Internet Engineering Task Force (IETF)
Request for Comments: 6595
Category: Standards Track
ISSN: 2070-1721

K. Wierenga
Cisco Systems, Inc.
E. Lear
Cisco Systems GmbH
S. Josefsson
SJD AB

April 2012

# A Simple Authentication and Security Layer (SASL) and GSS-API Mechanism for the Security Assertion Markup Language (SAML)

## Abstract

The Security Assertion Markup Language (SAML) has found its usage on the Internet for Web Single Sign-On. The Simple Authentication and Security Layer (SASL) and the Generic Security Service Application Program Interface (GSS-API) are application frameworks to generalize authentication. This memo specifies a SASL mechanism and a GSS-API mechanism for SAML 2.0 that allows the integration of existing SAML Identity Providers with applications using SASL and GSS-API.

## Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6595.

Table of Contents

## 1. Introduction

Security Assertion Markup Language (SAML) 2.0 [OASIS-SAMLv2-CORE] is a set of specifications that provide various means for a user to be identified to a Relying Party (RP) through the exchange of (typically signed) assertions issued by an Identity Provider (IdP). It includes a number of protocols, protocol bindings [OASIS-SAMLv2-BIND], and interoperability profiles [OASIS-SAMLv2-PROF] designed for different use cases.

The Simple Authentication and Security Layer (SASL) [RFC4422] is a generalized mechanism for identifying and authenticating a user and for optionally negotiating a security layer for subsequent protocol interactions. SASL is used by application protocols like IMAP [RFC3501], the Post Office Protocol (POP) [RFC1939], and the Extensible Message and Presence Protocol (XMPP) [RFC6120]. The effect is to make modular authentication, so that newer authentication mechanisms can be added as needed. This memo specifies just such a mechanism.

The Generic Security Service Application Program Interface (GSS-API) [RFC2743] provides a framework for applications to support multiple authentication mechanisms through a unified programming interface. This document defines a pure SASL mechanism for SAML, but it conforms to the new bridge between SASL and the GSS-API called GS2 [RFC5801]. This means that this document defines both a SASL mechanism and a GSS-API mechanism. The GSS-API interface is OPTIONAL for SASL implementers, and the GSS-API considerations can be avoided in environments that use SASL directly without GSS-API.

As currently envisioned, this mechanism enables interworking between SASL and SAML in order to assert the identity of the user and other attributes to RPs. As such, while servers (as RPs) will advertise SASL mechanisms (including SAML), clients will select the SAML SASL mechanism as their SASL mechanism of choice.

The SAML mechanism described in this memo aims to reuse the Web Browser Single Sign-On (SSO) profile defined in Section 4.1 of the SAML 2.0 profiles specification [OASIS-SAMLv2-PROF] to the maximum extent and therefore does not establish a separate authentication, integrity, and confidentiality mechanism. The mechanism assumes that a security layer, such as Transport Layer Security (TLS) [RFC5246], will continue to be used. This specification is appropriate for use when a browser instance is available. In the absence of a browser instance, SAML profiles that don’t require a browser, such as the Enhanced Client or Proxy profile (as defined in Section 4.2 of [OASIS-SAMLv2-PROF], may be used, but that is outside the scope of this specification.

Figure 1 describes the interworking between SAML and SASL: this document requires enhancements to the RP (the SASL server) and to the client, as the two SASL communication end points, but no changes to the SAML IdP are necessary. To accomplish this goal, some indirect messaging is tunneled within SASL, and some use of external methods is made.

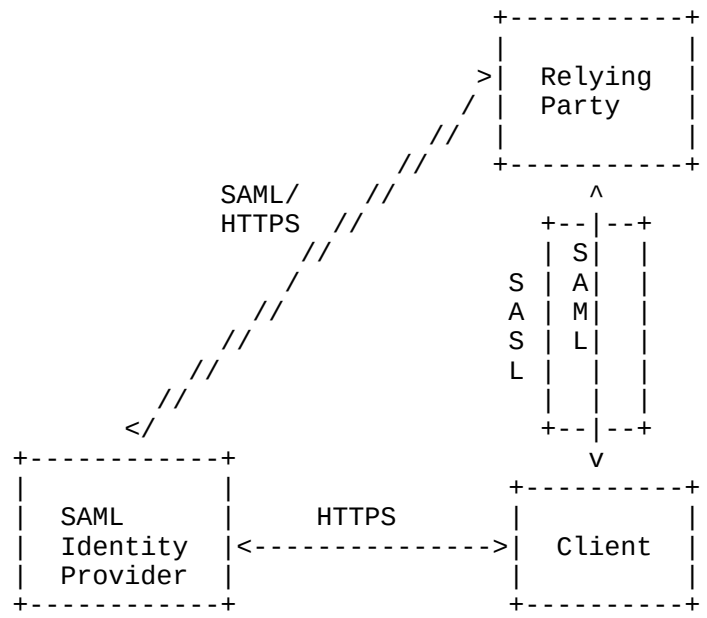


Figure 1: Interworking Architecture

## 1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

The reader is assumed to be familiar with the terms used in the SAML 2.0 core specification [OASIS-SAMLv2-CORE].

## 1.2. Applicability

Because this mechanism transports information that should not be controlled by an attacker, the SAML mechanism MUST only be used over channels protected by TLS, or over similar integrity-protected and authenticated channels. In addition, when TLS is used, the client MUST successfully validate the server's certificate ([RFC5280], [RFC6125]).

Note: An Intranet does not constitute such an integrity-protected and authenticated channel!

## 2. Authentication Flow

While SAML itself is merely a markup language, its common use case these days is with HTTP [RFC2616] or HTTPS [RFC2818] and HTML [W3C-REC-HTML401]. What follows is a typical flow:

1. The browser requests a resource of an RP (via an HTTP request).

2. The RP redirects the browser via an HTTP redirect (as described in Section 10.3 of [RFC2616]) to the IdP or an IdP discovery service. When it does so, it includes the following parameters: (1) an authentication request that contains the name of the resource being requested, (2) a browser cookie, and (3) a return URL as specified in Section 3.1 of [OASIS-SAMLv2-PROF].

3. The user authenticates to the IdP and perhaps authorizes the release of user attributes to the RP.

4. In its authentication response, the IdP redirects (via an HTTP redirect) the browser back to the RP with an authentication assertion (stating that the IdP vouches that the subject has successfully authenticated), optionally along with some additional attributes.

5. The RP now has sufficient identity information to approve access to the resource or not, and acts accordingly. The authentication is concluded.

When considering this flow in the context of SASL, we note that while the RP and the client both must change their code to implement this SASL mechanism, the IdP can remain untouched. The RP already has some sort of session (probably a TCP connection) established with the client. However, it may be necessary to redirect a SASL client to another application or handler. The steps are as follows:

1. The SASL server (RP) advertises support for the SASL SAML20 mechanism to the client.

2. The client initiates a SASL authentication with SAML20 and sends a domain name that allows the SASL server to determine the appropriate IdP.

3. The SASL server transmits an authentication request encoded using a Uniform Resource Identifier (URI) as described in RFC 3986 [RFC3986] and an HTTP redirect to the IdP corresponding to the domain.

4. The SASL client now sends a response consisting of "=". Authentication continues via the normal SAML flow, and the SASL server will receive the answer to the challenge out of band from the SASL conversation.

5. At this point, the SASL client MUST construct a URL containing the content received in the previous message from the SASL server. This URL is transmitted to the IdP either by the SASL client application or an appropriate handler, such as a browser.

6. Next, the user authenticates to the IdP. The manner in which the end user is authenticated to the IdP, and any policies surrounding such authentication, are out of scope for SAML and hence for this document. This step happens out of band from SASL.

7. The IdP will convey information about the success or failure of the authentication back to the SASL server (RP) in the form of an authentication statement or failure, using an indirect response via the client browser or the handler (and with an external browser, client control should be passed back to the SASL client). This step happens out of band from SASL.

8. The SASL server sends an appropriate SASL response to the client.

Please note: What is described here is the case in which the client has not previously authenticated. It is possible that the client already holds a valid SAML authentication token so that the user does not need to be involved in the process anymore, but that would still be external to SASL. This is classic Web Single Sign-On, in which the Web Browser client presents the authentication token (cookie) to the RP without renewed user authentication at the IdP.

With all of this in mind, the flow appears as follows in Figure 2:

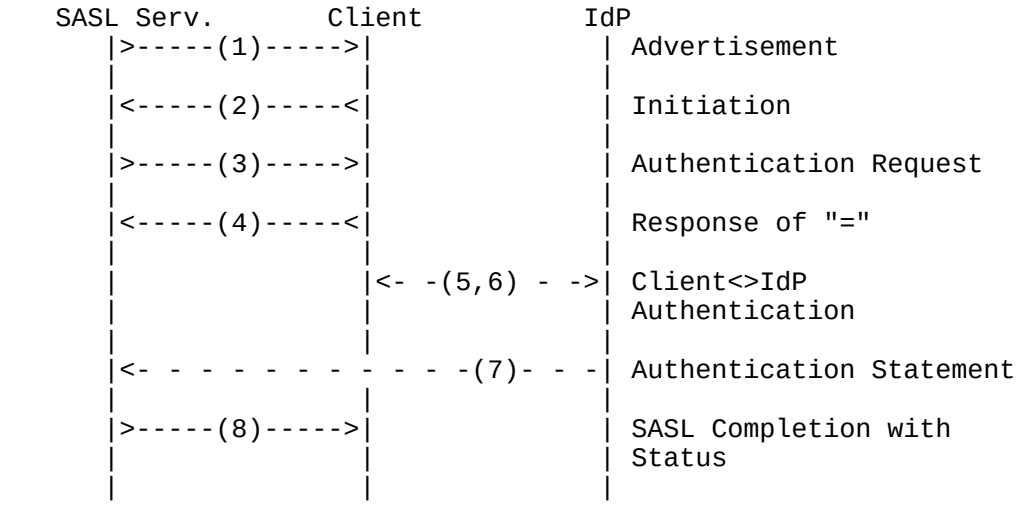


Figure 2: Authentication Flow

## 3. SAML SASL Mechanism Specification

This section specifies the details of the SAML SASL mechanism. See Section 5 of [RFC4422] for additional details.

The name of this mechanism is "SAML20". The mechanism is capable of transferring an authorization identity (via the "gs2-header"). The mechanism does not offer a security layer.

The mechanism is client-first. The first mechanism message from the client to the server is the "initial-response". As described in [RFC4422], if the application protocol does not support sending a client response together with the authentication request, the server will send an empty server challenge to let the client begin. The second mechanism message is from the server to the client, containing the SAML "authentication-request". The third mechanism message is from the client to the server and is the fixed message consisting of "=". The fourth mechanism message is from the server to the client, indicating the SASL mechanism outcome.

### 3.1. Initial Response

A client initiates a SAML20 authentication with SASL by sending the GS2 header followed by the Identity Provider identifier (message 2 in Figure 2) and is defined using ABNF [RFC5234] as follows:

```
initial-response = gs2-header IdP-Identifier
IdP-Identifier = domain ; domain name with corresponding IdP
```

The gs2-header is used as follows:

- The "gs2-nonstd-flag" MUST NOT be present.

- The "gs2-cb-flag" MUST be set to "n" because channel-binding [RFC5056] data cannot be integrity protected by the SAML negotiation. (Note: In theory, channel-binding data could be inserted in the SAML flow by the client and verified by the server, but that is currently not supported in SAML.)

- The "gs2-authzid" carries the optional authorization identity as specified in [RFC5801] (not to be confused with the IdP-Identifier).

A domain name is either a "traditional domain name" as described in [RFC1035] or an "internationalized domain name" as described in [RFC5890]. Clients and servers MUST treat the IdP-Identifier as a domain name slot [RFC5890]. They also SHOULD support internationalized domain names (IDNs) in the IdP-Identifier field; if they do so, all of the domain name’s labels MUST be A-labels or NR-LDH labels [RFC5890]. If necessary, internationalized labels MUST be converted from U-labels to A-labels by using the Punycode encoding [RFC3492] for A-labels prior to sending them to the SASL server, as described in the protocol specification for Internationalized Domain Names in Applications [RFC5891].

### 3.2. Authentication Request

The SASL server transmits to the SASL client a URI that redirects the SAML client to the IdP (corresponding to the domain that the user provided), with a SAML authentication request as one of the parameters (message 3 in Figure 2) using the following ABNF:

```
authentication-request = URI
```

The URI is specified in [RFC3986] and is encoded according to Section 3.4 ("HTTP Redirect Binding") of the SAML 2.0 bindings specification [OASIS-SAMLv2-BIND]. The SAML authentication request is encoded according to Section 3.4 ("Authentication Request

Protocol") of [OASIS-SAMLv2-CORE]. Should the client support Internationalized Resource Identifiers (IRIs) [RFC3987], it MUST first map the IRI to a URI before transmitting it to the server, as defined in Section 3.1 of [RFC3987].

Note: The SASL server may have a static mapping of domain to corresponding IdP or, alternatively, a DNS-lookup mechanism could be envisioned, but that is out of scope for this document.

Note: While the SASL client MAY sanity-check the URI it received, ultimately it is the SAML IdP that will be validated by the SAML client; this topic is out of scope for this document.

The client then sends the authentication request via an HTTP GET (sent over a server-authenticated TLS channel) to the IdP, as if redirected to do so from an HTTP server and in accordance with the Web Browser SSO profile, as described in Section 4.1 of [OASIS-SAMLv2-PROF] (messages 5 and 6 in Figure 2).

The client handles both user authentication to the IdP and confirmation or rejection of the authentication of the RP (out of scope for this document).

After all authentication has been completed by the IdP, the IdP will send a redirect message to the client in the form of a URI corresponding to the RP as specified in the authentication request ("AssertionConsumerServiceURL") and with the SAML response as one of the parameters (message 7 in Figure 2).

Please note: This means that the SASL server needs to implement a SAML RP. Also, the SASL server needs to correlate the session it has with the SASL client with the appropriate SAML authentication result. It can do so by comparing the ID of the SAML authentication request it has issued with the one it receives in the SAML authentication statement.

### 3.3. Outcome and Parameters

The SASL server (in its capacity as a SAML RP) now validates the SAML authentication response it received from the SAML client via HTTP or HTTPS.

The outcome of that validation by the SASL server constitutes a SASL mechanism outcome and therefore (as stated in [RFC4422]) SHALL be used to set state in the server accordingly, and it SHALL be used by the server to report that state to the SASL client, as described in [RFC4422], Section 3.6 (message 8 in Figure 2).

## 4. SAML GSS-API Mechanism Specification

This section and its sub-sections are not required for SASL implementors, but this section MUST be observed to implement the GSS-API mechanism discussed below.

This section specifies a GSS-API mechanism that, when used via the GS2 bridge to SASL, behaves like the SASL mechanism defined in this document. Thus, it can loosely be said that the SAML SASL mechanism is also a GSS-API mechanism. The SAML user takes the role of the GSS-API Initiator, and the SAML RP takes the role of the GSS-API Acceptor. The SAML IdP does not have a role in GSS-API and is considered an internal matter for the SAML mechanism. The messages are the same, but

a) the GS2 header on the client's first message and channel-binding data are excluded when SAML is used as a GSS-API mechanism, and

b) the initial context token header (Section 3.1 of [RFC2743]) is prefixed to the client's first authentication message (context token).

The GSS-API mechanism OID for SAML is 1.3.6.1.5.5.17 (see Section 7.2 for more information). The DER encoding of the OID is 0x2b 0x06 0x01 0x05 0x05 0x11.

SAML20 security contexts MUST have the mutual_state flag (GSS_C_MUTUAL_FLAG) set to TRUE. SAML does not support credential delegation; therefore, SAML security contexts MUST have the deleg_state flag (GSS_C_DELEG_FLAG) set to FALSE.

The mutual authentication property of this mechanism relies on successfully comparing the TLS server's identity with the negotiated target name. Since the TLS channel is managed by the application outside of the GSS-API mechanism, the mechanism itself is unable to confirm the name, while the application is able to perform this comparison for the mechanism. For this reason, applications MUST match the TLS server's identity with the target name, as discussed in [RFC6125]. More precisely, to pass identity validation, the client uses the securely negotiated targ_name as the reference identifier and matches it to the DNS-ID of the server's certificate, and it MUST reject the connection if there is a mismatch. For compatibility with deployed certificate hierarchies, the client MAY also perform a comparison with the Common Name ID (CN-ID) when there is no DNS-ID present. Wildcard matching is permitted. The targ_name reference identifier is a "traditional domain names"; thus, the comparison is made using case-insensitive ASCII comparison.

The SAML mechanism does not support per-message tokens or the GSS_Pseudo_random() function [RFC4401].

### 4.1. GSS-API Principal Name Types for SAML

SAML supports standard generic name syntaxes for acceptors such as GSS_C_NT_HOSTBASED_SERVICE (see [RFC2743], Section 4.1). SAML supports only a single name type for initiators: GSS_C_NT_USER_NAME. GSS_C_NT_USER_NAME is the default name type for SAML. The query, display, and exported name syntaxes for SAML principal names are all the same. There are no SAML-specific name syntaxes -- applications should use generic GSS-API name types, such as GSS_C_NT_USER_NAME and GSS_C_NT_HOSTBASED_SERVICE (see [RFC2743] Section 4). The exported name token, of course, conforms to [RFC2743], Section 3.2.

## 5. Examples

### 5.1. XMPP

Suppose the user has an identity at the SAML IdP saml.example.org and a Jabber Identifier (JID) "somenode@example.com" and wishes to authenticate his XMPP [RFC6120] connection to xmpp.example.com. The authentication on the wire would then look something like the following:

Step 1: Client initiates stream to server:

```
<stream:stream xmlns='jabber:client'
xmlns:stream='http://etherx.jabber.org/streams'
to='example.com' version='1.0'>
```

Step 2: Server responds with a stream tag sent to client:

```
<stream:stream
xmlns='jabber:client' xmlns:stream='http://etherx.jabber.org/streams'
id='some_id' from='example.com' version='1.0'>
```

Step 3: Server informs client of available authentication mechanisms:

```
<stream:features>
 <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
  <mechanism>DIGEST-MD5</mechanism>
  <mechanism>PLAIN</mechanism>
  <mechanism>SAML20</mechanism>
 </mechanisms>
</stream:features>
```

Step 4: Client selects an authentication mechanism and provides the initial client response -- containing the gs2-header and domain -- that has been encoded in base64 according to Section 4 of [RFC4648]:

```
<auth xmlns='urn:ietf:params:xml:ns:xmpp-sasl' mechanism='SAML20'>
biwsZXhhbXBsZS5vcmc=</auth>
```

The decoded string is

```
n,,example.org
```

Step 5: Server sends a base64-encoded challenge to client in the form of an HTTP redirect to the SAML IdP corresponding to example.org (https://saml.example.org) with the SAML authentication request as specified in the redirection URL:

```
aHR0cHM6Ly9zYW1sLmV4YW1wbGUub3JnL1NBTUwvQnJvd3Nlcj9TQU1MUmVx
dWVzdD1QSE5oYld4d09rRjFkR2h1VW1WeGRXVnpkQ0I0Yld4dWN6cHpZVzFz
Y0QwaWRYSnVPbTloYzJsek9tNWhiV1Z6T25Sak9sTkJUVXc2TWk0d09uQnli
M1J2WTI5c0lnMEtJQ0FnSUVsRVBTSmZZbVZqTkRJMFptRTFNVEF6TkRJNE9U
QTVZVE13Wm1ZeFpUTXhNVFk0TXpJM1pqYzVORGMwT1RnMElpQldaWEp6YVc5
dVBTSXlMakFpRFFvZ0lDQWdTWE56ZFdWSmJuTjBZVzUwUFNJeU1EQTNMVEV5
TFRFd1ZERXhPak01T2pNMFdpSWdSbTl5WTJWQmRYUm9iajBpWm1Gc2MyVWlE
UW9nSUNBZ1NYTlFZWE56YVhabFBTSm1ZV3h6WlNJTkNpQWdJQ0JRY205MGIy
TnZiRUpwYm1ScGJtYzlJblZ5YmpwdllYTnBjenB1WVcxbGN6cDBZenBUUVUx
TU9qSXVNRHBpYVc1a2FXNW5jenBJVkZSUUxWQlBVMVFpRFFvZ0lDQWdRWE56
WlhKMGFXOXVRMjl1YzNWdFpYSlRaWEoyYVdObFZWSk1QUTBLSUNBZ0lDQWdJ
Q0FpYUhSMGNITTZMeTk0YlhCd0xtVjRZVzF3YkdVdVkyOXRMMU5CVFV3dlFY
TnpaWEowYVc5dVEyOXVjM1Z0WlhKVFpYSjJhV05sSWo0TkNpQThjMkZ0YkRw
SmMzTjFaWElnZUcxc2JuTTZjMkZ0YkQwaWRYSnVPbTloYzJsek9tNWhiV1Z6
T25Sak9sTkJUVXc2TWk0d09tRnpjMlZ5ZEdsdmJpSStEUW9nSUNBZ0lHaDBk
SEJ6T2k4dmVHMXdjQzVsZUdGdGNHeGxMbU52YlEwS0lEd3ZjMkZ0YkRwSmMz
TjFaWEkrRFFvZ1BITmhiV3h3T2s1aGJXVkpSRkJ2YkdsamVTQjRiV3h1Y3pw
ellXMXNjRDBpZFhKdU9tOWhjMmx6T201aGJXVnpPblJqT2xOQlRVdzZNaTR3
T25CeWIzUnZZMjlzSWcwS0lDQWdJQ0JHYjNKdFlYUTlJblZ5YmpwdllYTnBj
enB1WVcxbGN6cDBZenBUUVUxTU9qSXVNRHB1WVcxbGFXUXRabTl5YldGME9u
Qmxjbk5wYzNSbGJuUWlEUW9nSUNBZ0lGTlFUbUZ0WlZGMVlXeHBabWxsY2ow
aWVHMXdjQzVsZUdGdGNHeGxMbU52YlNJZ1FXeHNiM2REY21WaGRHVTlJblJ5
ZFdVaUlDOCtEUW9nUEhOaGJXeHdPbEpsY1hWbGMzUmxaRUYxZEdodVEyOXVk
R1Y0ZEEwS0lDQWdJQ0I0Yld4dWN6cHpZVzFzY0QwaWRYSnVPbTloYzJsek9t
NWhiV1Z6T25Sak9sTkJUVXc2TWk0d09uQnliM1J2WTI5c0lpQU5DaUFnSUNB
Z0lDQWdRMjl0Y0dGeWFYTnZiajBpWlhoaFkzUWlQZzBLSUNBOGMyRnRiRHBC
ZFhSb2JrTnZiblJsZUhSRGJHRnpjMUpsWmcwS0lDQWdJQ0FnZUcxc2JuTTZj
MkZ0YkQwaWRYSnVPbTloYzJsek9tNWhiV1Z6T25Sak9sTkJUVXc2TWk0d09t
RnpjMlZ5ZEdsdmJpSStEUW9nb0NBZ0lDQjFjbTQ2YjJGemFYTTZibUZ0WlhN
NmRHTTZVMEZOVERveUxqQTZZV002WTJ4aGMzTmxjenBRWVhOemQyOXlaRkJ5
YjNSbFkzUmxaRlJ5WVc1emNHOXlkQTBLSUNBOEwzTmhiV3c2UVhWMGFHNURi
```

```
    MjUwWlhoMFEyeGhjM05TWldZK0RRb2dQQzl6WVcxc2NEcFNaWEYxWlhOMFpX
    UkJkWFJvYmtOdmJuUmxlSFErSUEwS1BDOXpZVzFzY0RwQmRYUm9ibEpsY1hW
    bGMzUSs=
```

The decoded challenge is as follows:

```
    https://saml.example.org/SAML/Browser?SAMLRequest=PHNhbWxwOk
    F1dGhuUmVxdWVzdCB4bWxuczpzYW1scD0idXJuOm9hc2lzOm5hbWVzOnRjOl
    NBTUw6Mi4wOnByb3RvY29sIg0KICAgIElEPSJfYmVjNDI0ZmE1MTAzNDI4OT
    A5YTMwZmYxZTMxMTY4MzI3Zjc5NDc0OTg0IiBWZXJzaW9uPSIyLjAiDQogIC
    AgSXNzdWVJbnN0YW50PSIyMDA3LTEyLTEwVDExOjM5OjM0WiIgRm9yY2VBdX
    Robj0iZmFsc2UiDQogICAgSXNQYXNzaXZlPSJmYWxzZSINCiAgICBQcm90b2
    NvbEJpbmRpbmc9InVybjpvYXNpczpuYW1lczp0YzpTQU1MOjIuMDpiaW5kaW
    5nczpIVFRQLVBPU1QiDQogICAgQXNzZXJ0aW9uQ29uc3VtZXJTZXJ2aWNlVV
    JMPQ0KICAgICAgICAiaHR0cHM6Ly94bXBwLmV4YW1wbGUuY29tL1NBTUwvQX
    NzZXJ0aW9uQ29uc3VtZXJTZXJ2aWNlIj4NCiA8c2FtbDpJc3N1ZXIgeG1sbn
    M6c2FtbD0idXJuOm9hc2lzOm5hbWVzOnRjOlNBTUw6Mi4wOmFzc2VydGlvbi
    I+DQogICAgIGh0dHBzOi8veG1wcC5leGFtcGxlLmNvbQ0KIDwvc2FtbDpJc3
    N1ZXI+DQogPHNhbWxwOk5hbWVJRFBvbGljeSB4bWxuczpzYW1scD0idXJuOm
    9hc2lzOm5hbWVzOnRjOlNBTUw6Mi4wOnByb3RvY29sIg0KICAgICBGb3JtYX
    Q9InVybjpvYXNpczpuYW1lczp0YzpTQU1MOjIuMDpuYW1laWQtZm9ybWF0On
    BlcnNpc3RlbnQiDQogICAgIFNQTmFtZVF1YWxpZmllcj0ieG1wcC5leGFtcG
    xlLmNvbSIgQWxsb3dDcmVhdGU9InRydWUiIC8+DQogPHNhbWxwOlJlcXVlc3
    RlZEF1dGhuQ29udGV4dA0KICAgICB4bWxuczpzYW1scD0idXJuOm9hc2lzOm
    5hbWVzOnRjOlNBTUw6Mi4wOnByb3RvY29sIiANCiAgICAgQ29tcGFyaX
    Nvbj0iZXhhY3QiPg0KICA8c2FtbDpBdXRobkNvbnRleHRDbGFzc1JlZg0KIC
    AgICAgeG1sbnM6c2FtbD0idXJuOm9hc2lzOm5hbWVzOnRjOlNBTUw6Mi4wOm
    Fzc2VydGlvbiI+DQogICAgICAgICAgIHVybjpvYXNpczpuYW1lczp0YzpTQU
    1MOjIuMDphYzpjbGFzc2VzOlBhc3N3b3JkUHJvdGVjdGVkVHJhbnNwb3J0DQ
    ogIDwvc2FtbDpBdXRobkNvbnRleHRDbGFzc1JlZj4NCiA8L3NhbWxwOlJlcX
    Vlc3RlZEF1dGhuQ29udGV4dD4gDQo8L3NhbWxwOkF1dGhuUmVxdWVzdD4=
```

Where the decoded SAMLRequest looks like the following:

```
 <samlp:AuthnRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
     ID="_bec424fa5103428909a30ff1e31168327f79474984" Version="2.0"
     IssueInstant="2007-12-10T11:39:34Z" ForceAuthn="false"
     IsPassive="false"
     ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
     AssertionConsumerServiceURL=
         "https://xmpp.example.com/SAML/AssertionConsumerService">
  <saml:Issuer xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
      https://xmpp.example.com
  </saml:Issuer>
  <samlp:NameIDPolicy xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
      Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent"
      SPNameQualifier="xmpp.example.com" AllowCreate="true" />
  <samlp:RequestedAuthnContext
```

```
      xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
         Comparison="exact">
   <saml:AuthnContextClassRef
       xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
       urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
   </saml:AuthnContextClassRef>
  </samlp:RequestedAuthnContext>
 </samlp:AuthnRequest>
```

Note: The server can use the request ID ("_bec424fa5103428909a30ff1e31168327f79474984") to correlate the SASL session with the SAML authentication.

Step 5 (alternative): Server returns error to client if no SAML authentication request can be constructed:

```
 <failure xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
  <temporary-auth-failure/>
 </failure>
 </stream:stream>
```

Step 6: Client sends the "=" response (base64-encoded) to the challenge:

```
 <response xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
  PQ==
 </response>
```

The following steps between brackets are out of scope for this document but are included to better illustrate the entire flow:

[The client now sends the URL to a browser instance for processing. The browser engages in a normal SAML authentication flow (external to SASL), like redirection to the IdP (https://saml.example.org); the user logs into https://saml.example.org and agrees to authenticate to xmpp.example.com. A redirect is passed back to the client browser. The client browser in turn sends the AuthN response, which contains the subject-identifier as an attribute, to the server. If the AuthN response doesn't contain the JID, the server maps the subject-identifier received from the IdP to a JID.]

Step 7: Server informs client of successful authentication:

```
<success xmlns='urn:ietf:params:xml:ns:xmpp-sasl'/>
```

Step 7 (alternative): Server informs client of failed authentication:

```
<failure xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
 <not-authorized/>
</failure>
</stream:stream>
```

Please note: Line breaks were added to the base64 data for clarity.

## 5.2. IMAP

The following sequence describes an IMAP exchange. Lines beginning with 'S:' indicate data sent by the server, and lines starting with 'C:' indicate data sent by the client. Long lines are wrapped for readability.

```
S: * OK IMAP4rev1
C: . CAPABILITY
S: * CAPABILITY IMAP4rev1 STARTTLS
S: . OK CAPABILITY Completed
C: . STARTTLS
S: . OK Begin TLS negotiation now
C: . CAPABILITY
S: * CAPABILITY IMAP4rev1 AUTH=SAML20
S: . OK CAPABILITY Completed
C: . AUTHENTICATE SAML20
S: +
C: biwsZXhhbXBsZS5vcmc=
S: + aHR0cHM6Ly9zYW1sLmV4YW1wbGUub3JnL1NBTUwvQnJvd3Nlcj9TQU1M
UmVxdWVzdD1QSE5oYld4d09rRg0KMWRHaHVVbVZ4ZFdWemRDQjRiV3h1Y3pwe
llXMXNjRDBpZFhKdU9tOWhjMmx6T201aGJXVnpPblJqT2xOQg0KVFV3Nk1pNH
dPbkJ5YjNSdlkyOXNJZzBLSUNBZ0lFbEVQU0pmWW1Wak5ESTBabUUxTVRBek5
ESTRPVEE1WQ0KVE13Wm1ZeFpUTXhNVFk0TXpJM1pqYzVORGMwT1RnMElpQlda
WEp6YVc5dVBTSXlMakFpRFFvZ0lDQWdTWA0KTnpkV1ZKYm5OMFlXNTBQU0l5T
URBM0xURXlMVEV3VkRFeE9qTTVPak0wV2lJZ1JtOXlZMlZCZFhSb2JqMA0KaV
ptRnNjMlVpRFFvZ0lDQWdTWE5RWVhOemFYWmxQU0ptWVd4elpTSU5DaUFnSUN
CUWNtOTBiMk52YkVKcA0KYm1ScGJtYzlZblZ5YmpwdllYTnBjenB1WVcxbGN6
cDBZenBUUVUxTU9qSXVNRHBpYVc1a2FXNW5jenBJVg0KRlJRTFZCUFUxUWlEU
W9nSUNBZ1FYTnpaWEowYVc5dVEyOXVjM1Z0WlhKVFpYSjJhV05sVlZKTVBRME
tJQw0KQWdJQ0FnSUNBaWFIUjBjSE02THk5dFlXbHNMbVY0WVcxd2JHVXVZMjl
0TDFOQlRVd3ZRWE56WlhKMGFXOQ0KdVEyOXVjM1Z0WlhKVFpYSjJhV05sSWo0
TkNpQThjMkZ0YkRwSmMzTjFaWElnZUcxc2JuTTZjMkZ0YkQwaQ0KZFhKdU9tO
WhjMmx6T201aGJXVnpPblJqT2xOQlRVdzZNaTR3T21GemMyVnlkR2x2YmlJK0
RRb2dJQ0FnSQ0KR2gwZEhCek9pOHZlRzF3Y0M1bGVHRnRjR3hsTG1OdmJRMEt
JRHd2YzJGdGJEcEpjM04xWlhJK0RRb2dQSA0KTmhiV3h3T2s1aGJXVkpSRkJ2
YkdsamVTQjRiV3h1Y3pwellXMXNjRDBpZFhKdU9tOWhjMmx6T201aGJXVg0Ke
k9uUmpPbE5CVFV3Nk1pNHdPbkJ5YjNSdlkyOXNJZzBLSUNBZ0lDQkdiM0p0WV
```

```
   hROUluVnlianB2WVhOcA0KY3pwdVlXMWxjenAwWXpwVFFVMU1Pakl1TURwdVl
   XMWxhV1F0Wm05eWJXRjBPbkJsY25OcGMzUmxiblFpRA0KUW9nSUNBZ0lGTlFU
   bUZ0WlZGMVlXeHBabWxsY2owaWVHMXdjQzVsZUdGdGNHeGxMbU52YlNJZ1FXe
   HNiMw0KZERjbVZoZEdVOUluUnlkV1VpSUM4K0RRb2dQSE5oYld4d09sSmxjWF
   ZsYzNSbFpFRjFkR2h1UTI5dWRHVg0KNGRBMEtJQ0FnSUNCNGJXeHVjenB6WVc
   xc2NEMGlkWEp1T205aGMybHpPbTVoYldWek9uUmpPbE5CVFV3Ng0KTWk0d09u
   QnliM1J2WTI5c0lpQU5DaUFnSUNBZ0lDQWdRMjl0Y0dGeWFYTnZiajBpWlhoa
   FkzUWlQZzBLSQ0KQ0E4YzJGdGJEcEJkWFJvYmtOdmJuUmxlSFJEYkdGemMxSm
   xaZzBLSUNBZ0lDQWdlRzFzYm5NNmMyRnRiRA0KMGlkWEp1T205aGMybHpPbTV
   oYldWek9uUmpPbE5CVFV3Nk1pNHdPbUZ6YzJWeWRHbHZiaUkrRFFvZ0lDQQ0K
   Z0lDQjFjbTQ2YjJGemFYTTZibUZ0WlhNNmRHTTZVMEZOVERveUxqQTZZV002W
   TJ4aGMzTmxjenBRWVhOeg0KZDI5eVpGQnliM1JsWTNSbFpGUnlZVzV6Y0c5eW
   RBMEtJQ0E4TDNOaGJXdzZRWFYwYUc1RGIyNTBaWGgwUQ0KMnhoYzNOU1pXWSt
   EUW9nUEM5ellXMXNjRHBTWlhGMVpYTjBaV1JCZFhSb2JrTnZiblJsZUhRK0lB
   MEtQQw0KOXpZVzFzY0RwQmRYUm9ibEpsY1hWbGMzUSs=
   C: PQ==
   S: . OK Success (TLS protection)
```

The decoded challenge is as follows:

```
   https://saml.example.org/SAML/Browser?SAMLRequest=PHNhbWxwOkF
   1dGhuUmVxdWVzdCB4bWxuczpzYW1scD0idXJuOm9hc2lzOm5hbWVzOnRjOlNB
   TUw6Mi4wOnByb3RvY29sIg0KICAgIElEPSJfYmVjNDI0ZmE1MTAzNDI4OTA5Y
   TMwZmYxZTMxMTY4MzI3Zjc5NDc0OTg0IiBWZXJzaW9uPSIyLjAiDQogICAgSX
   NzdWVJbnN0YW50PSIyMDA3LTEyLTEwVDExOjM5OjM0WiIgRm9yY2VBdXRobj0
   iZmFsc2UiDQogICAgSXNQYXNzaXZlPSJmYWxzZSINCiAgICBQcm90b2NvbEJp
   bmRpbmc9InVybjpvYXNpczpuYW1lczp0YzpTQU1MOjIuMDpiaW5kaW5nczpIV
   FRQLVBPU1QiDQogICAgQXNzZXJ0aW9uQ29uc3VtZXJTZXJ2aWNlVVJMPQ0KIC
   AgICAgICAiaHR0cHM6Ly9tYWlsLmV4YW1wbGUuY29tL1NBTUwvQXNzZXJ0aW9
   uQ29uc3VtZXJTZXJ2aWNlIj4NCiA8c2FtbDpJc3N1ZXIgeG1sbnM6c2FtbD0i
   dXJuOm9hc2lzOm5hbWVzOnRjOlNBTUw6Mi4wOmFzc2VydGlvbiI+DQogICAgI
   Gh0dHBzOi8veG1wcC5leGFtcGxlLmNvbQ0KIDwvc2FtbDpJc3N1ZXI+DQogPH
   NhbWxwOk5hbWVJRFBvbGljeSB4bWxuczpzYW1scD0idXJuOm9hc2lzOm5hbWV
   zOnRjOlNBTUw6Mi4wOnByb3RvY29sIg0KICAgICBGb3JtYXQ9InVybjpvYXNp
   czpuYW1lczp0YzpTQU1MOjIuMDpuYW1laWQtZm9ybWF0OnBlcnNpc3RlbnQiD
   QogICAgIFNQTmFtZVF1YWxpZmllcj0ieG1wcC5leGFtcGxlLmNvbSIgQWxsb3
   dDcmVhdGU9InRydWUiIC8+DQogPHNhbWxwOlJlcXVlc3RlZEF1dGhuQ29udGV
   4dA0KICAgICB4bWxuczpzYW1scD0idXJuOm9hc2lzOm5hbWVzOnRjOlNBTUw6
   Mi4wOnByb3RvY29sIiANCiAgICAgICAgQ29tcGFyaXNvbj0iZXhhY3QiPg0KI
   CA8c2FtbDpBdXRobkNvbnRleHRDbGFzc1JlZg0KICAgICAgeG1sbnM6c2FtbD
   0idXJuOm9hc2lzOm5hbWVzOnRjOlNBTUw6Mi4wOmFzc2VydGlvbiI+DQogICA
   gICB1cm46b2FzaXM6bmFtZXM6dGM6U0FNTDoyLjA6YWM6Y2xhc3NlczpQYXNz
   d29yZFByb3RlY3RlZFRyYW5zcG9ydA0KICA8L3NhbWw6QXV0aG5Db250ZXh0Q
   2xhc3NSZWY+DQogPC9zYW1scDpSZXF1ZXN0ZWRBdXRobkNvbnRleHQ+IA0KPC
   9zYW1scDpBdXRoblJlcXVlc3Q+
```

Where the decoded SAMLRequest looks like the following:

```
<samlp:AuthnRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    ID="_bec424fa5103428909a30ff1e31168327f79474984" Version="2.0"
    IssueInstant="2007-12-10T11:39:34Z" ForceAuthn="false"
    IsPassive="false"
    ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
    AssertionConsumerServiceURL=
        "https://mail.example.com/SAML/AssertionConsumerService">
 <saml:Issuer xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
     https://xmpp.example.com
 </saml:Issuer>
 <samlp:NameIDPolicy xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
     Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent"
     SPNameQualifier="xmpp.example.com" AllowCreate="true" />
 <samlp:RequestedAuthnContext
     xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
        Comparison="exact">
  <saml:AuthnContextClassRef
      xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
      urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
  </saml:AuthnContextClassRef>
 </samlp:RequestedAuthnContext>
</samlp:AuthnRequest>
```

## 6. Security Considerations

This section addresses only security considerations associated with the use of SAML with SASL applications. For considerations relating to SAML in general, and for general SASL security considerations, the reader is referred to the SAML specifications and to other literature.

### 6.1. Man-in-the-Middle and Tunneling Attacks

This mechanism is vulnerable to man-in-the-middle and tunneling attacks unless a client always verifies the server’s identity before proceeding with authentication (see [RFC6125]). Typically, TLS is used to provide a secure channel with server authentication.

### 6.2. Binding SAML Subject Identifiers to Authorization Identities

As specified in [RFC4422], the server is responsible for binding credentials to a specific authorization identity. It is therefore necessary that only specific trusted IdPs be allowed. This is a typical part of SAML trust establishment between RPs and the IdP.

### 6.3. User Privacy

The IdP is aware of each RP that a user logs into. There is nothing in the protocol to hide this information from the IdP. It is not a requirement to track the visits, but there is nothing that prohibits the collection of information. SASL server implementers should be aware that SAML IdPs will be able to track -- to some extent -- user access to their services.

### 6.4. Collusion between RPs

It is possible for RPs to link data that they have collected on the users. By using the same identifier to log into every RP, collusion between RPs is possible. In SAML, targeted identity was introduced. Targeted identity allows the IdP to transform the identifier the user typed in to an RP-specific opaque identifier. This way, the RP would never see the actual user identifier but instead would see a randomly generated identifier.

### 6.5. Security Considerations Specific to GSS-API

Security issues inherent in GSS-API [RFC2743] and GS2 [RFC5801] apply to the SAML GSS-API mechanism defined in this document. Further, and as discussed in Section 4, proper TLS server identity verification is critical to the security of the mechanism.

## 7. IANA Considerations

### 7.1. IANA Mech-Profile

The IANA has registered the following SASL profile:

SASL mechanism profile: SAML20

Security Considerations: See this document

Published Specification: See this document

For further information: Contact the authors of this document.

Owner/Change controller: the IETF

Intended usage: COMMON

Note: None

### 7.2. IANA OID

The IANA has also assigned a new entry for this GSS mechanism in the SMI Security for Mechanism Codes sub-registry, whose prefix is iso.org.dod.internet.security.mechanisms (1.3.6.1.5.5), and referenced this specification in the registry.

## 8. References

### 8.1. Normative References

[OASIS-SAMLv2-BIND]
Cantor, S., Ed., Hirsch, F., Ed., Kemp, J., Ed., Philpott, R., Ed., and E. Maler, Ed., "Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS Standard saml-bindings-2.0-os, March 2005, <http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf>.

[OASIS-SAMLv2-CORE]
Cantor, S., Ed., Kemp, J., Ed., Philpott, R., Ed., and E. Maler, Ed., "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS Standard saml-core-2.0-os, March 2005, <http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf>.

[OASIS-SAMLv2-PROF]
Hughes, J., Ed., Cantor, S., Ed., Hodges, J., Ed., Hirsch, F., Ed., Mishra, P., Ed., Philpott, R., Ed., and E. Maler, Ed., "Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS Standard OASIS.saml-profiles-2.0-os, March 2005, <http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC2743] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, January 2000.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[RFC3492] Costello, A., "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)", RFC 3492, March 2003.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, January 2005.

[RFC4422] Melnikov, A., Ed., and K. Zeilenga, Ed., "Simple Authentication and Security Layer (SASL)", RFC 4422, June 2006.

[RFC5056] Williams, N., "On the Use of Channel Bindings to Secure Channels", RFC 5056, November 2007.

[RFC5234] Crocker, D., Ed., and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[RFC5801] Josefsson, S. and N. Williams, "Using Generic Security Service Application Program Interface (GSS-API) Mechanisms in Simple Authentication and Security Layer (SASL): The GS2 Mechanism Family", RFC 5801, July 2010.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010.

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, August 2010.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, March 2011.

[W3C-REC-HTML401] Le Hors, A., Ed., Raggett, D., Ed., and I. Jacobs, Ed., "HTML 4.01 Specification", World Wide Web Consortium Recommendation REC-html401-19991224, December 1999, <http://www.w3.org/TR/1999/REC-html401-19991224>.

## 8.2. Informative References

[RFC1939] Myers, J. and M. Rose, "Post Office Protocol - Version 3", STD 53, RFC 1939, May 1996.

[RFC3501] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, March 2003.

[RFC4401] Williams, N., "A Pseudo-Random Function (PRF) API Extension for the Generic Security Service Application Program Interface (GSS-API)", RFC 4401, February 2006.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, March 2011.

## Appendix A. Acknowledgments

The authors would like to thank Scott Cantor, Joe Hildebrand, Josh Howlett, Leif Johansson, Thomas Lenggenhager, Diego Lopez, Hank Mauldin, RL "Bob" Morgan, Stefan Plug, and Hannes Tschofenig for their review and contributions.

## Authors' Addresses

Klaas Wierenga
Cisco Systems, Inc.
Haarlerbergweg 13-19
1101 CH Amsterdam
The Netherlands

Phone: +31 20 357 1752
EMail: klaas@cisco.com

Eliot Lear
Cisco Systems GmbH
Richtistrasse 7
CH-8304 Wallisellen
Switzerland

Phone: +41 44 878 9200
EMail: lear@cisco.com

Simon Josefsson
SJD AB
Johan Olof Wallins vag 13
Solna 171 64
Sweden

EMail: simon@josefsson.org
URI: http://josefsson.org/