Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 23, 2010

E. Lear
Cisco Systems GmbH
H. Tschofenig
Nokia Siemens Networks
H. Mauldin
Cisco Systems, Inc.

January 19, 2010

A SASL Mechanism for OpenID
draft-lear-ietf-sasl-openid-00.txt

Abstract

OpenID has found its usage on the Internet for Web Single Sign-On. Simple Authentication and Security Layer (SASL) is an application framework to generalize authentication. This memo specifies a SASL mechanism for OpenID that allows the integration of existing OpenID Identity Providers with applications using SASL.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

OpenID [OpenID] is a 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] (SASL) is used by application protocols such IMAP, POP and XMPP, 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.

As currently envisioned, this mechanism is to allow the interworking between SASL and OpenID in order to assert identity and other attributes to relying parties. As such, while servers (as relying parties) will advertise SASL mechanisms, clients will select the OpenID mechanism.

The OpenID mechanism described in this memo aims to re-use the available OpenID specification to a 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), will continued to be used.

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 within SASL.

```
                                    +-----------+
                                    |           |
                                   >|  Relying  |
                                  / |  Party    |
                                //  |           |
                              //    +-----------+
                            //            ^
                   OpenID //           +--|--+
                        //             | O|  |
                       /             S | p|  |
                     //              A | e|  |
                   //                S | n|  |
                 //                  L | I|  |
               //                      | D|  |
             </                        +--|--+
      +------------+                      v
      |            |                 +----------+
      |  OpenID    |   OpenID        |          |
      |  Provider  |<--------------->|  Client  |
      |            |                 |          |
      +------------+                 +----------+
```

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.

2. Applicability for non-HTTP Use Cases

OpenID was originally envisioned for HTTP/HTML based communications, and with the associated semantic, the idea being that the user would be redirected by the Relying Party to an identity provider who authenticates the user, and then sends identity information and other attributes (either directly or indirectly) to the Relying Party. The actual protocol flow, as copied from the OpenID 2.0 specification, is as follows:

1. The end user initiates authentication by presenting a User-Supplied Identifier to the Relying Party via their User-Agent (e.g., http://user.example.com).

2. After normalizing the User-Supplied Identifier, the Relying Party performs discovery on it and establishes the OP Endpoint URL that the end user uses for authentication. It should be noted that the User-Supplied Identifier may be an OP Identifier, which allows selection of a Claimed Identifier at the OP or for the protocol to proceed without a Claimed Identifier if something else useful is being done via an extension.

3. 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 sign subsequent messages and the Relying Party to verify those messages; this removes the need for subsequent direct requests to verify the signature after each authentication request/response.

4. The Relying Party redirects the end user's User-Agent to the OP with an OpenID Authentication request. This occurs as stated in Section 10.3 of [RFC2616].

5. The OP authenticates the end user and establishes whether the end user will authenticate to, and share specific attributes with, the Relying Party. For instance, the OP often asks the user what to do. The manner in which the end user authenticates to their OP and any policies surrounding such authentication is out of scope of OpenID.

6. The OP redirects the end user's User-Agent back to the Relying Party with either an assertion that authentication is approved or a message that authentication failed.

7. The Relying Party verifies the information received from the OP including checking the Return URL, verifying the discovered information, checking the nonce, and verifying the signature by using either the shared key established during the association or

by sending a direct request to the OP.

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 OP must remain untouched. 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 to redirect a SASL client to another application. This will be discussed below. By doing so, we externalize much of the authentiction from SASL.

The steps are shown from below:

1. The Relying Party or SASL server advertises support for the SASL

OpenID mechanism to the client.

2. The client initiates a SASL authentiation and transmits the User-Supplied Identifier as well as an optional return_to parameter.

3. After normalizing the User-Supplied Identifier, 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 sign subsequent messages and the Relying Party to verify those messages; 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. Both mechanisms are not directly applicable for usage with SASL. To ensure that a standard OpenID 2.0 capable OP can be used a new method is defined in this document that requires the OpenID message content to be encoded using a Universal Resource Idenitifier (URI). [RFC3986]

6. The SASL client now sends an empty response, as 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 either by the SASL client application or an appropriate handler, such as a browser.

8. Next the client optionally authenticates to the OP and then approves or disapproves authentication to the Relying Party. The manner in which the end user is authenticated to their respective OP and any policies surrounding such authentication is out of scope of OpenID and 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 over HTTP the redirect of the OP result to the RP. 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 be expected to 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.


```
         SASL Serv.       Client          OP
```

```
            |>-----(1)----->|              | Advertisement
            |               |              |
            |<-----(2)-----<|              | Initiation
            |               |              |
            |> - - (3) - - - - - - - - - ->| Discovery
            |                              |
            |>- - -(4)- - - - - - - - - - >| Association
            |<- - -(4)- - - - - - - - - - <|
            |               |              |
            |>-----(5)----->|              | Indirect Auth Request
            |               |              |
            |<-----(6)-----<|              | Client Empty Response
            |               |              |
            |               |>- - (7)- - ->| Client GET to the OP (ext)
            |               |              |
            |               |<- - (8)- - ->| Client / OP Auth. (ext.)
            |               |              |
            |<- - -(9)- - - + - - - - - - <| HTTP(s) Indirect id_res
            |               |              |
            |<- - -(10)- - - - - - - - - ->| Optional check_authenticate
            |               |              |
            |>-----(11)---->|              | SASL completion with status

        ----- = SASL
        - - - = HTTP or SSL
```

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

2.1. 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, fancy style-sheets, etc. Because of this lack of structure, implementations will need to invoke a fairly rich browser in order to insure that the authentication can be completed.

Once there is an outcome, the SASL server needs to know about it. The astute will hopefully by now have noticed an empty client SASL challenge. This is not to say that nothing is happening, but rather that authentication flow has shifted from SASL to OpenID, and will

return when the server has an outcome to hand to the client. The alternative to this flow is some signal from the HTML browser to the SASL client of the results that is in turn passed to the SASL server. The IPC issue this raises is substantial. Better, we conclude, to externalize the authentication to the browser, and have an empty client challenge.

3. OpenID SASL Mechanism Specification

Based on the previous figure, the following operations are performed with the OPENID SASL mechanism:

3.1. Advertisement

To advertise that a server supports OpenID, during application session initiation, it displays the name "OPENID" in the list of supported SASL mechanisms.

3.2. Initiation

A client initiates an OpenID authentication with SASL by the XRI or URI, as specified in the OpenID specification. Additionally, the supported version of OpenID is indicated.

```
initial-response = Identifier UTF8NUL openid-version
Identifier = URI | XRI      ;  Identifer is specified in
                            ;  Sec. 7.2 of the OpenID 2.0 spec.
; XRI as specified  by OASIS 2.0 Syntax
; URI is specified in RFC 3986.
openid-version = 1*DIGIT [ "." 1*DIGIT ]
```

The XRI syntax is defined in [XRI2.0].

3.3. Authentication Request

The SASL Server sends an OpenID message that contains an openid.mode of either "checkid_immediate" or "checkid_setup", as specified in Section 9.1 of the OpenID 2.0 specification.

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 of the RP.

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 or SSL.

3.4. Server Response

The Relying Party now validates the response it received from the client via HTTP or SSL, as specified in the OpenID specification.

The response by the Relying Party consists of an application specific response code indicating success or failure of authentication. In the additional data, the server MAY include OpenID Simple Registry (SREG) attributes that are listed in Section 4 of [SREG1.0]. They are encoded as follows:

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

2. Treat the concatentation of results as URI parameters that are separated by an ambersand (&) 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 3896
               ; unreserved from Section 2.3 of RFC 3896
```

If the application protocol allows, openid.error and openid.error_code and any other useful diagnostic information SHOULD be included in authentication failures.

4. Example

Suppose one has an OpenID of http://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 Consumer 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. 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=OPENID
S: C1 OK Capability Completed
C: C2 AUTHENTICATE OPENID aHR0cDovL29wZW5pZC5leGFtcGxlLwAy
[  This is the base64 encoding of "http://openid.example/\02"
   with line breaks and spaces added here for readibility.
   Server performs discovery on https://openid.example/ ]
S: + aHR0cDovL29wZW5pZC5leGFtcGxlL29wZW5pZC8/b3BlbmlkLm5zPWh
     0dHA6Ly9zcGVjcy5vcGVuaWQubmV0L2F1dGgvMi4wJm9wZW5pZC5yZX
     R1cm5fdG89aHR0cHM6Ly9tYWlsLmV4YW1wbGUvY29uc3VtZXImb3Blb
     mlkLmNsYWltZWRfaWQ9aHR0cHM6Ly9vcGVuaWQuZXhhbXBsZS8mb3Bl
     bmlkLmlkZW50aXR5PWh0dHBzOi8vb3BlbmlkLmV4YW1wbGUvJm9wZW5
     pZC5yZWFsbT1pbWFwOi8vbWFpbC5leGFtcGxlJm9wZW5pZC5tb2RlPW
     NoZWNraWRfc2V0dXA=
```

```
   [ This is the base64 encoding of http://openid.example/openid/
              ?openid.ns=http://specs.openid.net/auth/2.0
              &openid.return_to=https://mail.example/consumer
              &openid.claimed_id=https://openid.example/
              &openid.identity=https://openid.example/
              &openid.realm=imap://mail.example
              &openid.mode=checkid_setup

   ]
   C:
   [ The client now sends the URL it received to a browser for
     processing. The user logs into http://openid.example, and
     agrees to authenticate imap://mail.example.  A redirect is
     passed back to the client browser who then connects to
     https://imap.example/consumer via SSL with the results.
     From an IMAP perspective, however, the client sends an empty
     response, and awaits mail.example.
     Server mail.example would now contact openid.example with an
     openid.check_authenticate message.  After that...
   ]
   S: C2 OK [OPENID ZW1haWw9bGVhckBtYWlsLmV4YW1wbGUsZnVsbG5hbW
             U9RWxpb3QlMjBMZWFy] authenticated.
   [ Here the IMAP server has returned an SREG attribute of
     email=lear@mail.example,fullname=Eliot%20Lear.
     Line break added in this example for clarity. ]
```


5. Security Considerations

This section will address only security considerations associated with the use of OpenID with SASL applications. For considerations relating to OpenID in general, the reader is referred to the OpenID specification and to other literature. Similarly, for general SASL Security Considerations, the reader is referred to that specification.

5.1. 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 either some sort of registration process takes place to register specific OpenIDs, or that only specific trusted OpenID Providers be allowed. Some out of band knowledge may help this process along. For instance, users of a particular domain may utilize a particular OP that enforces a mapping.

5.2. RP redirected by malicious URL to take an improper action

In the initial SASL client response a user or host can transmit a malicious 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 the RP does a port scan of the site. The URL could send the connection to an internal host or even the local host, which the attacker would not normally have access to. The URL could contain a protocol other than http or https, such as file or ftp.

To mitigate this attack, implementations should carefully analyze URLs received, eliminating any that would in some way be privileged. 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.

5.3. Session Swapping (Cross-Site Request Forgery)

There is no defined mechanism in the OpenID protocol to bind the OpenID session to the user's browser. An attacker may forge a cross-site request in the log-in form, which has the user logging into a proper RP as the attacker. The user would not recognize they are logged into the site as the attacker, and so may reveal information at the RP. Cross-site request forgery is a widely exploited vulnerability at web sites. This is only concern in the context SASL in as much as the client is not configured with the Relying Party (e.g., SASL server) in a safe manner.

5.4. 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 track - to some extent - user access to their services and any additional information that OP provides.

5.5. Collusion between RPs

It is possible for RPs to link data that they have collected on you. By using the same identifier to log into every RP, collusion between RPs is possible. In OpenID 2.0, directed identity was introduced. Directed identity allows the OP to transform the identifier the user typed in to another identifier. This way the RP would never see the actual user identifier, but a randomly generated identifier. This is an option the user has to understand and decide to use if the OP is supporting it.

6. IANA Considerations

The IANA is requested to register the following SASL profile:

SASL mechanism profile: OPENID

Security Considerations: See this document

Published Specification: See this document

For further information: Contact the authors of this document.

Owner/Change controller: the IETF

Note: None

7. Acknowledgments

The authors would like to thank Alexey Melenkov, Joe Hildebrand, Mark Crispin, and Klaas Wierenga for their review and contributions.

8. Normative References

[OpenID] OpenID Foundation, "OpenID Authentication 2.0 - Final", December 2007.

[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.

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

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

[SREG1.0] OpenID Foundation, "OpenID Simple Registration Extension version 1.0", June 2006.

[XRI2.0] Reed, D. and D. McAlpin, "Extensible Resource Identifier (XRI) Syntax V2.0", OASIS Standard xri-syntax-V2.0-cs, September 2005.

Appendix A. Changes

This section to be removed prior to publication.

o 00 Initial Revision.

Authors' Addresses

Eliot Lear
Cisco Systems GmbH
Richtistrasse 7
Wallisellen, ZH CH-8304
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