OASIS SSTC Bindings Model

Prateek Mishra, Netegrity
Marlena Erdos, Tivoli
Chris Ferris, SUN Microsystems
Simon Godik, Crosslogix
Jeff Hodges, Oblix
<big><small>Tim Moses, Entrust
Bob Morgan, University of Washington
Evan Prodromou, Securant
Krishna Sankar, Cisco
</small>

draft-sstc-bindings-model-05.doc

22 August 2001

1 Revision History

Revision	Date	Author	1.1.1.1.1.1 Title
0.5	18 August 2001	Prateek Mishra	Bindings model draft

2 Introduction

2.1 Scope

<big>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</big><big> </big><big>request-response message exchanges. </big>

<big>The high-level goal of this document is to specify how: </big>

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

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.</big><big> </big>

<big>(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, </big><big></big><big>communicated from the originating site to a destination, and subsequently processed at the destination. A set of rules</big><big> </big><big>describing how to embed and extract SAML assertions into a framework or protocol is termed a </big><big>*profile*</big><big> for SAML. A set of rules for embedding and extracting SAML assertions into a </big><big></big><big>specific class of <FOO> objects is termed a </big><big>*<FOO> profile*</big><big> 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.

</big>

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

2.2 Contents

<big>The remainder of this document is in four sections:
</big>

- <big>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.
</big>
- <big>A process framework for describing and registering proposed and future protocol bindings and profiles.
</big>
- <big>Protocol bindings for selected protocols. Bindings MUST be specified in enough detail to satisfy the inter-operability requirement.
</big>
- <big>Profiles for selected protocols and frameworks. Profiles MUST be specified in enough detail to satisfy the inter-operability requirement.
</big>

2.3 Guidelines for Specifying Protocol Bindings and Profiles<big> </big>

<big>Issues that MUST be identified in each protocol binding and profile:</big><big>
</big><big></big><big></big><big>
</big><big>(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.</big><big>
</big><big>
</big><big>(2) Identification of parties involved in each interaction: how many parties are involved in the interaction? Can intermediaries be involved?
</big>

<big>(3) Authentication of parties involved in each interaction: Is authentication required? What types of authentication are acceptable?</big><big>
</big><big>
</big><big>(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? </big><big></big><big>
</big><big>
</big><big>(6) Error states: characterization of error states at each participant, especially those that receive and process SAML assertions or messages.</big>

(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

<big>When a profile or protocol binding is registered, the following information is supplied:</big>

<big> </big>

1. <big>Identification: specify a URI that authoritatively identifies this profile or protocol binding.
</big>
2. <big>Contact information: specify the postal and electronic contact information for the author of the profile or protocol binding.
</big>
3. <big>Description: the description MUST follow the guidelines for profiles and protocol bindings given above.
</big>
4. <big>Updates: references to previously registered profiles or bindings that the current entry improves or obsoletes.</big><big>
</big>

*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.*

<big>Whe</big>

3 Protocol Bindings

3.1 HTTP

3.1.1 Introduction

<big>
</big><big></big><big>The following binding description derives from the HTTP binding provided with [AuthXML]. </big><big></big><big>Note that this section does not treat the issue of passing SAML assertions or assertion tokens from a standard Web browser to a Web server. Instead, it concentrates on using HTTP as a transport layer for SAML messages, without the restrictions that standard Web browsers impose. In most cases, this binding will be used as a service-to-service binding, rather than a user-to-service binding.
</big>

<big> </big><big>Some design goals of this binding are as follows:
</big>

- <big>Enable using existing HTTP software (Web servers, client libraries) to create SAML services.</big>
- <big>Minimize requirements for supporting the somewhat complex HTTP protocol.</big>
- <big>Minimize the information carried in HTTP headers and other data. Except in extreme situations, information should be passed as SAML.</big>

<big> </big><big>Readers of this document should be familiar with HTTP/1.1, which is described in [RFC2616].</big>

3.1.2 Overview

<big>The message protocol for SAML is based on a request-response metaphor. This naturally maps to the HTTP request-response method. So, for most types of interaction between systems, a request message is sent as an HTTP request, and a response message is sent as an HTTP response. There are two parties involved in the interaction: a requester and a responder. There is no provision for intermediaries in the current framework.
</big>

<big>In the discussion that follows, the following terms are used:</big>

<big>* request message -or- request: A SAML request XML object.</big>

<big>* response message -or- response: A SAML response XML object.</big>

<big>* HTTP request: An HTTP request, as distinct from a SAML request.</big>

<big>* HTTP response: An HTTP response, as distinct from a SAML response.</big>

<big>* requester: The party sending the request.</big>

<big>* responder: The party sending the response.</big>

3.1.3 HTTP Binding

3.1.3.1 Connections

<big>As with all HTTP connections, the requester will initiate the connection. Connections MUST be one way. Multiple requests and corresponding responses MAY be sent over a single connection, per the HTTP 1.1 specification. The requester MUST only send requests through the connection, and the responder MUST only send responses through the connection.</big><big> </big><big></big>

3.1.3.2 Request Messages

<big>A request message is bound to an HTTP request.
</big>

<big>The request MUST use the POST method. The HTTP version MUST be one of"1.0" or "1.1".</big>

<big>The request MUST have a Content-Type of "application/saml+xml".

ISSUE:[BINDINGS-02] We will need to register "application/saml+xml" as a new MIME sub-type following RFC3023. Alternatively, can we live with "text/xml" for SAML 1.0?</big>

<big>The content of the HTTP request MUST be exactly one request message. Additional content MUST NOT be included in the HTTP request.</big>

<big>The Host, Date, Content-Type and Content-Length headers MUST be provided in the HTTP request and be correct. A Connection header may be added as is standard in HTTP 1.1.

<big>Additional HTTP headers MAY be provided, but parties in the conversation MUST ignore those other headers.</big>

<big>[Rationale: many existing HTTP libraries will add additional headers to an HTTP request. The intent is to ensure a minimal number of headers required to handle the binding, without requiring that implementations write their own HTTP code.]</big>

<big>Content-Encoding or Transfer-Encoding schemes MUST NOT be used.</big>

<big>[Rationale: SAML messages are relatively small and should not require chunked encoding or compression. Forbidding Content- or Transfer-Encoding will allow implementers to safely ignore these fairly advanced and costly HTTP features.]</big>

3.1.3.3 Response Messages

<big>If a request can be handled and generates a response, the response will be bound to an HTTP response message. If the responder cannot or will not generate a SAML response, the responder MUST send one of the HTTP error responses defined below. The rest of this section will treat only successful responses.
</big>
<big>[Note that success, in this context, means that a SAML response was generated. It does not mean that the request was fulfilled or have domain level meaning, such as that authorization was granted, etc. The SAML response may have failure notifications per the SAML protocol.]</big>

<big>The HTTP response MUST have a status code of 200. The HTTP version MUST be one of "1.0", "1.1".</big>

<big>The response MUST have a Content-Type of "application/saml+xml".</big>

<big>The content of the HTTP response MUST be exactly one response message. Additional content MUST NOT be included in the HTTP response.</big>

<big>The Host, Date, Content-Type and Content-Length headers MUST be provided in the HTTP response and be correct. A Connection header may be added as noted above in section 4.1.</big>

<big>Additional HTTP headers MAY be provided, but parties in the conversation MUST ignore those other headers.</big>

<big>Content-Encoding or Transfer-Encoding schemes MUST NOT be used.</big>

3.1.3.4 Authentication and Message Integrity

Authentication of parties and message integrity of both requests and responses is REQUIRED and may be handled in one of two ways.

3.1.3.4.1 XML Signature

If this technique is used, an XML digital signature MUST be added to the entire request or response.

ISSUE:[BINDINGS-3] *We need a SAML Profile of DSIG to characterize acceptable forms of signing (enveloped, enveloping, detached) and acceptable keyinfo contents.*

3.1.3.4.2 HTTP/S with Certificates

Alternately, the HTTP conversation may be conducted over a Secure Sockets Layer (SSL) connection. In this case, both parties (requester and responder) MUST provide digital certificates for the SSL layer.

3.1.3.5 Message Confidentiality

HTTP/S MAY be used preserve message confidentiality. A server-side certificate is required.

3.1.3.6 Errors

The following error messages may be sent by the responder for a SAML message. Note that in the following section, the error text is not normative, but gives an indication of what the error code means. Only the error number is normative.

For all status values besides "200", the "Connection: close" header MUST be sent, and the connection between requester and responder MUST be closed.

3.1.3.6.1 200 OK

The responder received the request and successfully generated a response. The response may contain a SAML error code or further SAML information. The meaning of the 200 message is "more info in SAML content."

3.1.3.6.2 400 Bad Request

The responder received the request, but the request was ill-formed at the HTTP level in some way. The content of the Response is undefined, but it SHOULD NOT be a SAML message. The content of the Response MAY be a stock piece of HTML or plain text explaining the nature of the error.

[Rationale: Some HTTP server software will add stock explanations for error status codes.]

This result code is appropriate for requests with bad HTTP headers, HTTP methods other than "POST", or an ill-formed HTTP request.

3.1.3.6.3 403 Forbidden

The responder has received the request, but refuses to perform a SAML message exchange with the requestor. The content of the Response is undefined, but it SHOULD NOT be a SAML message. The content of the Response MAY be a stock piece of HTML or plain text explaining the nature of the request.

3.1.3.6.4 500 Internal Server Error

T<big>he responder has received the request but has failed to produce a response, due to internal error. The content of the Response is</big><big> </big><big></big><big>undefined, but it SHOULD NOT be a SAML message. The content of the Response MAY be a stock piece of HTML or plain text explaining the nature of the request.

3.2 SOAP 1.1

3.2.1 Introduction

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 had increased attention, and it is expected to provide the foundation for many future Web-based services.

SOAP 1.1 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 enhanced 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, this section describes using SAML over SOAP, and that section describes using SAML for SOAP.

Like SAML, SOAP can be used over multiple underlying transports. This document does not address the use of underlying transports directly, although it makes recommendations for some transports in addressing message integrity and confidentiality concerns.

3.2.2 Overview

SOAP messages consist of three elements: an envelope, header data, and a message body. SAML messages (queries and responses) are enclosed in the SOAP message body.

SOAP 1.1 also defines an optional data encoding system. This system is not used for the SOAP protocol binding for SAML. This means that SAML messages can be transported using SOAP without re-encoding from "standard" SAML to a SAML-like SOAP encoding.

The system model used for SAML conversations over SOAP is a simple request-response model. A sending party sends a SAML query in the body of a SOAP message. The receiving party processes the SAML query and returns a SAML query response in the body of another SOAP message.

A brief glossary:

SAML conversation: an exchange of a SAML query and a SAML response.

sending party: The party sending a message.

receiving party: The party receiving a message.

querying party: The party sending a query message.

responding party: The party sending a response.

3.2.3 SOAP Binding

3.2.3.1 Namespaces

All SAML messages encoded in SOAP MUST include XML namespace qualifiers, as specified by the core assertions and messages definition.

[Rationale: Some SOAP message processors require a namespace. Also, the namespace prevents conflicts with other standards and schemata.]

3.2.3.2 Headers

The sending party 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.]

The receiving party MAY NOT require any headers for the SOAP message.

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

3.2.3.3 SAML Queries

A SAML query is stored as the child of the <SOAP:body> element of a SOAP message. The querying party MUST send one SAML query. The querying party MUST NOT send more than one SAML query per SOAP message. The querying party MUST NOT include any additional XML elements in the SOAP body.

On receiving a SAML query as a SOAP message, the receiving party MUST return either a SAML query response or a SOAP fault code.

3.2.3.4 SAML Query Responses

A SAML query response is stored as the child of the <SOAP:body> element of a SOAP message. The message MUST contain exactly one SAML query response. The querying party MUST NOT include any additional XML elements in the SOAP body.

On receiving a SAML query response in a SOAP message, the querying party MUST NOT send a fault code or other error messages to the sending party.

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

3.2.3.5 Fault Codes

If a responding party cannot, for some reason, process a SAML query, 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 an object in an authorization query.

The four fault codes (VersionMismatch, MustUnderstand, Client, Server) defined by SOAP 1.1 are sufficient to define any SOAP-related errors. Responding parties MUST NOT use any additional fault codes, or sub-defined fault codes, in a fault response.

Responding parties MAY provide additional fault information, such as descriptions and details, as defined by SOAP.

[Rationale: some SOAP processors may add fault information automatically.]

3.2.3.6 Authentication and Message Integrity

Authentication of parties and message integrity of both requests and responses is REQUIRED and may be handled in one of the following ways.

3.2.3.6.1 XML Digital Signature

To ensure authentication and message integrity, the parties in a SAML conversation MAY add a XML Digital Signature to the SAML query and SAML response.

3.2.3.6.2 HTTP/S with Certificates

Alternately, the parties MAY use the underlying transport of the SOAP conversation to ensure authentication and message integrity. For SOAP messages sent over HTTP, this would be HTTP/S with client and server certificates.

3.2.3.7 Confidentiality

Unfortunately, at the SOAP level itself there is no 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 substrate protocol over which SOAP is layered.

For the case where SOAP messages are used over HTTP, this would be HTTP/S with the use of a server-side certificate.</big>

4 Profiles</big>

4.1 Web Browser

4.1.1 Background

The web browser profile utilizes terminology taken from Use Case 1, Scenario 1-1 and Scenario 1-2 of [OASIS-Use-Case]. This material should be reviewed at this point.

<big>The user is utilizing a standard commercial browser and has authenticated to a *source site*. We assume that the source site has some form of security engine in place that can track locally authenticated users [websso]. 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 strong assumption 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 URL* 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 URL* 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.

ISSUE:[BINDINGS-4] The use-case document and those given above, do not describe the case where the user contacts the destination site first and is then sent to the source site. Do we need to include this step in SAML 1.0 web browser profiles?

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, 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 </big><big></big><big>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 SAML artifact architecture by a mandatory two byte artifact type code in the representation:

```
<SAML_artifact> :=
         B64 representation of <TypeCode> <artifact contents>
         <TypeCode> := Byte1Byte2
```

We describe one specific architecture with 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> := <PartnerID> <AssertionHandle>
<PartnerID> := byte1byte2byte3byte4
```

```
<AssertionHandle> := byte1byte2byte3byte4byte5byte6byte7byte8
```

`<PartnerID>` is a four byte value used by the destination site to determine source site identity as well as the URL (or address) for the “assertion lookup” service. This information needs to have been agreed upon between the source and destination site using an out-of-band technique. On receiving the SAML artifact, the destination site determines if the `<PartnerID>` belongs to a valid partner, accesses the “assertion lookup” service URL and invokes it with the `<AssertionHandle>` value as an argument.

`<AssertionHandle>` is an eight byte value which MUST be drawn from a random number sequence [RFC1750] generated at the source site and serves to identify the assertion to the source site. The `<AssertionHandle>` value is completely opaque to the destination site; further, its construction ensures that it has no predictable relationship to the contents of the referenced assertion at 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. The SAML artifact profile involves two sub-cases: a pull case (corresponds to Scenario 1-1 from [OASIS-Use-Case]) and a push case (corresponds to Scenario 1-2 from [OASIS-Use-Case]). For each type of profile, a section describing the threat model and relevant counter-measures is also included.

Two types of information are 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 has been completed.

(2) information describing one or more SAML assertions. There are two restrictions here. First, each such assertion MUST be a ``bearer'' assertion. Second, one (and only one) assertion MUST take the form of an authentication assertion.

4.1.4.1 SAML Artifact (Pull)

</big>

<big>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 is required, and the inter-site transfer URL MUST be 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 destination URL with target and one or more SAML artifacts carried on the URL query string.

In response, the user browser attempts to access the destination URL (step (3)) and delivers both the destination URL, the SAML artifact(s) and target to (a web server at) the destination site. As this step is over the open internet, confidentiality is required, and the destination URL MUST be 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 communicated assertions and target information, further interaction with the user and other information and 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 SAML artifacts which identify SAML assertions at the source site. If the source site can find the required assertions it responds with a <*samlp:Response*> message with the desired assertion. Otherwise, it returns an "assertion not found" error to destination site.

The selected SAML protocol binding for assertion pull MUST support confidentiality and bi-lateral authentication. The source site MUST implement a SAML HTTP binding with support for confidentiality (HTTPS); support for other protocol bindings is not mandatory.

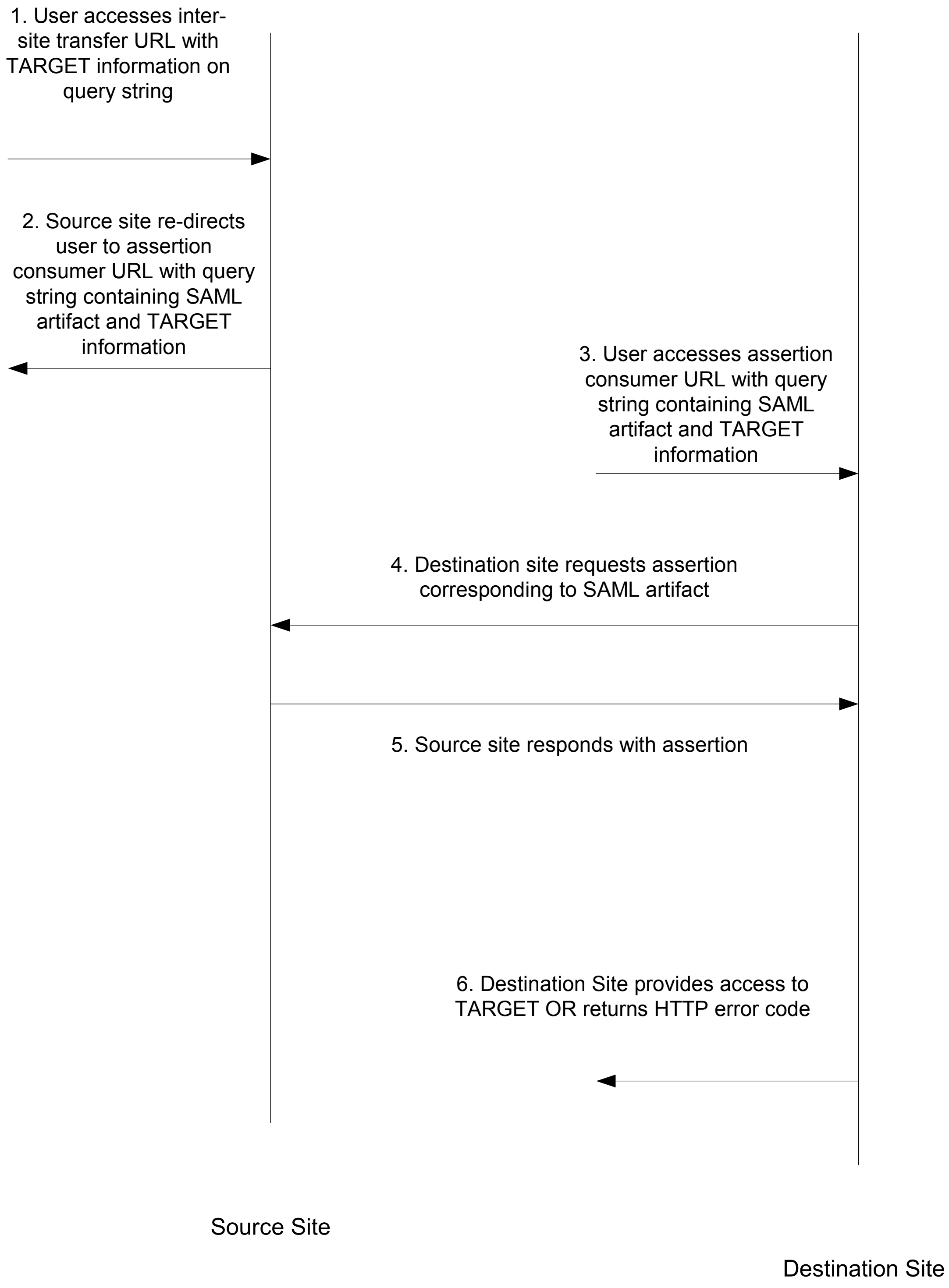

1. User accesses inter-site transfer URL with TARGET information on query string
2. Source site re-directs user to assertion consumer URL with query string containing SAML artifact and TARGET information
3. User accesses assertion consumer URL with query string containing SAML artifact and TARGET information
4. Destination site requests assertion corresponding to SAML artifact
5. Source site responds with assertion
6. Destination Site provides access to TARGET OR returns HTTP error code
Source Site
Destination Site

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

Action	HTTP
(1)	GET https://www.example.com/<inter-site-transfer URL>?TARGET=<target>..
(2)	HTTP 1.1 301 GET https://destination_URL?SAMLart=<artifact body>?TARGET=<target>..
(3)	GET https://destination_URL?SAMLart=<artifact body>?TARGET=<target>..
(4)	<samlp:Request> message is sent to source site with artifact information over selected protocol bindings.
(5)	<samlp:Response> message with an 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 corresponding to an Authentication Assertion 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 destination site MUST implement a "one-time lookup" property for any authentication assertion exposed via a SAML artifact. Many simple implementations meet this requirement: for example deleting the relevant authentication assertion from persistent storage at the source site after first successful 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.

4.1.4.2 SAML Artifact (Push)

Figure 2 describes the interaction sequence for the "push" case of the SAML artifact profile. The number of parties and interactions is very similar to the "pull" case. An authenticated user visits an inter-site transfer URL with information about the destination site target resource as part of the URL query string (step (1)). The source site and destination site participate in a message exchange (steps (2) and (3)) and determine whether the user has the right to access the desired target resource. The source site sends the destination site a `<samlp:Query>` message of type

AuthorizationQueryType and the destination site returns a <samlp:Response> message. The response message carries either SAML artifact(s) or an authorization failure code.

The message exchange utilizes a SAML protocol binding which MUST support bilateral authentication and confidentiality. The destination site MUST implement a SAML HTTP binding with support for confidentiality (HTTPS); support for other protocol bindings is not mandatory.

The destination site (step (4)) either returns an HTTP error-code to the user, or, re-directs the user to the assertion consumer URL at the destination site. Target and SAML artifact information are carried as part of the URL query string. The user browser accesses the assertion consumer URL at the destination site (step(5)) delivering target and artifact information. The destination site “looks up” the assertions corresponding to the delivered artifact and provides access to the desired target (step (6)).

As interactions with the inter-site transfer and assertion consumer URLs is over the open internet, confidentiality is required, and both URLs MUST be exposed over HTTPS (HTTP over server-side SSL).

Action	HTTP
(1)	GET https://www.example.com/<inter-site-transfer URL>?TARGET=<target>..
(2)	<samlp:Request> message of AuthorizationQueryType is sent to destination site utilizing a protocol binding supporting confidentiality.
(3)	<samlp:Response> message with a SAML artifact(s) or an access denied response is returned to destination site utilizing a protocol binding supporting confidentiality.
(4)	HTTP 1.1 301 GET https://assertion_consumer_URL?SAMLart=<artifact body>?TARGET=<target>.. OR “403 Forbidden” returned to browser.
(5)	GET https://assertion_consumer_URL?SAMLart=<artifact body>?TARGET=<target>..
(6)	User is given access to TARGET OR “400 Bad Request” is returned

The destination site MUST implement the following restrictions when processing SAML artifacts:

The SAML artifact corresponding to an authentication assertion MUST be "one-time use"; once the user completes step (6) above, any repitition of step (5) MUST fail with the destination site returning HTTP code “403 Forbidden”.

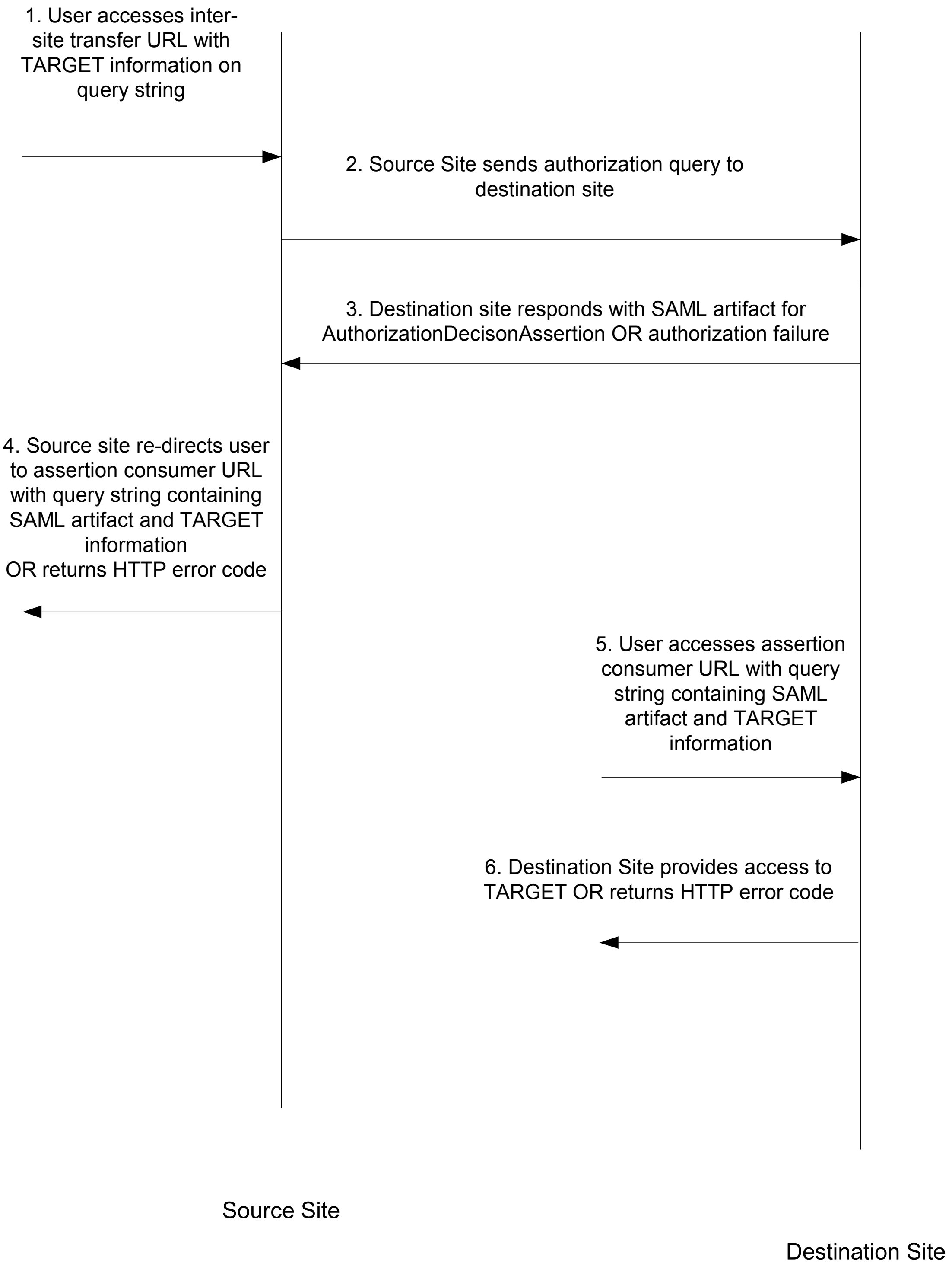

1. User accesses inter-site transfer URL with TARGET information on query string
2. Source Site sends authorization query to destination site
3. Destination site responds with SAML artifact for AuthorizationDecisonAssertion OR authorization failure
4. Source site re-directs user to assertion consumer URL with query string containing SAML artifact and TARGET information
OR returns HTTP error code
5. User accesses assertion consumer URL with query string containing SAML artifact and TARGET information
6. Destination Site provides access to TARGET OR returns HTTP error code
Source Site
Destination Site

Figure 2: web Browser Profile: SAML Artifact (Push)

4.1.4.3 Form POST

Figure 3 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 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 is required, and both URLs MUST 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 authentication assertions 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 life-time. As authentication assertions are recommended to have short life-times in the web browser context, such a table would be of manageable size.

3. Privacy reasons may require that SAML assertions be encrypted. This is an area that requires further investigation.

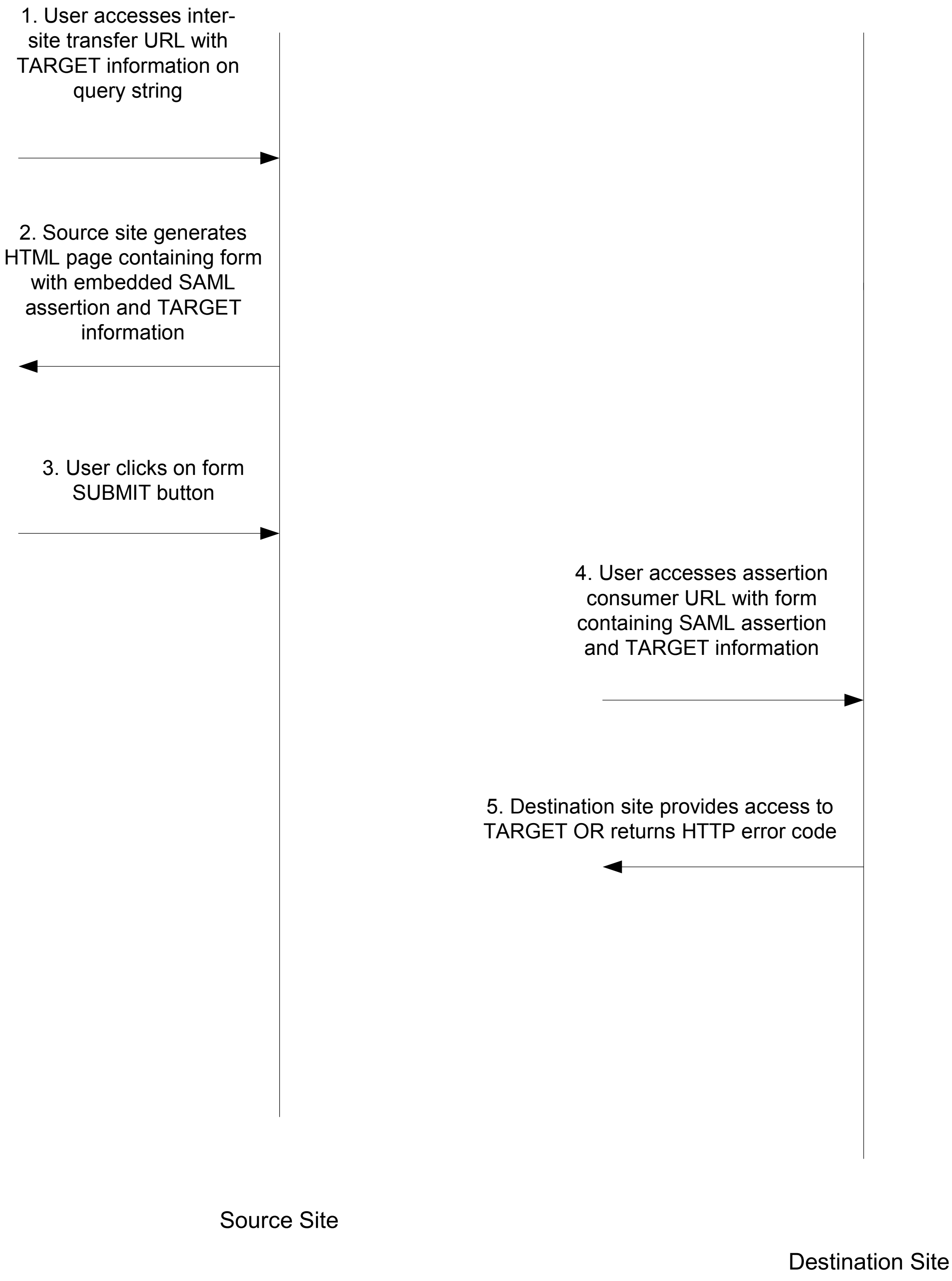

1. User accesses inter-site transfer URL with TARGET information on query string
2. Source site generates HTML page containing form with embedded SAML assertion and TARGET information
3. User clicks on form SUBMIT button
4. User accesses assertion consumer URL with form containing SAML assertion and TARGET information
5. Destination site provides access to TARGET OR returns HTTP error code
Source Site
Destination Site

Figure 3: Web Browser Profile: POST

693

4.1.5 Threat Model and Counter-Measures

This section utilizes materials from [Shib-Marlena].

4.1.5.1 Stolen artifact or assertion

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

   Counter-Measure:
   (a) SAML artifact: SAML artifacts communicated through a web browser profile must always reference a SAML authentication assertion. An authentication assertion communicated through a web browser profile MUST include (1) issue instant and (2) validity period. It MAY include the IP address of the user. It is recommended that a SAML authentication assertion communicated through a web browser profile have the shortest possible validity period consistent with successful functioning of the profile. This is typically of the order of a few minutes.

   The destination site should check the browser IP address against the IP address contained in the authentication assertion (if available) and also ensure that the validity period of the assertion has not expired.

   (b) Form POST: As above.

2. 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 new destination site must obtain the SAML assertions corresponding to the SAML artifacts from the source site through a bilaterally authenticated channel. The SAML artifact profile requires that the source site only allow access to assertions to those sites to which it has directly provided the corresponding SAML artifacts.

   (b) Form POST: A SAML authentication assertion communicated through a form POST is always digitally signed and MUST include the <AudienceRestrictionCondition> element. The destination site must check the <AudienceRestrictionCondition> element to ensure that its value 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 <AudienceRestrictionCondition> value).

4.1.5.2 Forged SAML artifact or Assertion

A MAL could forge a SAML artifact (SAML artifact) or 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.

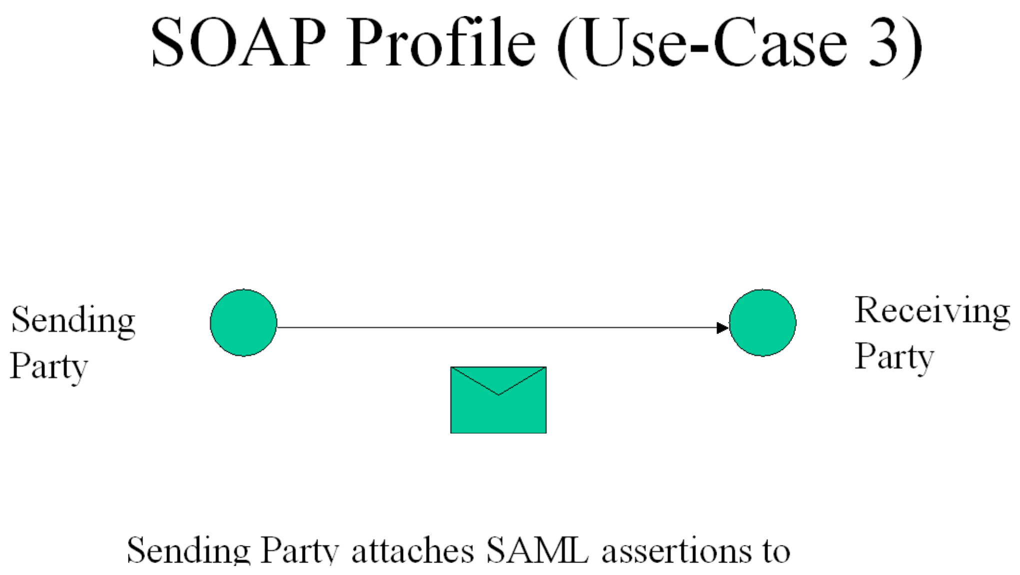
A SAML artifact is a eight byte (sixty-four bit) number drawn from a random sequence. A MAL could attempt to repeatedly “guess” a valid SAML artifact value (one that corresponds to an existing assertion at a source site) but given the size of the value space (2**64 possible values) would likely require a very large number of failed attempts.

4.1.5.3 Browser State Exposure

Both the SAML artifact profile and the POST browser profile involve upload of SAML artifacts or 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 or artifact may be “re-used” at some later point in time.

Counter-Measure: The “one-use” property of SAML artifacts corresponding to authentication assertion presentation and lookup ensures that an authentication assertion artifact may not be re-used from a browser. The form POST case similarly includes a requirement that an authentication assertion cannot be re-presented at the destination site. The web browser profile always requires an authentication assertion.

4.2 SOAP


SOAP Profile (Use-Case 3)

Sending Party attaches SAML assertions to

{PRIVATE "TYPE=PICT;ALT=Figure 2: SOAP Message Transfer"}

4.2.1 Overview

The SOAP profile for SAML is based on a single interaction between a sending party and a receiving party. The sending party adds with one or more SAML assertions to a SOAP document and sends the message to the receiving party. The receiving party processes the SAML assertion and either returns an error or goes 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 [SOAP].

SOAP provides a flexible header mechanism, 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>Subject not authorized</Faultstring>
</Fault>
```

SAML assertions contained with a SOAP message MUST be digitally signed. This ensures that the receiving party can authenticate the issuer and ensure that the assertion hasn't been tampered with.

4.2.3 Confidentiality

In the absence of a mature [XML-Encryption] specification, confidentiality has to be ensured by selection of a "substrate" SOAP protocol binding which preserves confidentiality. This would include, for example, HTTPS or S/MIME. MANDATORY TO IMPLEMENT: HTTPS with server-side certificates.

4.2.4 Example

The following example illustrates the addition of SAML assertions to a SOAP message:

{PRIVATE "TYPE=PICT;ALT=Figure 3: SOAP document with inserted assertions"}

```
<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>
</SOAP-ENV:Header>
…
<SOAP-ENV:Body>
   <message_payload/>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

4.2.5 Integrity of Assertion Attachment

When processing an assertion, the receiving party MUST check the integrity of assertion attachment to the SOAP message. In general, the mandatory `<SubjectConfirmation>` element contained within assertions the may be used to specify this information. Two specific techniques are called as out as mandatory to implement.

4.2.5.1 Digest of SOAP Message

```
<SubjectConfirmation>
  <AuthenticationMethod>DSIG-DIGEST</AuthenticationMethod>
 <SubjectConfirmationData>
                  <dsig:CanonicalizationMethod>…</dsig:CanonicalizationMethod>
                 <dsig:Reference>
                      <dsig:Transforms> . . . </Transforms>
                     <dsig:DigestMethod> . . . </DigestMethod>
                     <dsig:DigestValue>. . .</DigestValue>
                  </Reference>
</SubjectConfirmationData>
</SubjectConfirmation>
```

We plan to re-use elements from [XML-DSIG] to represent the hash of SOAP message within the assertion. The hash value should be computed by EXCLUDING the SAML assertion within which the hash needs to be placed. The `<Transforms>` element is provided for this purpose. We need to (1) ensure that there is DSIG-DIGEST authentication method, and (2) profile the use of the required elements from [XML-DSIG].

4.2.5.2 Digital Signature

Using this technique, the <SubjectConfirmation> element carries the sender's public key or X509 certificate within the `<dsig:KeyInfo>` element. The signature itself is carried separately as a `<dsig:signature>` element as part of the SOAP envelope. This ensures that an assertion can re-used with many different SOAP messages. Note that the signature element is used only for checking integrity of assertion attachment (message integrity). Therefore, there is no requirement that the receiving party validate the key or certificate. This also suggests that servers can generate public/private key pairs and utilize them for this purpose.

```
<SubjectConfirmation>
  <AuthenticationMethod>DSIG-SIGNATURE</AuthenticationMethod>
  <dsig:KeyInfo>…<dsig:KeyInfo>
 </SubjectConfirmation>
```

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

[draft-sstc-core-07.doc] Security Assertions Markup Language, Version 0.7, May 14th, 2001. http://www.oasis-open.org/committees/security/docs/draft-sstc-core-07.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.

[SOAP] 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

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.

<map></map>

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>
```