Internet Engineering Task Force (IETF)
Request for Comments: 6616
Category: Standards Track
ISSN: 2070-1721

E. Lear
Cisco Systems GmbH
H. Tschofenig
Nokia Siemens Networks
H. Mauldin
Cisco Systems, Inc.
S. Josefsson
SJD AB

May 2012

A Simple Authentication and Security Layer (SASL) and Generic Security Service Application Program Interface (GSS-API) Mechanism for OpenID

Abstract

OpenID has found its usage on the Internet for Web Single Sign-On. 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 and GSS-API mechanism for OpenID that allows the integration of existing OpenID 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/rfc6616.

Table of Contents

1. Introduction

OpenID 2.0 [OpenID] is a web-based three-party protocol that provides a means for a user to offer identity assertions and other attributes to a web server (Relying Party) via the help of an identity provider. The purpose of this system is to provide a way to verify that an end user controls an identifier.

Simple Authentication and Security Layer (SASL) [RFC4422] is used by application protocols such as IMAP [RFC3501], Post Office Protocol (POP) [RFC1939], and Extensible Messaging and Presence Protocol (XMPP) [RFC6120], with the goal of modularizing authentication and security layers, so that newer 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 interface. This document defines a pure SASL mechanism for OpenID, 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. Implementors of the SASL component MAY implement the GSS-API interface as well.

This mechanism specifies interworking between SASL and OpenID in order to assert identity and other attributes to Relying Parties. As such, while SASL servers (as Relying Parties) will advertise SASL mechanisms, clients will select the OpenID mechanism.

The OpenID mechanism described in this memo aims to reuse the OpenID mechanism to the maximum extent and therefore does not establish a separate authentication, integrity, and confidentiality mechanism. It is anticipated that existing security layers, such as Transport Layer Security (TLS) [RFC5246], continue to be used. Minimal changes are required to non-web applications, as most of the transaction occurs through a normal web browser. Hence, this specification is only appropriate for use when such a browser is available.

Figure 1 describes the interworking between OpenID and SASL. This document requires enhancements to the Relying Party and to the Client (as the two SASL communication end points), but no changes to the OpenID Provider (OP) are necessary. To accomplish this goal, indirect messaging required by the OpenID specification is tunneled through the SASL/GSS-API mechanism.

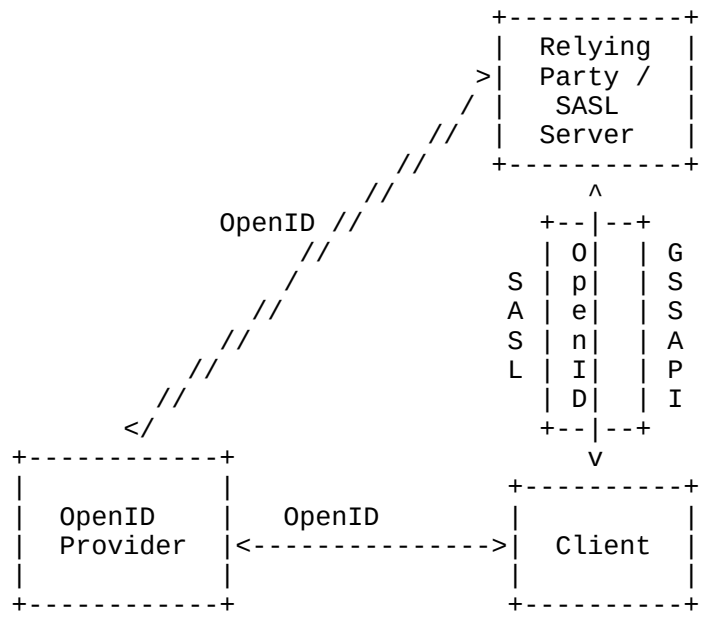


Figure 1: Interworking Architecture

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 OpenID 2.0 specification.

1.2. Applicability

Because this mechanism transports information that should not be controlled by an attacker, the OpenID mechanism MUST only be used over channels protected by TLS, and the client MUST successfully validate the server certificate [RFC5280][RFC6125].

2. Applicability for Application Protocols other than HTTP

OpenID was originally envisioned for HTTP- [RFC2616] and HTML-based [W3C.REC-html401-19991224] communications, and with the associated semantic; the idea being that the user would be redirected by the Relying Party (RP) to an identity provider (IdP) who authenticates the user and then sends identity information and other attributes (either directly or indirectly) to the Relying Party. The identity

provider in the OpenID specifications is referred to as an OpenID Provider (OP). The actual protocol flow can be found in Section 3 of the OpenID 2.0 specification [OpenID]. The reader is strongly encouraged to be familiar with that specification before continuing.

When considering that flow in the context of SASL, we note that while the RP and the client both need to change their code to implement this SASL mechanism, it is a design constraint that the OP behavior remain untouched, in order for implementations to interoperate with existing IdPs. Hence, an analog flow that interfaces the three parties needs to be created. In the analog, we note that unlike a web server, the SASL server already has some sort of session (probably a TCP connection) established with the client. However, it may be necessary for a SASL client to invoke to another application. This will be discussed below. By doing so, we externalize much of the authentication from SASL.

The steps are listed below:

1. The SASL server advertises support for the SASL OpenID mechanism to the client.

2. The client initiates a SASL authentication and transmits the User-Supplied Identifier as its first response. The SASL mechanism is client-first, and, as explained in [RFC4422], the server will send an empty challenge if needed.

3. After normalizing the User-Supplied Identifier as discussed in [OpenID], the Relying Party performs discovery on it and establishes the OP Endpoint URL that the end user uses for authentication.

4. The Relying Party and the OP optionally establish an association -- a shared secret established using Diffie-Hellman Key Exchange. The OP uses an association to validate those messages through the use of a Hashed Message Authentication Code (HMAC); this removes the need for subsequent direct requests to verify the signature after each authentication request/response.

5. The Relying Party transmits an authentication request to the OP to obtain an assertion in the form of an indirect request. These messages are passed through the client rather than directly between the RP and the OP. OpenID defines two methods for indirect communication -- namely, HTTP redirects and HTML form submission. Neither mechanism is directly applicable for usage with SASL. To ensure that an OP that is OpenID 2.0 capable can be used, a new method is defined in this document that requires the OpenID message content to be encoded using a

Universal Resource Identifier (URI) [RFC3986]. Note that any Internationalized Resource Identifiers (IRIs) must be normalized to URIs by the SASL client, as specified in [RFC3987], prior to transmitting them to the SASL server.

6. The SASL client now sends a response consisting of "=" to the server, to indicate that authentication continues via the normal OpenID flow.

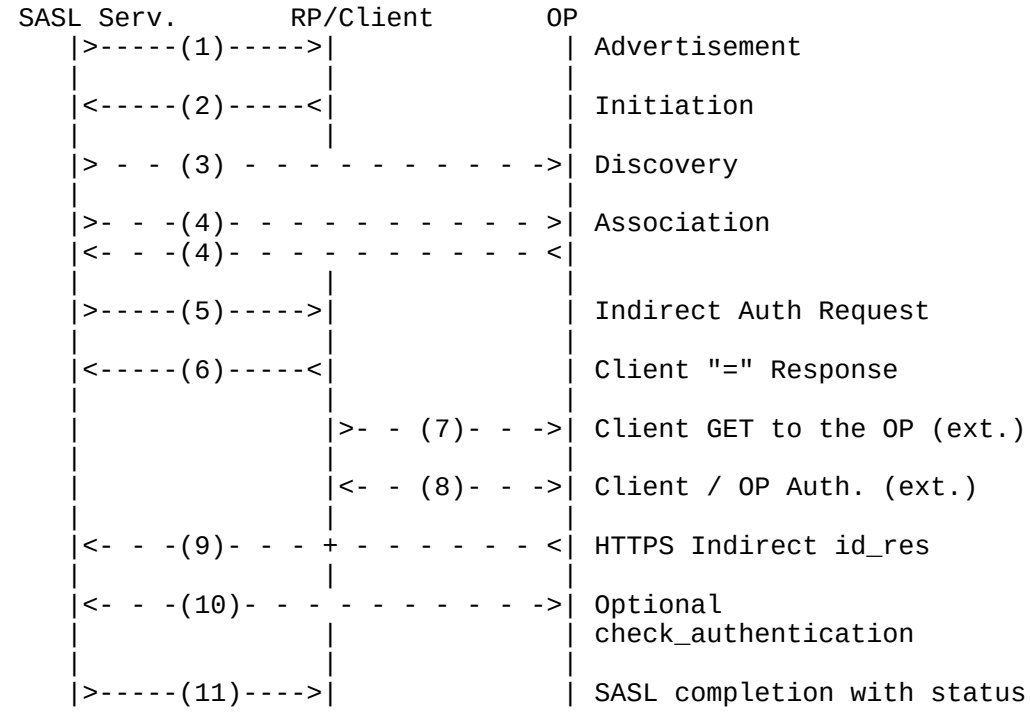
7. At this point, the client application MUST construct a URL containing the content received in the previous message from the RP. This URL is transmitted to the OP by either the SASL client application or an appropriate handler, such as a browser.

8. Next, the end user optionally authenticates to the OP and then, depending on the OP, may approve or disapprove authentication to the Relying Party. For reasons of its own, the OP has the option of not authenticating a request. The manner in which the end user is authenticated to their respective OP and any policies surrounding such authentication are out of scope of OpenID and, hence, also out of scope for this specification. This step happens out of band from SASL.

9. The OP will convey information about the success or failure of the authentication phase back to the RP, again using an indirect response via the client browser or handler. The client transmits to the RP (over HTTP/TLS) the redirect of the OP result. This step happens out of band from SASL.

10. The RP MAY send an OpenID check_authentication request directly to the OP, if no association has been established, and the OP should respond. Again, this step happens out of band from SASL.

11. The SASL server sends an appropriate SASL response to the client, with optional Open Simple Registry (SREG) attributes.



Note the directionality in SASL is such that the client MUST send the "=" response. Specifically, the SASL client processes the redirect and then awaits a final SASL decision, while the rest of the OpenID authentication process continues.

2.1. Binding SASL to OpenID in the Relying Party

OpenID is meant to be used in serial within the web, where browser cookies are easily accessible. As such, there are no transaction IDs within the protocol. To ensure that a specific request is bound, and in particular to ease inter-process communication, the Relying Party MUST encode a nonce or transaction ID in the URIs it transmits through the client for success or failure, as either a base URI or fragment component to the "return_to" URI. This value is to be used to uniquely identify each authentication transaction. The nonce value MUST be at least 2^32 bits and large enough to handle well in excess of the number of concurrent transactions a SASL server shall see.

2.2. Discussion

As mentioned above, OpenID is primarily designed to interact with web-based applications. Portions of the authentication stream are only defined in the crudest sense. That is, when one is prompted to approve or disapprove an authentication, anything that one might find on a browser is allowed, including JavaScript, complex style-sheets, etc. Because of this lack of structure, implementations will need to invoke a rich browser in order to ensure that the authentication can be completed.

Once there is an outcome, the SASL server needs to know about it. The astute reader will hopefully by now have noticed an "=" client SASL response. This is not to say that nothing is happening, but rather that authentication flow has shifted from SASL and the client application to OpenID within the browser, and it will return to the client application when the server has an outcome to hand to the client. The alternative to this flow would be some sort of signal from the HTML browser to the SASL client of the results that would in turn be passed to the SASL server. The inter-process communication issue this raises is substantial. Better, we conclude, to externalize the authentication to the browser and have an "=" client response.

3. OpenID SASL Mechanism Specification

This section specifies the details of the OpenID SASL mechanism. Recall Section 5 of [RFC4422] for what needs to be described here.

The name of this mechanism is "OPENID20". The mechanism is capable of transferring an authorization identity (via "gs2-header"). The mechanism does not offer a security layer.

The mechanism is client-first. The first mechanism message is from the client to the server, and it is the "initial-response" described below. 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, and it is the "authentication_request" described below.

The third mechanism message is from client to the server, and it is the fixed message consisting of "=".

The fourth mechanism message is from the server to the client, described below as "outcome_data" (with SREG attributes), sent as additional data when indicating a successful outcome.

3.1. Initiation

A client initiates an OpenID authentication with SASL by sending the GS2 header followed by the URI, as specified in the OpenID specification.

The ABNF [RFC5234] syntax is as follows:

```
initial-response = gs2-header Auth-Identifier
Auth-Identifier = Identifier ; authentication identifier
Identifier = URI             ; Identifier is specified in
                             ; Sec. 7.2 of the OpenID 2.0 spec.
```

The syntax and semantics of the "gs2-header" are specified in [RFC5801], and we use it here with the following limitations: The "gs2-nonstd-flag" MUST NOT be present. The "gs2-cb-flag" MUST be "n" because channel binding is not supported by this mechanism.

URI is specified in [RFC3986]. Extensible Resource Identifiers (XRIs) [XRI2.0] MUST NOT be used.

3.2. Authentication Request

The SASL server sends the URL resulting from the OpenID authentication request, containing an "openid.mode" of either "checkid_immediate" or "checkid_setup", as specified in Section 9.1 of the OpenID 2.0 specification [OpenID].

```
authentication-request = URI
```

As part of this request, the SASL server MUST append a unique transaction ID to the "return_to" portion of the request. The form of this transaction is left to the RP to decide, but it SHOULD be large enough to be resistant to being guessed or attacked.

The client now sends that request via an HTTP GET to the OP, as if redirected to do so from an HTTP server.

The client MUST handle both user authentication to the OP and confirmation or rejection of the authentication by the RP via this SASL mechanism.

After all authentication has been completed by the OP, and after the response has been sent to the client, the client will relay the response to the Relying Party via HTTP/TLS, as specified previously in the transaction ("return_to").

3.3. Server Response

The Relying Party now validates the response it received from the client via HTTP/TLS, as specified in the OpenID specification, using the "return_to" URI given previously in the transaction.

The response by the Relying Party constitutes a SASL mechanism outcome, and it SHALL be used to set state in the server accordingly. Also, it SHALL be used by the server to report that state to the SASL client as described in Section 3.6 of [RFC4422]. In the additional data, the server MAY include OpenID Simple Registry (SREG) attributes that are listed in Section 4 of [SREG1.0]. SREG attributes are encoded as follows:

1. Strip "openid.sreg." from each attribute name.

2. Treat the concatenation of results as URI parameters that are separated by an ampersand (&) and encode as one would a URI, absent the scheme, authority, and the question mark.

For example: email=lear@example.com&fullname=Eliot%20Lear

More formally:

```
outcome-data = [ sreg-avp *( "," sreg-avp ) ]
sreg-avp     = sreg-attr "=" sreg-val
sreg-attr    = sreg-word
sreg-val     = sreg-word
sreg-word    = 1*( unreserved / pct-encoded )
               ; pct-encoded from Section 2.1 of RFC 3986
               ; unreserved from Section 2.3 of RFC 3986
```

A client who does not support SREG MUST ignore SREG attributes sent by the server. Similarly, a client MUST ignore unknown attributes.

In the case of failures, the response MUST follow this syntax:

```
outcome-data = "openid.error" "=" sreg-val *( "," sregp-avp )
```

3.4. Error Handling

Section 3.6 of [RFC4422] explicitly prohibits additional information in an unsuccessful authentication outcome. Therefore, the openid.error and openid.error_code are to be sent as an additional challenge in the event of an unsuccessful outcome. In this case, as the protocol is in lockstep, the client will follow with an additional exchange containing "=", after which the server will respond with an application-level outcome.

4. OpenID GSS-API Mechanism Specification

This section MUST be observed to properly implement the GSS-API mechanism that is described below.

The OpenID SASL mechanism is actually also a GSS-API mechanism. The OpenID user takes the role of the GSS-API Initiator and the OpenID Relying Party takes the role of the GSS-API Acceptor. The OpenID Provider does not have a role in GSS-API and is considered an internal matter for the OpenID mechanism. The messages are the same, but a) the GS2 header on the client's first message and channel binding data are excluded when OpenID is used as a GSS-API mechanism, and b) the initial context token header (described in Section 3.1 of RFC 2743) is prefixed to the client's first authentication message (context token).

The GSS-API OID for the OpenID 2.0 mechanism is 1.3.6.1.5.5.16 (see Section 7 for more information). The DER encoding of the OID is 0x2b 0x06 0x01 0x05 0x05 0x10.

OpenID security contexts MUST have the mutual_state flag (GSS_C_MUTUAL_FLAG) set to TRUE. OpenID does not support credential delegation; therefore, OpenID 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 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 identity with the target name, as discussed in [RFC6125].

The OpenID mechanism does not support per-message tokens or GSS_Pseudo_random.

The [RFC5587] mechanism attributes for this mechanism are GSS_C_MA_MECH_CONCRETE, GSS_C_MA_ITOK_FRAMED, and GSS_C_MA_AUTH_INIT.

4.1. GSS-API Principal Name Types for OpenID

OpenID supports standard generic name syntaxes for acceptors such as GSS_C_NT_HOSTBASED_SERVICE (see Section 4.1 of [RFC2743]).

OpenID supports only a single name type for initiators: GSS_C_NT_USER_NAME. GSS_C_NT_USER_NAME is the default name type for OpenID.

OpenID name normalization is covered by the OpenID specification; see Section 7.2 of [OpenID].

The query, display, and exported name syntaxes for OpenID principal names are all the same. There are no OpenID-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 Section 4 of [RFC2743]). The exported name token does, of course, conform to Section 3.2 of [RFC2743], but the "NAME" part of the token should be treated as a potential input string to the OpenID name normalization rules. For example, the OpenID Identifier "https://openid.example/" will have a GSS_C_NT_USER_NAME value of "https://openid.example/".

GSS-API name attributes may be defined in the future to hold the normalized OpenID Identifier.

5. Example

Suppose a user has an OpenID of https://openid.example and wishes to authenticate his IMAP connection to mail.example (where .example is the top-level domain specified in [RFC2606]). The user would input his OpenID into his mail user agent when he configures the account. In this case, no association is attempted between the OpenID RP and the OP. The client will make use of the "return_to" attribute to capture results of the authentication to be redirected to the server. Note the use of [RFC4959] for the initial response. The authentication on the wire would then look something like the following:

```
(S = IMAP server; C = IMAP client)

C: < connects to IMAP port>
S: * OK
C: C1 CAPABILITY
S: * CAPABILITY IMAP4rev1 SASL-IR SORT [...] AUTH=OPENID20
S: C1 OK Capability Completed
C: C2 AUTHENTICATE OPENID biwsaHR0cHM6Ly9vcGVuaWQuZXhhbXBsZS8=
[  This is the base64 encoding of "n,,https://openid.example/".
   Server performs discovery on http://openid.example/ ]
S: + aHR0cHM6Ly9vcGVuaWQuZXhhbXBsZS9vcGVuaWQvP29wZW5pZC5ucz1
     odHRwOi8vc3BlY3Mub3BlbmlkLm5ldC9hdXRoLzIuMCZvcGVuaWQucm
     V0dXJuX3RvPWh0dHBzOi8vbWFpbC5leGFtcGxlL2NvbnN1bWVyLzFlZ
     jg4OGMmb3BlbmlkLmNsYWltZWRfaWQ9aHR0cHM6Ly9vcGVuaWQuZXhh
     bXBsZS8mb3BlbmlkLmlkZW50aXR5PWh0dHBzOi8vb3BlbmlkLmV4YW1
     wbGUvJm9wZW5pZC5yZWFsbT1pbWFwOi8vbWFpbC5leGFtcGxlJm9wZW
     5pZC5tb2RlPWNoZWNraWRfc2V0dXA=

[ This is the base64 encoding of "https://openid.example/openid/
      ?openid.ns=http://specs.openid.net/auth/2.0
      &openid.return_to=https://mail.example/consumer/1ef888c
      &openid.claimed_id=https://openid.example/
      &openid.identity=https://openid.example/
      &openid.realm=imap://mail.example
      &openid.mode=checkid_setup"
   with line breaks and spaces added here for readability.
]
C: PQ==
[ The client now sends the URL it received to a browser for
  processing.  The user logs into https://openid.example and
  agrees to authenticate imap://mail.example.  A redirect is
  passed back to the client browser that then connects to
  https://imap.example/consumer via SSL with the results.
  From an IMAP perspective, however, the client sends the "="
  response, and awaits mail.example.
  Server mail.example would now contact openid.example with an
  openid.check_authentication message.  After that...
]
S: + ZW1haWw9bGVhckBtYWlsLmV4YW1wbGUsZnVsbG5hbWU9RWxp
     b3QlMjBMZWFy
  [ Here, the IMAP server has returned an SREG attribute of
    email=lear@mail.example,fullname=Eliot%20Lear.
    Line break in response added in this example for readability. ]
C:
  [ In IMAP, client must send a blank response after receiving
    the SREG data. ]
S: C2 OK
```

In this example, the SASL server / RP has made use of a transaction ID 1ef888c.

6. Security Considerations

This section will address only security considerations associated with the use of OpenID with SASL and GSS-API. For considerations relating to OpenID in general, the reader is referred to the OpenID specification [OpenID] and to other literature [OpReview]. Similarly, for general SASL [RFC4422] and GSS-API [RFC5801] security considerations, the reader is referred to those specifications.

6.1. Binding OpenIDs to Authorization Identities

As specified in [RFC4422], the server is responsible for binding credentials to a specific authorization identity. It is therefore necessary that a registration process takes place in advance that binds specific OpenIDs to specific authorization identities, or that only specific trusted OpenID Providers be allowed, where a mapping is predefined. For example, it could be prearranged between an IdP and RP that "https://example.com/user" maps to "user" for purposes of authorization.

6.2. RP Redirected by Malicious URL to Take an Improper Action

In the initial SASL client response, a user or host can transmit a malicious response to the RP for purposes of taking advantage of weaknesses in the RP's OpenID implementation. It is possible to add port numbers to the URL so that the outcome is that the RP does a port scan of the site. The URL could contain an unauthorized host or even the local host. The URL could contain a protocol other than http or https, such as file or ftp.

One mitigation would be for RPs to have a list of authorized URI bases. OPs SHOULD only redirect to RPs with the same domain component of the base URI. RPs MUST NOT automatically retry on failed attempts. A log of those sites that fail SHOULD be kept, and limitations on queries from clients SHOULD be imposed, just as with any other authentication attempt. Applications SHOULD NOT invoke browsers to communicate with OPs that they are not themselves configured with.

6.3. User Privacy

The OP is aware of each RP that a user logs into. There is nothing in the protocol to hide this information from the OP. It is not a requirement to track the visits, but there is nothing that prohibits the collection of information. SASL servers should be aware that

OpenID Providers will be able to track -- to some extent -- user access to their services and any additional information that OP provides.

7. IANA Considerations

IANA has updated the "SASL Mechanisms" registry using the following template, as described in [RFC4422].

SASL mechanism name: OPENID20

Security Considerations: See this document

Published specification: See this document

Person & email address to contact for further information: Authors of this document

Intended usage: COMMON

Owner/Change controller: IESG

Note: None

IANA has also assigned an OID for this GSS mechanism in the "SMI Security for Mechanism Codes" registry, with the prefix of iso.org.dod.internet.security.mechanisms (1.3.6.1.5.5) and referencing this specification in the registry.

8. Acknowledgments

The authors would like to thank Alexey Melnikov, Joe Hildebrand, Mark Crispin, Chris Newman, Leif Johansson, Sam Hartman, Nico Williams, Klaas Wierenga, Stephen Farrell, and Stephen Kent for their review and contributions.

9. References

9.1. Normative References

[OpenID] OpenID Foundation, "OpenID Authentication 2.0 - Final", December 2007, <http://specs.openid.net/auth/2.0>.

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

[RFC2606] Eastlake, D. and A. Panitz, "Reserved Top Level DNS Names", BCP 32, RFC 2606, June 1999.

[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.

[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. and K. Zeilenga, "Simple Authentication and Security Layer (SASL)", RFC 4422, June 2006.

[RFC5234] Crocker, D. 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.

[RFC5587] Williams, N., "Extended Generic Security Service Mechanism Inquiry APIs", RFC 5587, July 2009.

[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.

[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.

[SREG1.0] OpenID Foundation, "OpenID Simple Registration Extension version 1.0", June 2006, <http://openid.net/sreg/1.0>.

9.2. Informative References

[OpReview] "Google Sites OpenID Reference Page", <http://sites.google.com/site/openidreview/resources>.

[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.

[RFC4959] Siemborski, R. and A. Gulbrandsen, "IMAP Extension for Simple Authentication and Security Layer (SASL) Initial Client Response", RFC 4959, September 2007.

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

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

[XRI2.0] Reed, D., Ed. and D. McAlpin, Ed., "Extensible Resource Identifier (XRI) Syntax V2.0", OASIS Standard xri-syntax-V2.0-cs, September 2005, <http://www.oasis-open.org/committees/download.php/15376/xri-syntax-V2.0-cs.html>.

Authors' Addresses

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

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

Hannes Tschofenig
Nokia Siemens Networks
Linnoitustie 6
Espoo 02600
Finland

Phone: +358 (50) 4871445
EMail: Hannes.Tschofenig@gmx.net
URI: http://www.tschofenig.priv.at

Henry Mauldin
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134
USA

Phone: +1 (800) 553-6387
EMail: hmauldin@cisco.com

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

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