple>
                                <saml:AuthenticationStatement>
                                        ...
                                </saml:AuthenticationStatement>
                        </saml:AssertionSimple>
                </samlp:Response>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

4 Profiles

4.1 Web Browser

4.1.1 Background

The web browser profile utilizes terminology taken from Use Case 1 and Scenario 1-1. In this use-case, a web user authenticates with a *source site*. The web user then uses a secured resource at a destination site, without directly authenticating to the *destination site*.

We assume that the user is utilizing a standard commercial browser and has authenticated to a source site. Further, the source site has some form of security engine in place that can track locally authenticated users [WEB-SSO]. Typically, this takes the form of a session which may be represented by an encrypted cookie or an encoded URL or by the use of some other technology [SESSION]. This is a substantial requirement but one which is met by a large class of security engines.

At some point, the user attempts to access a *target* resource available from the destination site and subsequently through one or more steps (e.g., re-direction) arrives at an *inter-site transfer service* at the source site. Starting at this point, the SAML web browser profiles describe a canonical sequence of HTTP protocol exchanges that transit the user browser to a distinguished *assertion consumer service* at the destination site. Information about *SAML assertions* associated with the user and the desired target are conveyed, from the source to the destination site, by the protocol exchange.

The destination site can examine both the assertions and target information and determine whether to allow access to the target resource, thereby achieving web single sign-on for authenticated users originating from the source site. Often, the destination site also utilizes a standard security engine that will create and maintain a session, possibly utilizing information contained in the source site assertions, for the user at the destination site.

4.1.2 Relevant Technology

We describe two HTTP-based techniques available for conveying information from one site to another via a stock commercial browser. We do not discuss the use of cookies, as these impose the limitation that both the source and destination site belong to the same "cookie domain".

- *Form POST*: SAML assertions are uploaded to the user browser within a HTML Form [HTML] and conveyed to the destination site as part of a HTTP POST payload when the user "submits" the form,

- *SAML Artifact*: A "small", bounded-size SAML artifact, which unambiguously identifies an assertion to the source site, is carried as part of a URL query string and conveyed via re-direction to the destination site; the destination site must acquire the referenced assertion by some further steps. Typically, this involves the use of a registered SAML protocol binding.

The need for a ``small'' SAML artifact is motivated by restrictions on URL size imposed by commercial web browsers. While [RFC2616] does not specify any restrictions on URL length, in practice commercial web browsers and application servers impose size constraints on URLs (maximum size of 2000 characters [Appendix A]). Further, as developers will need to estimate and set aside URL ``real-estate'' for the artifact, it is important that the artifact have a bounded size (predefined maximum size). These measures ensure that the artifact can be reliably carried as part of the URL query string and thereby transferred from source to destination site.

4.1.3 SAML artifact structure

Depending on upon the level of security desired and associated profile protocol steps, many viable architectures may be proposed for the SAML artifact ([Core-Assertions-Examples, Shib-Marlena]. We accommodate variability in the architecture by a mandatory two byte artifact type code in the representation:

```
<SAML_artifact> :=
         B64 representation of <TypeCode> <RemainingArtifact>
         <TypeCode> := Byte1Byte2
```

The following MANDATORY-TO-IMPLEMENT fixed size artifact architecture has the property that it is simple to implement but at the same time its use has adequate safeguards against attacks such as artifact forgery, browser state exposure and impersonation.

```
<TypeCode> := 0x0001
<RemainingArtifact> := <SourceID> <AssertionHandle>
<SourceID> := 20 byte sequence
<AssertionHandle> := 20 byte sequence
```

<SourceID> is a twenty byte sequence used by the destination site to determine source site identity. We assume that the destination site will maintain a table of sourceID values as well as the URL (or address) for the corresponding SAML query service. This information is communicated between the source and destination sites using an out-of-band technique. On receiving the SAML artifact, the destination site determines if the <SourceID> belongs to a valid partner, retrieves the "assertion lookup" service information and invokes the service with the <SAML_artifact> and other values as an argument.

The following practices are RECOMMENDED for the creation of SAML artifacts at source sites:

(1) Each source site selects a single *Identification URL* which it communicates to all potential destination sites. The domain name used within the identification URL MUST be administered by source site.

(2) The source site constructs the <SourceID> component of the artifact by taking the SHA-1 [SHA-1] hash of the identification URL.

(3) Construction of <AssertionHandle> values is governed by the principle that it should have no predictable relationship to the contents of the referenced assertion at the source site and must also be difficult to "guess". Use of either one of the following techniques is RECOMMENDED:

(a) the value is taken from a random number sequence [RFC1750] generated by the source site. The sequence must consist of values of size at least eight bytes.

(b) the value is taken from the SHA-1 hash of a sequence of distinct values generated by the source site.

4.1.4 Profile Overview

In this section, we describe two distinct web browser profiles: one based on a SAML artifact and one based on form POST. For each type of profile, a section describing the threat model and relevant counter-measures is also included.

Two types of information may be communicated through the web browser profiles:

(1) information about the "target" of interest to the user. This is essentially some contextual information originating from the source web site. Typically, this takes the form of a URL at the destination web site but more generally it could take the form of a category or resource name. The destination site may use the target information to present an appropriate category of resources to the user (e.g., redirect to the target URL) once sign-on completes.

(2) information describing one or more SAML assertions.

4.1.5 SAML Artifact

This profile consists of a single interaction between three parties (source site, user equipped with a browser, destination site), with a nested sub-interaction between two parties (source site, destination site). We refer to the sub-interaction as an *assertion pull* interaction. The interaction sequence is diagrammed in Figure 1.

The user has authenticated to the source web site and subsequently visits an inter-site transfer URL with information about the desired target on the URL query string (step (1)). As this step is over the open internet, confidentiality of the query string MUST be maintained. One way of achieving this is to have the inter-site transfer URL exposed over HTTPS (HTTP over server-side SSL). Otherwise, the artifact(s) returned on (step (2)) will be available in plain text to any attacker.

The inter-site transfer URL redirects the user (step (2) to the assertion consumer URL with target and one or more SAML artifacts carried on the URL query string.

In response, the user browser attempts to access the assertion consumer URL (step (3)) and delivers both the assertion consumer URL, the SAML artifact(s) and target to (a web server at) the destination site. As this step takes place over the open internet, confidentiality of the query string MUST be maintained. One way of achieving this is to have the destination URL exposed over HTTPS (HTTP over server-side SSL). This is because a SAML artifact represents a bearer

token, and its disclosure may allow an adversary to impersonate the user.

If the destination site is unable to process this information it MUST return a HTTP "400 Bad Request" error code to the browser (step 6)). Otherwise, it MUST carry out the *assertion pull* interaction (steps (4) and (5)) described below, and obtain assertions from the source site.

Thereafter, the destination site may utilize the communicated assertions and target information, further interaction steps with the user and other information to make an access control judgement. If the user is refused access to the desired resource, the destination site MUST return a HTTP “403 Forbidden” error code to the browser (step (6)).

The assertion pull interaction consists of a SAML message exchange between source and destination site (steps (4) and (5))) utilizing a registered SAML protocol binding. The destination site sends a <*samlp:Request*> message to the source site, containing all of the SAML artifacts delivered to the destination site (step (3)). If the source site can find or construct the requested assertions it responds with a <*samlp:Response*> message with the requested assertions. Otherwise, it returns an "assertion not found" error to the destination site.

The selected SAML protocol binding for assertion pull MUST provide confidentiality and bilateral authentication. The source site MUST implement the SAML SOAP binding with support for confidentiality (HTTPS); support for other protocol bindings is not mandatory.

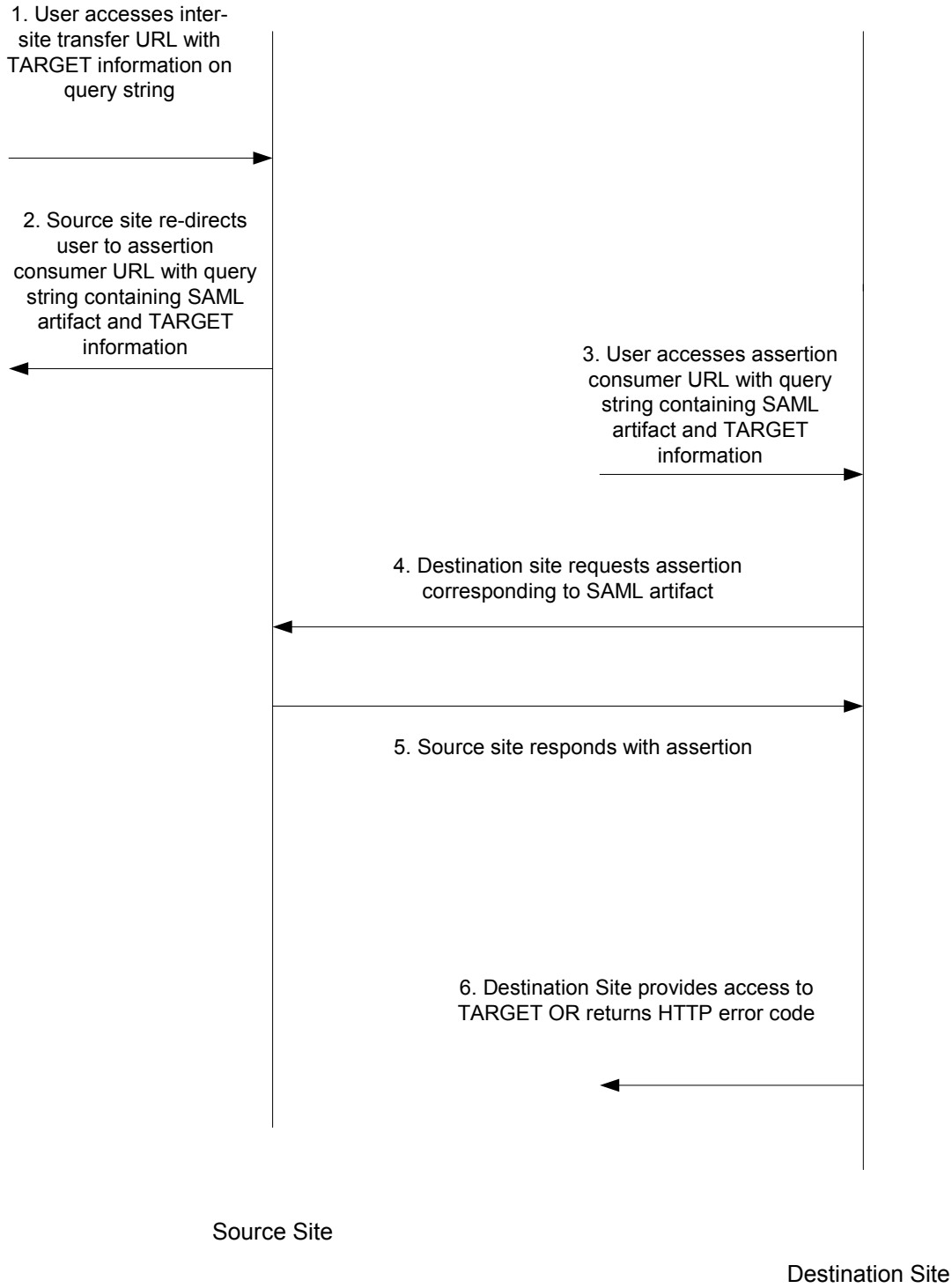

Figure 1: web Browser Profile: SAML Artifact (Pull)

546

Action	HTTP
(1)	GET https://www.example.com/<inter-site-transfer URL>?TARGET=<target>..
(2)	HTTP 1.1 302 GET https://destination_URL?SAMLart=<artifact body>?TARGET=<target>..
(3)	GET https://destination_URL?SAMLart=<artifact body>?TARGET=<target>..
(4)	<samlp:Request> message sent to source site and with artifacts utilizing a selected SAML protocol binding.
(5)	<samlp:Response> message with assertions is returned to destination site over selected protocol binding.
(6)	User is given access to TARGET OR "400 Bad Request" is returned OR "403 Forbidden" is returned

The source and destination sites MUST implement the following additional restrictions when processing SAML artifacts:

1. The SAML artifact MUST be "one-time use"; once the user completes step (6) above, any repitition of step (3) MUST fail with the destination site returning HTTP code "403 Forbidden".

2. The source site MUST implement a "one-time request" property for any SAML artifact. Many simple implementations meet this constraint: for example deleting the relevant assertion from persistent storage at the source site after first lookup.

3. A successful <samlp:Response> message is returned from the source site only if the <samlp:Request> message originates from the destination site to whom the artifact was issued. Thus, step (4) above would complete successfully at most once and only if originating from the (unique) destination site.

4. SAML assertions returned to the destination site MUST include at least one authentication statement. An assertion containing an authentication statement MUST include a <saml:Audience> element.

5. The <saml:ConfirmationMethod> element of each assertion MUST be set to SAML Artifact (5.1.1 of [Core-20]).

4.1.5.1 Threat Model and Counter-Measures

This section utilizes materials from [Shib-Marlena].

4.1.5.1.1 Stolen artifact

1. If a malicious user (MAL) can copy the real user's SAML artifact, then the MAL could construct a URL with the real user's SAML artifact and be able to impersonate the user at the destination site.

   Counter-Measure:

   SAML assertions communicated through a web browser profile must always include a SAML authentication statement. An authentication statement communicated through a web browser profile MUST include (1) issue instant and (2) validity period. It MAY include the IP address of the user.

   Source and destination sites MUST make some reasonable effort to ensure that clock settings are both sites differ by at most a few minutes. Many forms of time service are available, both over the internet and from proprietary sources.

   RECOMMENDATIONS for Source Site (Asserting party):

   (a) Source sites should track the time at which a SAML artifact is generated and when the destination site "calls back" for an assertion. A maximum time limit of a few minutes is recommended. Should an assertion be requested by a destination site after this time limit a SAML error should be returned by the source site.

   (b) Assertions containing authentication statements may be created by the source site either when the corresponding SAML artifact is created or when the destination site "calls back" for an assertion. In each of these cases, the validity period of the assertion will need to be set differently.

   (c) Issue instant and validity period of assertions with authentication statements should have the shortest possible validity period consistent with successfully communication of the assertion from source to destination site. This is typically of the order of a few minutes.

   RECOMMENDATIONS for Destination Site (Relying Party):

   (a) The destination site MUST check the (1) issue instant and (2) validity period of assertions obtained from the source site and reject expired assertions. A destination site may choose to implement a stricter test of validity for assertions containing authentication statements, such as for example, requiring the issue instant of the assertion to be within a few minutes of the time at which the assertion is received at the destination site.

   (b) The destination site MUST check the browser IP address against the IP address

contained in the assertion statement (if available).

(c) The destination site MUST correlate the value of assertion Issuer attributes against the credentials obtained from the source site during the assertion pull interaction.

2. Since the destination site obtains assertions from the source site with <ConfirmationMethod> element set to "SAML artifact", a malicious site could impersonate the user at some "new" destination site. The new destination site would believe the malicious site to be the user.

   Counter-Measure:

   The new destination site MUST obtain the SAML assertions corresponding to the SAML artifacts from the source site through a bilaterally authenticated channel. This ensures that the malicious site cannot simulate the original source site when communicating with new destination site.

4.1.5.1.2 Forged SAML artifact

A MAL could forge a SAML artifact.

Counter-Measure:

A SAML artifact must be constructed in such a way that it is very hard to guess and Section 4.1.3 provides specific recommendations in this space. A MAL could attempt to repeatedly "guess" a valid SAML artifact value (one that corresponds to an existing assertion at a source site) but given the size of the value space would likely require a very large number of failed attempts.

4.1.5.1.3 Browser State Exposure

The SAML artifact profile involves "upload" of SAML artifacts to the web browser from a source site. This information is available as part of the web browser state and is usually stored in persistent storage on the user system in a completely unsecured fashion. The threat here is that the artifact may be "re-used" at some later point in time.

Counter-Measure: The "one-use" property of SAML artifacts ensures that they may not be re-used from a browser.

4.1.6 Form POST

Figure 2 provides a description of a web browser profile based upon the use of "POST" to convey SAML assertions from source to destination site [S2ML, Anders-Browser-Profile]. An

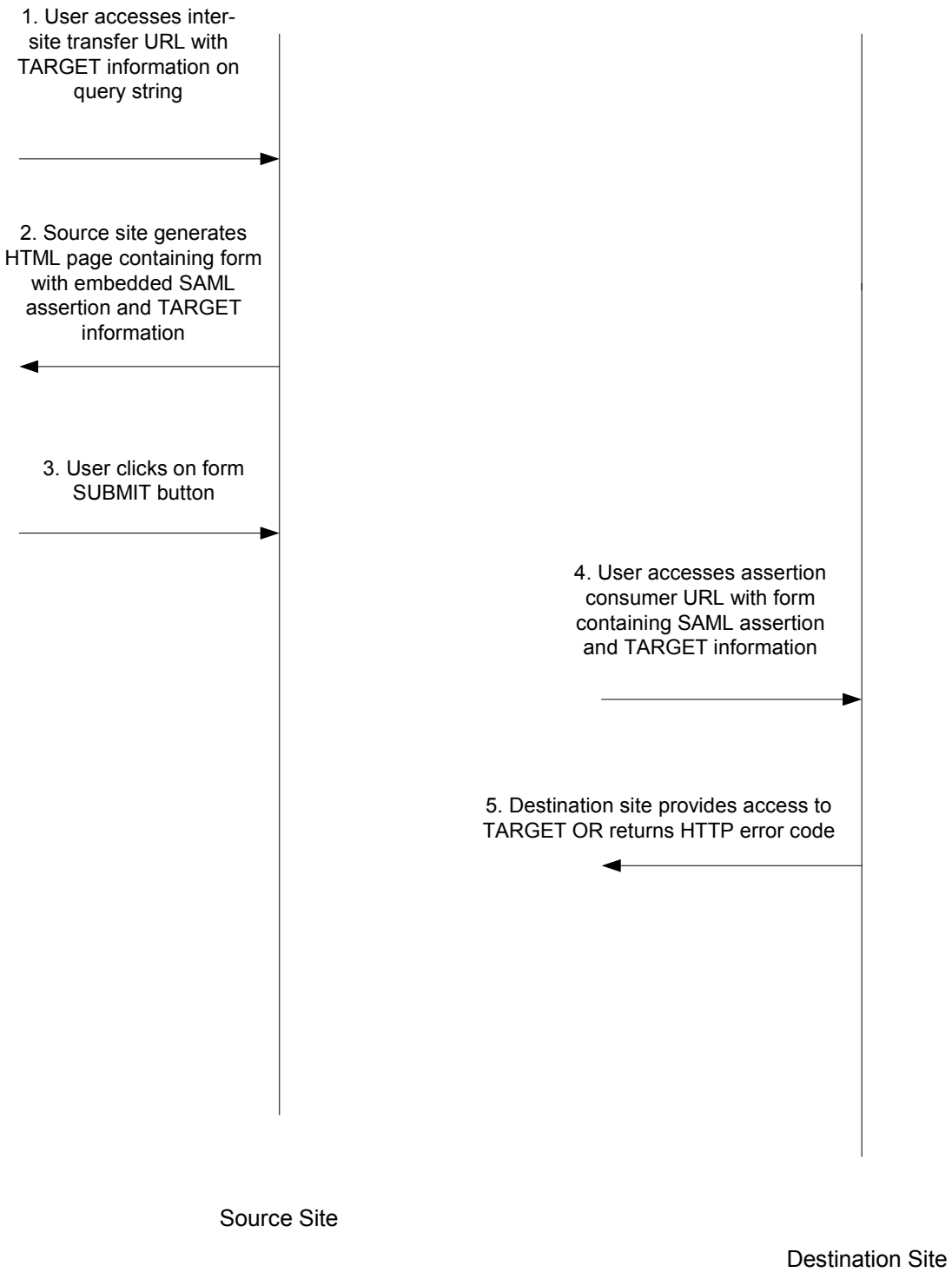

Figure 2: Web Browser Profile: POST

653

authenticated user visits an inter-site transfer URL with information about the target as part of the URL query string (step (1)). The source site generates an HTML page containing a form with one or more embedded SAML assertions and target information (step (3)). The user browser "clicks on" the form SUBMIT button and navigates to the assertion consumer URL at the destination site (step (4)). The destination site scrutinizes the posted assertion and target information and determines whether to allow the user access to the target resource (step (5)).

As interactions with the inter-site transfer and assertion consumer URLs is over the open internet, confidentiality of assertions MUST be preserved. One way to achieve this to have both URLs be exposed over HTTPS (HTTP over server-side SSL).

Action	HTTP
(1)	GET Https://www.example.com/<inter-site-transfer URL>?TARGET=<target>..
(2)	HTTP 1.1 Content-Type: application/x-www-form-urlencoded Content-length:… <BODY> <FORM METHOD="post" ACTION="assertion_consumer_URL"> <INPUT TYPE="submit" NAME="button" VALUE="submit"> <INPUT TYPE="hidden" NAME="SAMLAssertion" VALUE="B64(SAML Assertion)"> <INPUT TYPE="hiddent" NAME="TARGET" VALUE="<target>"> </FORM> </BODY>
(3)	This step may be eliminated in a Javascript-enabled browser. See Appendix B.
(4)	POST assertion_consumer_URL [standard POST payload corresponding to form in (2)]
(5)	User is given access to TARGET OR "403 Forbidden" is returned

Notes:

1. All SAML assertions communicated to the destination site using the POST web browser profile MUST be digitally signed by the issuing party.

2. The destination site MUST ensure a "single use" policy for assertions containing an authentication statement communicated using form POST. The implication here is that the destination site will need to be stateful. A simple implementation maintains a table of pairs:

   Assertion Id, Time at which entry is to be deleted

The time at which an entry is to be deleted is based upon the authentication assertion lifetime. Assertions containing authentication statements are recommended to have short lifetimes in the web browser context, such a table would be of manageable size.

3. The <saml:ConfirmationMethod> element of each assertion MUST be set to Assertion Bearer (5.1.2 of [Core-20]).

4. SAML assertions included in a POST body MUST include at least one authentication statement. An assertion containing an authentication statement MUST include a <saml:Audience> element.

4.1.6.1 Threat Model and Counter-Measures

This section utilizes materials from [Shib-Marlena].

4.1.6.1.1 Stolen assertion

1. If a malicious user (MAL) can copy the real user's SAML assertion (Form POST), then the MAL could construct an appropriate POST body and be able to impersonate the user at the destination site.

   Counter-Measure: SAML assertions communicated through a web browser profile must always include a SAML authentication statement. An authentication statement communicated through a web browser profile MUST include (1) issue instant and (2) validity period. It MAY include the IP address of the user.

   Source and destination sites MUST make some reasonable effort to ensure that clock settings are both sites differ by at most a few minutes. Many forms of time service are available, both over the internet and from proprietary sources.

   RECOMMENDATIONS for Source Site (Asserting party):

   (a) Issue instant and validity period of assertions with authentication statements should have the shortest possible validity period consistent with successfully communication of the assertion from source to destination site. This is typically of the order of a few minutes.

   RECOMMENDATIONS for Destination Site (Relying Party):

   (a) The destination site MUST check the (1) issue instant and (2) validity period of assertions obtained from the source site and reject expired assertions. A destination site may choose to implement a stricter test of validity for assertions containing authentication statements, such as for example, requiring the issue instant of the assertion to be within a few minutes of the time at which the assertion is received at the destination site.

(b) The destination site MUST check the browser IP address against the IP address contained in the assertion statement (if available).

(c) The destination site MUST check the digital signature of all assertions obtained through the POST profile to ensure that the assertion originates from the assertion issuer.

3. Since the destination site obtains "bearer" SAML artifacts or SAML assertions from the user via a web browser profile, a malicious site could impersonate the user at some "new" destination site. The new destination site would believe the malicious site to be the user.

   Counter-Measure:

   (a) SAML artifact: The destination site must check the <saml:Audience> elements to ensure that at least one of their values matches the destination site expectations. It is strongly recommended that assertions communicated through the web browser profile have extremely "narrow" values for this field (e.g., each destination site has a unique <saml:Audience> value). As the assertion is digitally signed, the <saml:Audience> value cannot be altered by an intermediary.

4.1.6.1.2 Forged Assertion

A MAL could forge a SAML assertion (form POST).

Counter-Measure: The POST browser profile requires SAML assertions to be signed, thus providing both message integrity and authentication. The destination site must always verify the signature and ensure that it corresponds to the assertion issuer.

4.1.6.1.3 Browser State Exposure

The POST browser profile involve upload of assertions to the web browser from a source site. This information is available as part of the web browser state and is usually stored in persistent storage on the user system in a completely unsecured fashion. The threat here is that the assertion may be "re-used" at some later point in time.

Counter-Measure: The form POST case similarly includes a requirement that an assertions with authentication statements cannot be re-presented at the destination site.

4.2 SOAP

4.2.1 Overview

The SOAP profile for SAML is based on a single interaction between a sender and a receiver. The sender adds with one or more SAML assertions to a SOAP document and sends the message to the receiver. The receiver extracts the SAML assertion from the message and processes them. It may either return an error or go on to process the message in the standard way. The message may be sent over any protocol for which a SOAP protocol binding is available [SOAP1.1].

SOAP provides a flexible header mechanism [SOAP1.1], which may be (optionally) used for extending SOAP payloads with additional information. A header entry is identified by its fully qualified element name, which consists of the namespace URI and the local name. All immediate child elements of the SOAP Header element MUST be namespace-qualified.

4.2.2 SOAP Headers and Error Processing

SAML assertions MUST be contained within the SOAP `<Header>` element contained within the SOAP `<Envelope>` element. Two standard SOAP attributes are available for use with header elements: `actor` and `mustUnderstand`. Use of the `actor` attribute is application dependent and no normative use is specified herein.

The SOAP `mustUnderstand` global attribute can be used to indicate whether a header entry is mandatory or optional for the recipient to process. SAML assertions MUST have the `mustUnderstand` attribute set to 1; this ensures that a SOAP processor to which the message is directed must be able to successfully process the SAML assertions or return a SOAP message with `<Fault>` element as the message body. The returned `<Fault>` element takes the form:

```
<Fault>
   <Faultcode>mustUnderstand</Faultcode>
   <Faultstring>…</Faultstring>
</Fault>
```

If the receiving party is able to successfully process the attached SAML assertions, and based on their contents does not further process the body of the SOAP message, it MUST return a SOAP message with `<Fault>` element as the message body. The returned `<Fault>` element takes the form:

```
<Fault>
   <Faultcode>Client.SAML</Faultcode>
   <Faultstring>…</Faultstring>
</Fault>
```

It is recommended that the `<FaultString>` element contain a helpful message but this specification does not describe any normative text.

4.2.3 SOAP Profile Architectures

Two SOAP profile architectures for adding assertions to an arbitrary SOAP message are described below. Both architectures are mandatory to implement.

4.2.3.1 HolderOfKey

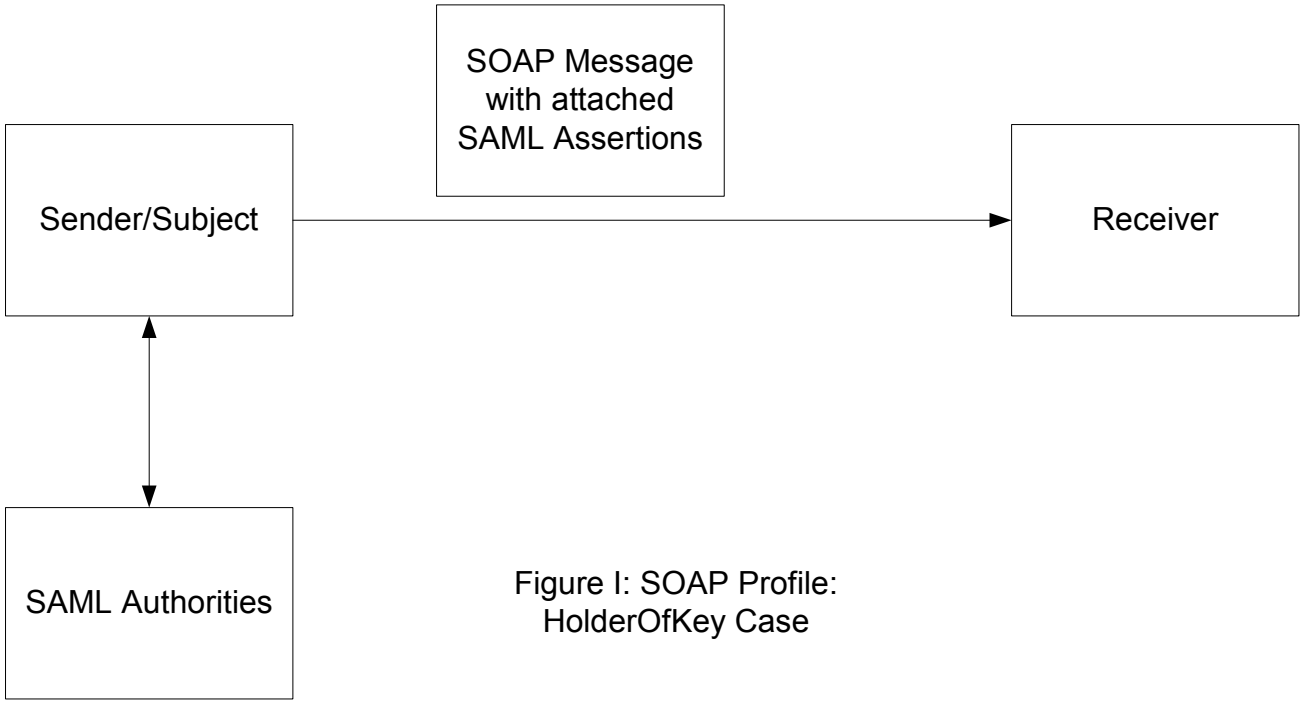

Figure I: SOAP Profile: HolderOfKey Case

4.2.3.1.1 Sender

In this case, the sender and subject are the same entity. The sender obtains one or more assertions from one or more authorities. Each assertion MUST have the following characteristics:

(1) Each assertion MUST be signed by the issuer.

(2) Each assertion MUST include the following <SubjectConfirmation> element:

```
<SubjectConfirmation>
  <AuthenticationMethod>HolderOfKey</AuthenticationMethod>
  <dsig:KeyInfo>…<dsig:KeyInfo>
 </SubjectConfirmation>
```

In this technique, the <SubjectConfirmation> element carries information about the sender's key within the `<dsig:KeyInfo>` element. The `<dsig:KeyInfo>` provides varied ways for describing information about the sender's public or secret key.

Each assertion is added to the SOAP `<Header>` element as described above. In addition, the sender MUST include an enveloped digital signature `<dsig:Signature>` element within the SOAP `<Header>` element utilizing the transform http://www.w3.org/2000/09/xmldsig#enveloped-signature as described in [XML-DSIG]. The `<dsig:Signature>` element MUST include all of the elements within the SOAP message including all headers, assertions and the business payload.

4.2.3.1.2 Receiver

The receiver MUST verify that each assertion carries a <SubjectConfirmation> element of the form:

```
<SubjectConfirmation>
  <ConfirmationMethod>HolderOfKey</ConfirmationMethod>
  <dsig:KeyInfo>…<dsig:KeyInfo>
 </SubjectConfirmation>
```

The receiving party MUST check the validity of the signature found in the `<SOAP:Envelope>/<dsig:Signature>` element. Information about the sender's public or secret key may be found in the `<saml:SubjectConfirmation>/<dsig:KeyInfo>` element carried within each assertion.

Notice the `<ds:KeyInfo>` element is used only for checking integrity of assertion attachment (message integrity). Therefore, there is no requirement that the receiver validate the key or certificate. This suggests that, if needed, senders may generate public/private key pairs and utilize them for this purpose.

Once the above steps are complete, the receiver may further process the assertions and SOAP message contents with the assurance that (a) the SOAP message has been constructed by the subject, and (b) neither the assertions nor the enclosing SOAP message have been altered by an intermediary.

4.2.3.1.3 Example

The following example illustrates the HolderOfKey architecture for adding SAML assertions to a SOAP message:

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV:Header>
        <saml:AssertionList mustUnderstand="1"
        AssertionID="192.168.2.175.1005169137985" IssueInstant="2001-11-07T21:38:57Z"
        Issuer="M and M Consulting" MajorVersion="1" MinorVersion="0"
        xmlns:saml="http://… /security/docs/draft-sstc-schema-assertion-16.xsd">
        <saml:Conditions NotBefore="2001-11-07T21:33:57Z"
                                NotOnOrAfter="2001-11-07T21:48:57Z">
        <saml:AbstractCondition
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:type="AudienceRestrictionConditionType">

        <saml:Audience>http://www.example.com/research_finance_agreement.xml
         </saml:Audience>
        </saml:AbstractCondition>
        </saml:Conditions>
        <saml:AuthenticationStatement AuthenticationInstant="2001-11-07T21:38:57Z"
                                        AuthenticationMethod="Password">
        <saml:Subject>
            <saml:NameIdentifier Name="goodguy" SecurityDomain="www.example.com"/>
             <saml:SubjectConfirmation>HolderOfKey</SubjectConfirmation>
             <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
                        <KeyValue>
                        ...
                        </KeyValue>
                        <X509Data>
                        ...
                        </X509Data>
             </KeyInfo>
        </saml:Subject>
        <saml:AuthenticationLocality DNSAddress="some_computer" IPAddress="111.111.111.111"/>
        </saml:AuthenticationStatement>
        <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
                <SignedInfo>
                <CanonicalizationMethod
                Algorithm="http://www.w3.org/TR/2000/WD-xml-c14n-20000119"/>
                <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
```

```
                <Reference URI="">
                <Transforms>
                <Transform
                Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                </Transforms>
                <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                <DigestValue>GSUvQSPfYkAC9wpHbLSfPEjMlIo=</DigestValue>
                </Reference>
                </SignedInfo>
                <SignatureValue>
                iLJj64yusw7h4FTbiyKRvAQoALlmeCnKxhKqStrFahVXIZUXacmDJw==
                </SignatureValue>
                <KeyInfo>
                <KeyValue>
                ...
                </KeyValue>
                <X509Data>
                ...
                </X509Data>
        </KeyInfo>
        </Signature>
        </saml:AssertionList>
        <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
                <SignedInfo>
                <CanonicalizationMethod
                Algorithm="http://www.w3.org/TR/2000/WD-xml-c14n-20000119"/>
                <SignatureMethod
                Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
                <Reference URI="">
        <Transforms>
                <Transform
                Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
        </Transforms>
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>UYRsLhRffJagF7d+RfNt8CPKhbM=</DigestValue>
                </Reference>
                </SignedInfo>
                <SignatureValue>
                 HJJWbvqW9E84vJVQkjjLLA6nNvBX7mY00TZhwBdFNDEIgscSXZ5Ekw==
                </SignatureValue>
        </Signature>
</SOAP-ENV:Header>
```

```
<SOAP-ENV:Body>
        <ReportRequest>
        <TickerSymbol>SUNW</TickerSymbol>
        </ReportRequest>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

4.2.3.2 SenderVouches

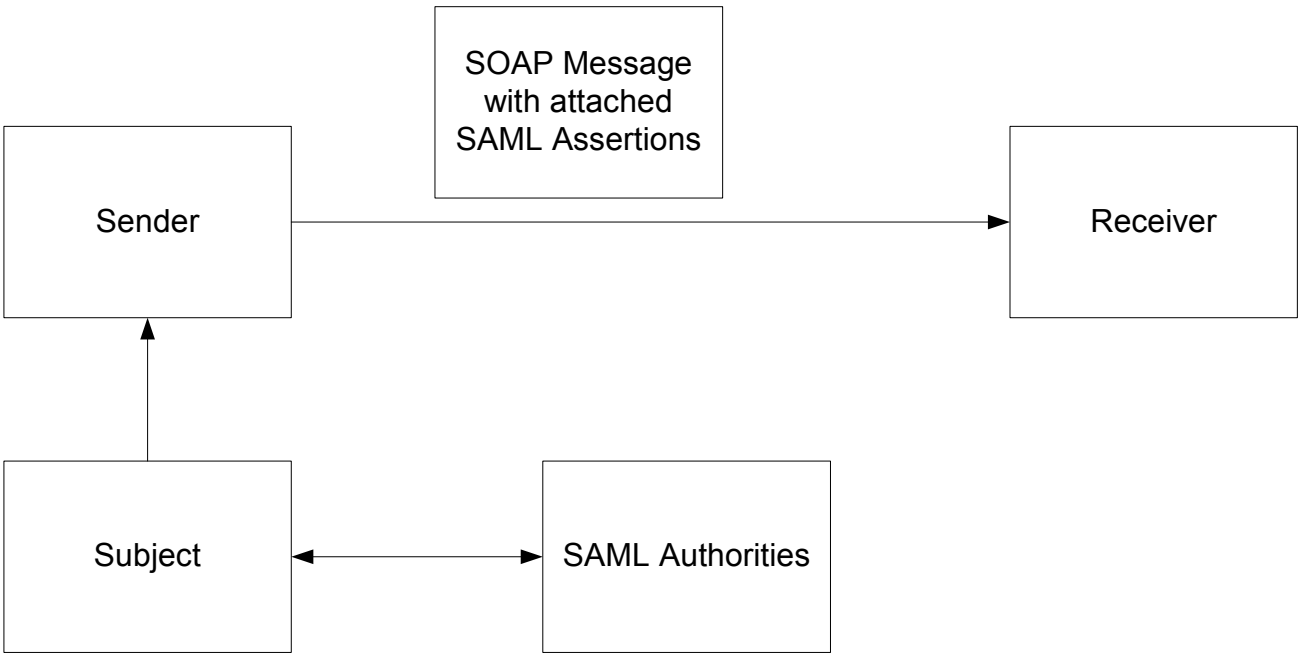

Figure 2:
SOAP Profile:
SenderVouches Case

4.2.3.2.1 Sender

In this case, the sender and subject may be distinct entities. The subject obtains one or more assertions from one or more authorities. Each assertion MUST have the following characteristics:

(3) Each assertion MUST be signed by the issuer.

(4) Each assertion MUST include the following <SubjectConfirmation> element:

```
<SubjectConfirmation>
  <AuthenticationMethod>SenderVouches</AuthenticationMethod>
 </SubjectConfirmation>
```

The assumption here is that the subject provides the sender with the assertions, which the sender attaches to a SOAP payload through a signing act. In this model, information about the sender's key is held within the `<dsig:KeyInfo>` element associated with the senders signature. The `<dsig:KeyInfo>` provides varied ways for describing information about the sender's public or secret key.

Each assertion is added to the SOAP `<Header>` element as in the HolderOfKey case. In addition, the sender MUST include an enveloped digital signature `<dsig:Signature>` element within the SOAP `<Header>` element utilizing the transform http://www.w3.org/2000/09/xmldsig#enveloped-signature as described in [XML-DSIG]. The `<dsig:Signature>` element MUST include all of the elements within the SOAP message including all headers, assertions and the business payload. The sender MUST also include a `<dsig:KeyInfo>` element with the `<dsig:Signature>` element.

4.2.3.2.2 Receiver

The receiver MUST verify that each assertion carries a <SubjectConfirmation> element of the form:

```
<SubjectConfirmation>
  <ConfirmationMethod>SenderVouches</AuthenticationMethod>
 </SubjectConfirmation>
```

The receiving party MUST check the validity of the signature found in the `<SOAP:Envelope>/<dsig:Signature>` element. Information about the sender's public or secret key may be found in the `<SOAP:Envelope>/<dsig:Signature>/<dsig:KeyInfo>` element carried within each assertion.

Once the above steps are complete, the receiver may further process the assertions and SOAP message contents with the assurance that (a) the sender (identified by `<SOAP:Envelope>/<dsig:Signature>/<dsig:KeyInfo>`) constructed the SOAP message, (b) neither the assertions nor the enclosing SOAP message have been altered.

4.2.3.2.3 Example

The following example illustrates the SenderVouches architecture for adding SAML assertions to a SOAP message:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV=http://schema.xmlsoap.org/soap/envelope/>

<SOAP-ENV:Header xmlns:SAML="…">
        <SAML:Assertion  mustUnderstand=1>…</SAML:Assertion>
        <SAML:Assertion mustUnderstand=1>…</SAML:Assertion>
        <dsig:signature>…</signature>
</SOAP-ENV:Header>
…
<SOAP-ENV:Body>
   <message_payload/>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

4.2.4 Confidentiality

In some circumstances, there may be a requirement to ensure confidentiality of SAML assertions. In the near future we would anticipate use of the [XML-Encryption] specification which we would point to as mandatory-to-implement. In the interim, confidentiality has to be ensured by selection of a "substrate" SOAP protocol binding which preserves confidentiality. This would include, for example, HTTPS with server-side certificates or S/MIME.

ISSUE:[BINDINGS-03] The web browser SSO Profile and the SOAP profile require the <ConfirmationMethod> to be set to a specific value. Is this consistent with core-20? Is this consistent with our domain model?

5 References

[Anders-Browser-Profile] A suggestion on how to implement SAML browser bindings without using "Artifacts", http://www.x-obi.com/OBI400/andersr-browser-artifact.ppt

[AuthXML] AuthXML: A Specification for Authentication Information in XML. http://www.oasis-open.org/committees/security/docs/draft-authxml-v2.pdf

[Glossary] OASIS Security Services TC: Glossary. http://www.oasis-open.org/committees/security/docs/draft-sstc-hodges-glossary-02.html

[S2ML] S2ML: Security Services Markup Language, Version 0.8a, January 8, 2001. http://www.oasis-open.org/committees/security/docs/draft-s2ml-v08a.pdf

[Shib] Shiboleth Overview and Requirements http://middleware.internet2.edu/shibboleth/docs/draft-internet2-shibboleth-requirements-00.htmlhttp://middleware.internet2.edu/shibboleth/docs/draft-internet2-shibboleth-requirements-00.html

[Shib-Marlena] Marlena Erdos, Shibboleth Architecture DRAFT v1.1, http://middleware.internet2.edu/shibboleth/docs/draft-erdos-shibboleth-architecturel-00.pdf

[RFC2616] Hypertext Transfer Protocol -- HTTP/1.1

[RFC1750] Randomness Recommendations for Security.

[SOAP1.1] Simple Object Access Protocol (SOAP) 1.1 , W3C Note 08 May 2000

[Core-Assertions-Examples] Core Assertions Architecture, Examples and Explanations, http://www.oasis-open.org/committees/security/docs/draft-sstc-core-phill-07.pdf

[XML-DSIG] XML – Signature Syntax and Processing, available from http://www.w3.org

[WEBSSO] RL "Bob" Morgan, Interactions between Shibboleth and local-site web sign-on services, http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-websso-00.txt

[SESSION] RL "Bob" Morgan, Support of target web server sessions in Shibboleth,

http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-session-00.txt

[rfc1945] Hypertext Transfer Protocol -- HTTP/1.0, http://www.ietf.org/rfc/rfc1945.txt

[rfc2616] Hypertext Transfer Protocol -- HTTP/1.1, http://www.ietf.org/rfc/rfc2616.txt

[rfc2617] HTTP Authentication: Basic and Digest Access Authentication, http://www.ietf.org/rfc/rfc2617.txt

[rfc2774] An HTTP Extension Framework, http://www.ietf.org/rfc/rfc2774.txt

6 Appendix A

http://support.microsoft.com/support/kb/articles/Q208/4/27.ASP

The information in this article applies to:

Microsoft Internet Explorer (Programming) versions 4.0, 4.01, 4.01 SP1, 4.01 SP2, 5, 5.01, 5.5

SUMMARY

Internet Explorer has a maximum uniform resource locator (URL) length of 2,083 characters, with a maximum path length of 2,048 characters. This limit applies to both POST and GET request URLs.

If you are using the GET method, you are limited to a maximum of 2,048 characters (minus the number of characters in the actual path, of course).

POST, however, is not limited by the size of the URL for submitting name/value pairs, because they are transferred in the header and not the URL.

RFC 2616, Hypertext Transfer Protocol -- HTTP/1.1, does not specify any requirement for URL length.

REFERENCES

Further breakdown of the components can be found in the Wininet header file. Hypertext Transfer Protocol -- HTTP/1.1 General Syntax, section 3.2.1

Additional query words: POST GET URL length

Keywords : kbIE kbIE400 kbie401 kbGrpDSInet kbie500 kbDSupport kbie501 kbie550 kbieFAQ

Issue type : kbinfo

Technology :

Issue: 19971110-3 Product: Enterprise Server

Created: 11/10/1997 Version: 2.01

Last Updated: 08/10/1998 OS: AIX, Irix, Solaris

Does this article answer your question?

Please let us know!

Question:

How can I determine the maximum URL length that the Enterprise server will accept? Is this configurable and, if so, how?

Answer:

Any single line in the headers has a limit of 4096 chars; it is not configurable.

---

issue: 19971015-8 Product: Communicator, Netcaster

Created: 10/15/1997 Version: all

Last Updated: 08/10/1998 OS: All

Does this article answer your question?

Please let us know!

Question:

Is there a limit on the length of the URL string?

Answer:

Netscape Communicator and Navigator do not have any limit. Windows 3.1 has a restriction of 32kb (characters). (Note that this is operating system limitation.) See this article for information about Netscape Enterprise Server.

---

7 Appendix B

Javascript may be used to avoid an additional “submit” step from the user. This material is taken from [Anders-Browser-Profile].

```
<HTML>
<BODY Onload="javascript:document.forms[0].submit ()">
```

```
<FORM METHOD="POST" ACTION="Destination-site URL">
…
<INPUT TYPE="HIDDEN" NAME="SAMLAssertion" VALUE="Assertion in Base64-
coding">
</FORM>
</BODY>
</HTML>
```