A SASL Mechanism for OAuth
draft-mills-kitten-sasl-oauth-02.txt

Abstract

Simple Authentication and Security Layer (SASL) is a framework for providing authentication and data security services in connection-oriented protocols via replaceable mechanisms. OAuth is a protocol framework for delegated HTTP authentication and thereby provides a method for clients to access a protected resource on behalf of a resource owner.

This document defines the use of HTTP authentication over SASL, and additionally defines authorization and token issuing endpoint discovery. Thereby, it enables schemes defined within the OAuth framework for non-HTTP-based application protocols. A future version of this document will describe the integration into the Generic Security Services Application Program Interface (GSS-API).

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 10, 2011.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.
Table of Contents

1. Introduction . . . . . . . . . . . . . . . . . . . . . . . . .  3
2. Terminology . . . . . . . . . . . . . . . . . . . . . . . . .  5
3. The OAuth SASL Mechanism . . . . . . . . . . . . . . . . . .  6
   3.1. Channel Binding . . . . . . . . . . . . . . . . . . . . .  6
   3.2. Initial Client Response . . . . . . . . . . . . . . . . .  6
      3.2.1. Query String in OAUTH-SSL . . . . . . . . . . . . .  7
   3.3. Server's Response . . . . . . . . . . . . . . . . . . . .  7
      3.3.1. Mapping to SASL Identities . . . . . . . . . . . . .  8
   3.4. Discovery Information . . . . . . . . . . . . . . . . . .  8
   3.5. Use of Signature Type Authorization . . . . . . . . . . .  9
4. Implementation Requirements . . . . . . . . . . . . . . . . . . 11
5. Examples . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
   5.1. Successful Bearer Token Exchange . . . . . . . . . . . . . 12
   5.2. MAC Authentication with Channel Binding . . . . . . . . . 12
   5.3. Failed Exchange . . . . . . . . . . . . . . . . . . . . . 13
   5.4. Failed Channel Binding . . . . . . . . . . . . . . . . . 14
6. Security Considerations . . . . . . . . . . . . . . . . . . . . 15
7. IANA Considerations . . . . . . . . . . . . . . . . . . . . . . 16
   7.1. SASL Registration . . . . . . . . . . . . . . . . . . . . 16
   7.2. Link Type Registration . . . . . . . . . . . . . . . . . 16
      7.2.1. OAuth 2 Authentication Endpoint . . . . . . . . . . . . 16
      7.2.2. OAuth 2 Token Endpoint . . . . . . . . . . . . . . . 17
      7.2.3. OAuth 1.0a Request Initiation Endpoint . . . . . . . . 17
      7.2.4. OAuth 1.0a Authorization Endpoint . . . . . . . . . . 17
      7.2.5. OAuth 1.0a Token Endpoint . . . . . . . . . . . . . 18
8. Appendix A -- Document History . . . . . . . . . . . . . . . . . 19
9. References . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
   9.1. Normative References . . . . . . . . . . . . . . . . . . 20
   9.2. Informative References . . . . . . . . . . . . . . . . . 21
Authors’ Addresses . . . . . . . . . . . . . . . . . . . . . . . . . 22
1. Introduction

OAuth [I-D.ietf-oauth-v2] offers a standard mechanism for delegating authentication typically used for the purpose of control access to resources. The core OAuth specification defines a framework for authentication and token usage in an HTTP-based environment. The HTTP authorization schemes and tokens in this model are defined separately, some are defined within the OAuth 2 framework such as OAuth 2.0 Protocol: Bearer Tokens [I-D.ietf-oauth-v2-bearer], and some are free standing with OAuth 2 framework bindings such as MAC Authentication [I-D.hammer-oauth-v2-mac-token] tokens. This mechanism takes advantage of the OAuth protocol and infrastructure to provide a way to use SASL [RFC4422] for access to resources for non-HTTP-based protocols such as IMAP [RFC3501], which is what this memo uses in the examples.

The general authentication flow is that the application will first obtain an access token from an OAuth token service for the resource. Once the client has obtained an OAuth access token it then connects and authenticated using this SASL mechanism.

Figure 1 shows the relationship between SASL and OAuth graphically. Item (1) denotes the part of the OAuth exchange that remains unchanged from [I-D.ietf-oauth-v2], i.e. where the client obtains and refreshes Access Tokens. This document focuses on item (2) where the Access Token is presented to the resource server over SASL.
Figure 1: Interworking Architecture

Note: The discovery procedure in OAuth is still work in progress. Hence, the discovery components described in this document should be considered incomplete and a tentative proposal. In general, there is a tradeoff between a generic, externally available defined discovery mechanisms (such as Webfinger using host-meta [I-D.hammer-hostmeta]) and configuration information exchanged inband between the protocol endpoints.
2. 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 [RFC2119].

The reader is assumed to be familiar with the terms used in the OAuth 2.0 specification.

In examples, "C:" and "S:" indicate lines sent by the client and server respectively. Line breaks have been inserted for readability.

Note that the IMAP SASL specification requires base64 encoding message, not this memo.
3. The OAuth SASL Mechanism

SASL is used as a generalized authentication method in a variety of protocols. This document defines the "OAUTH" mechanism to allow HTTP Authorization schemes in the OAuth framework to be used within the SASL framework. In this model, a client authenticates to an OAuth-capable authorization server over HTTPS. This server then issues tokens after successfully authenticating the resource owner. Subsequently, the obtained token may be presented in an OAuth-authenticated request to the resource server. This mechanism further provides compatibility with OAuth 1.0a [RFC5849] and the "OAuth" authentication scheme defined there.

3.1. Channel Binding

Channel binding [RFC5056] in this mechanism is defined in order to allow satisfying the security requirements of the authorization schemes used. This document defines the "OAUTH-SSL" mechanism to provide TLS channel binding [RFC5929] to the OAUTH mechanism, and specifically the "tls-unique" type of channel binding.

If the specification for the underlying authorization scheme requires a security layer such as TLS [RFC5246] the server SHOULD only provide that scheme in a mechanism with channel binding enabled.

3.2. Initial Client Response

The client response is formatted as an HTTP [RFC2616] request. The HTTP request is limited in that the path MUST be "/". In the OAUTH mechanism no query string is allowed. The following header lines are defined in the client response:

User (OPTIONAL): Contains the user identifier being authenticated, and is provided to allow correct discovery information to be returned.

Host (REQUIRED): Contains the host name to which the client connected.


The user name is provided by the client to allow the discovery information to be customised for the user, a given server could allow multiple authenticators and it needs to return the correct one. For instance, a large ISP could provide mail service for several domains who manage their own user information. For instance, users at foo-
example.com could be authenticated by an OAuth service at
https://oauth.foo-example.com/, and users at bar-example.com could be
authenticated by https://oauth.bar-example.com, but both could be
served by a hypothetical IMAP server running at a third domain,
imap.example.net.

3.2.1. Query String in OAUTH-SSL

In the OAUTH-SSL mechanism the channel binding information is carried
in the query string. OAUTH-SSL defines following query parameter(s):

  cbdata (REQUIRED): Contains the base64 encoded first TLS Finished
  message sent.

3.3. Server’s Response

The server validates the response per the specification for the
authorization scheme used. If the authorization scheme used includes
signing of the request parameters the client must provide a complete
HTTP style request that satisfies the data requirements for the
scheme in use.

In the OAUTH-SSL mechanism the server must also extract and base64
decode the first TLS Finished message sent from the client out of the
query parameters of the tunneled HTTP request. It then compares that
to the server’s own copy of that message.

The server responds to a successful OAuth authentication by
completing the SASL negotiation. The authentication scheme MUST
carry the user ID to be used as the authorization identity (identity
to act as). The server MUST use that ID as the user being
authorized, that is the user assertion we accept and not other
information such as from the URL or "User:" header.

The server responds to failed authentication by sending discovery
information in an HTTP style response with the HTTP status code set
to 401, and then failing the authentication. If channel binding is
in use and the channel binding fails the server responds with a
minimal HTTP response without discovery information and the HTTP
status code set to 412 to indicate that the channel binding
precondition failed. If the authentication scheme in use does not
include signing the server SHOULD revoke the presented credential and
the client SHOULD discard that credential.
3.3.1. Mapping to SASL Identities

Some OAuth mechanisms can provide both an authorization identity and an authentication identity. An example of this is OAuth 1.0a [RFC5849] where the consumer key (oauth_consumer_key) identifies the entity using to token which equates to the SASL authentication identity, and is authenticated using the shared secret. The authorization identity in the OAuth 1.0a case is carried in the token (per the requirement above), which SHOULD validated independently. The server MAY use a consumer key or other comparable identity in the OAuth authorization scheme as the SASL authentication identity. If an appropriate authentication identity is not available the server MUST use the identity asserted in the token.

3.4. Discovery Information

The server MUST send discovery information in response to a failed authentication exchange or a request with an empty Authorization header. If discovery information is returned it MUST include an authentication endpoint appropriate for the user. If the "User" header is present the discovery information MUST be for that user. Discovery information is provided by the server to the client to allow a client to discover the appropriate OAuth authentication and token endpoints. The client then uses that information to obtain the access token needed for OAuth authentication. The client SHOULD cache and re-use the user specific discovery information for service endpoints.

Discovery information makes use of both the WWW-Authenticate header as defined in HTTP Authentication: Basic and Digest Access Authentication [RFC2617] and Link headers as defined in [RFC5988]. The following elements are defined for discovery information:

WWW-Authenticate A WWW-Authenticate header for each authentication scheme supported by the server. Authentication scheme names are case insensitive. The following [RFC2617] authentication parameters are defined:

realm REQUIRED -- (as defined by RFC2617)

scope OPTIONAL -- A quoted string. This provides the client an OAuth 2 scope known to be valid for the resource.

oauth2-authenticator An [RFC5988] Link header specifying the [I-D.ietf-oauth-v2] authentication endpoint. This link has an OPTIONAL link-extension "scheme", if included this link applies ONLY to the specified scheme.
oauth2-token An [RFC5988] Link header specifying the [I-D.ietf-oauth-v2] token endpoint. This link has an OPTIONAL link-extension "scheme", if included this link applies ONLY to the specified scheme.

oauth-initiate (Optional) An [RFC5988] Link header specifying the Oauth 1.0a [RFC5849] initiation endpoint. The server MUST send this if "OAuth" is included in the supported list of HTTP authentication schemes for the server.

oauth-authorize (Optional) An [RFC5988] Link header specifying the Oauth 1.0a [RFC5849] authentication endpoint. The server MUST send this if "OAuth" is included in the supported list of HTTP authentication schemes for the server.

oauth-token (Optional) An [RFC5988] Link header specifying the Oauth 1.0a [RFC5849] token endpoint. The server MUST send this if "OAuth" is included in the supported list of HTTP authentication schemes for the server. This link type has one link-extension "grant-types" which is a space separated list of the the OAuth 2.0 grant types that can be used at the token endpoint to obtain a token.

Usage of the URLs provided in the discovery information is defined in the relevant specifications. If the server supports multiple authenticators the discovery information returned for unknown users MUST be consistent with the discovery information for known users to prevent user enumeration. The OAuth 2.0 specification [I-D.ietf-oauth-v2] supports multiple types of authentication schemes and the server MUST specify at least one supported authentication scheme in the discovery information. The server MAY support multiple schemes and MAY support schemes not listed in the discovery information.

If the resource server provides a scope the client SHOULD always request scoped tokens from the token endpoint. The client MAY use a scope other than the one provided by the resource server. Scopes other than those advertised by the resource server must be defined by the resource owner and provided in service documentation (which is beyond the scope of this memo).

3.5. Use of Signature Type Authorization

This mechanism supports authorization using signatures, which requires that both client and server construct the string to be signed. OAuth 2 is designed for authentication/authorization to access specific URIs. SASL is designed for user authentication, and has no facility for being more specific. In this mechanism we
require an HTTP style format specifically to support signature type
authentication, but this is extremely limited. The HTTP style
request is limited to a path of "/". This mechanism is in the SASL
model, but is designed so that no changes are needed if there is a
revision of SASL which supports more specific resource authorization,
e.g. IMAP access to a specific folder or FTP access limited to a
specific directory.

Using the example in the MAC specification
[I-D.hammer-oauth-v2-mac-token] as a starting point, on an IMAP
server running on port 143 and given the MAC style authorization
request (with long lines wrapped for readability) below:

GET / HTTP/1.1
Host: server.example.com
User: user@example.com
Authorization: MAC token="h480djs93hd8",timestamp="137131200",
nonce="dj83hs9s",signature="YTVjyNSujYs1WsDurFnvF14JK6o="

The normalized request string would be constructed per the MAC
specification [I-D.hammer-oauth-v2-mac-token]. In this example the
normalized request string with the new line separator character is
represented by "\n" for display purposes only would be:

h480djs93hd8\n137131200\n
\n
GET\n
server.example.com\n143\n/
\n
\n
4. Implementation Requirements

Tokens typically have a restricted lifetime. In addition a previously obtained token MAY be revoked or rendered invalid at any time. The client MAY request a new access token for each connection to a resource server, but it SHOULD cache and re-use access credentials that appear to be valid. Credential lifetime and how that is communicated to the client is defined in the authentication scheme specifications. Clients MAY implement any of the OAuth 2 profiles since they are largely outside the scope of this specification, and the mentioned profiles in this document are just examples.
5. Examples

These example illustrate exchanges between an IMAP client and an IMAP server.

5.1. Successful Bearer Token Exchange

This example shows a successful OAuth 2.0 bearer token exchange with an initial client response. Note that line breaks are inserted for readability.

S: * IMAP4rev1 Server Ready
C: t0 CAPABILITY
S: * CAPABILITY IMAP4rev1 AUTH=OAUTH
S: t0 OK Completed
C: t1 AUTHENTICATE OAUTH R0VUIl8gSFRUUC8xLjENCkhvc3Q6IGltyXAUuZXhbXBsZS5jb20NCKf1dGhvcml6YXRpb246IEJFQVJFUikyY2tCaGJIUmhkbWx6ZEdFdVkyOKRDZz09Ig0KDQo=
S: +
S: t1 OK SASL authentication succeeded

As required by IMAP [RFC3501], the payloads are base64-encoded. The decoded initial client response is:

GET / HTTP/1.1
Host: imap.example.com
Authorization: BEARER "vF9dft4qmTc2Nvb3RlckBhbHRhdm1zdGEuY29tCg=="

The line containing just a "+" and a space is an empty response from the server. This response contains discovery information, and in the success case no discovery information is necessary so the response is empty. Like other messages, and in accordance with the IMAP SASL binding, the empty response is base64-encoded.

5.2. MAC Authentication with Channel Binding

This example shows a channel binding failure. The example sends the same request as above, but in the context of an OAuth-SSL exchange the channel binding information is missing. Note that line breaks are inserted for readability.
As required by IMAP \[RFC3501\], the payloads are base64-encoded. The decoded initial client response is:

```
GET /?cbdata="SG93IGJpZyBpcyBhIFRMUyBmaW5hbCBtZXNzYWdlPwo=" HTTP/1.1
Host: server.example.com
User: user@example.com
Authorization: MAC token="h480djs93hd8",timestamp="137131200",
nonce="dj83hs9s",signature="WW91IG11c3QgYmUgYm9yZWQuIAo="
```

The line containing just a "+" and a space is an empty response from the server. This response contains discovery information, and in the success case no discovery information is necessary so the response is empty. Like other messages, and in accordance with the IMAP SASL binding, the empty response is base64-encoded.

### 5.3. Failed Exchange

This example shows a failed exchange because of the empty Authorization header, which is how a client can query for discovery information. Note that line breaks are inserted for readability.

```
S: * CAPABILITY IMAP4rev1 AUTH=OAUTH SASL-IR IMAP4rev1 Server Ready
S: t0 OK Completed
C: t1 AUTHENTICATE OAUTH R0VUIC8gSFRUUC8xLjENC1VzXI6IHNb290ZXJAYWx0YXZpc3RhLnVbQ0KSG9zdDogaW1hcW5kF1dGh1bnRpb2F0ZTogQ0Nc==
S: + SFRUUC8xLjEgNDaXIFVuyXV0aG9yaXp1KdVLUF1dGlhbnRpb2F0ZToZogQkBUKV5J1J1JxPSJieGEyGx1LmNvbSINCxknbms6IDxobHRlc2NhcmV2xZ2uLm1aG9vbW5kF1dGh1bnRpb2F0ZToZogQkBUKV5J1J1JxPSJieG50dG5uLkZ0LkYhLk5A
S: t1 NO SASL authentication failed
```

The decoded initial client response failed is:
GET / HTTP/1.1
User: alice@example.com
Host: imap.example.com
Authorization:

The decoded server discovery response is:

HTTP/1.1 401 Unauthorized
WWW-Authenticate: BEARER realm="example.com"
Link: <https://login.yahoo.com/oauth> rel="oauth2-authenticator"
Link: <https://login.yahoo.com/oauth> rel="ouath2-token"

5.4. Failed Channel Binding

This example shows a channel binding failure in a discovery request. The channel binding information is empty. Note that line breaks are inserted for readability.

S: * CAPABILITY IMAP4rev1 AUTH=OAUTH SASL-IR IMAP4rev1 Server Ready
S: t0 OK Completed
C: t1 AUTHENTICATE OAUTH R0VUIC8/Y2JkYXRhPSIiIEhUVFAvMS4xDQpVc2VyOi
   BhbGljZUB1eGItcGx1LmNvbQ0KSG9zdDogaW1hc2U6IGluc3RhbmdJZD0K
   SG9zdDogaW1hc2U6IGluc3RhbmdJZD0K
S: + SFRUUC8xLjEgNDEyIFByZWNvbmRpdGluZw==
S: t1 NO SASL authentication failed

The decoded initial client response is:

GET /?cbdata="" HTTP/1.1
User: alice@example.com
Host: imap.example.com
Authorization:

The decoded server response is:

HTTP/1.1 412 Precondition Failed
6. Security Considerations

This mechanism does not provide a security layer, but does provide a
provision for channel binding. The OAuth 2 specification
[I-D.ietf-oauth-v2] allows for a variety of usages, and the security
properties of these profiles vary. The usage of bearer tokens, for
example, provide security features similar to cookies. Applications
using this mechanism SHOULD exercise the same level of care using
this mechanism as they would in using the SASL PLAIN mechanism. In
particular, TLS 1.2 MUST be implemented and its usage is RECOMMENDED
unless tokens expire quickly.

Channel binding in this mechanism has different properties based on
the authentication scheme used. Bearer tokens have the same
properties as cookies, and the bearer token authentication scheme has
no signature or message integrity. Channel binding to TLS with a
bearer token provides only a binding to the TLS layer.
Authentication schemes like MAC tokens have a signature over the
channel binding information. These provide protection against a man
in the middle, and the MAC authorization header is bound to the
channel and only valid in that context.

A significant benefit of OAuth for usage in clients that usually
store passwords is that the password is not stored in the client, a
token is. This means that the password is not exposed, what we risk
is a token that can be more limited or can be easily revoked.

It is possible that SASL will be authenticating a connection and the
life of that connection may outlast the life of the token used to
authenticate it. This is a common problem in application protocols
where connections are long-lived, and not a problem with this
mechanism per se. Servers MAY unilaterally disconnect clients in
accordance with the application protocol.

An OAuth credential is not equivalent to the password or primary
account credential. There are protocols like XMPP that allow actions
like change password. The server SHOULD ensure that actions taken in
the authenticated channel are appropriate to the strength of the
presented credential.

It is possible for an application server running on Evil.example.com
to tell a client to request a token from Good.example.org. A client
following these instructions will pass a token from Good to Evil.
This is by design, since it is possible that Good and Evil are merely
names, not descriptive, and that this is an innocuous activity
between cooperating two servers in different domains. For instance,
a site might operate their authentication service in-house, but
outsource their mail systems to an external entity.
7. IANA Considerations

7.1. SASL Registration

The IANA is requested to register the following SASL profile:

SASL mechanism profile: OAUTH

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

The IANA is requested to register the following SASL profile:

SASL mechanism profile: OAUTH-SSL

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

7.2. Link Type Registration

Pursuant to [RFC5988] The following link type registrations [[will be]] registered by mail to link-relations@ietf.org.

7.2.1. OAuth 2 Authentication Endpoint

- Relation Name: oauth2- authenticator

- Description: An OAuth 2.0 authentication endpoint.

- Reference:

- Notes: This link type indicates an OAuth 2.0 authentication endpoint that can be used for user authentication/authorization for the endpoint providing the link.
7.2.2. OAuth 2 Token Endpoint

- **Relation Name:** oauth2-token
- **Description:** The OAuth token endpoint used to get tokens for access.
- **Reference:**
- **Notes:** The OAuth 2.0 token endpoint to be used for obtaining tokens to access the endpoint providing the link.

- **Application Data:** This link type has one link-extension "grant-types" which is the OAuth 2.0 grant types that can be used at the token endpoint to obtain a token. This is not an exclusive list, it provides a hint to the application of what SHOULD be valid. A token endpoint MAY support additional grant types not advertised by a resource endpoint.

7.2.3. OAuth 1.0a Request Initiation Endpoint

- **Relation Name:** oauth-initiate
- **Description:** The OAuth 1.0a request initiation endpoint used to get tokens for access.
- **Reference:**
- **Notes:** The OAuth 1.0a endpoint used to initiate the sequence, this temporary request is what the user approves to grant access to the resource.

- **Application Data:**

7.2.4. OAuth 1.0a Authorization Endpoint

- **Relation Name:** oauth-authorize
- **Description:** The OAuth 1.0a authorization endpoint used to approve an access request.
- **Reference:**
- **Notes:**
7.2.5. OAuth 1.0a Token Endpoint

- Relation Name: oauth-token
- Description: The OAuth 1.0a token endpoint used to get tokens for access.
- Reference:
- Notes:
- Application Data:
8. Appendix A -- Document History

[[ to be removed by RFC editor before publication as an RFC ]]

-02

- Filling out Channel Binding

- Added text clarifying how to bind to the 2 kinds of SASL identities.

-01

- Bringing this into line with rdraft 12 of the core spec, the bearer token spec, and references the MAC token spec

- Changing discovery over to using the Link header construct from RFC5988.

- Added the seeds of channel binding.

-00

- Initial revision
9. References

9.1. Normative References

[I-D.hammer-oauth-v2-mac-token]

[I-D.ietf-oauth-v2]

[I-D.ietf-oauth-v2-bearer]


9.2. Informative References

[I-D.hammer-hostmeta]
Hammer-Lahav, E., "Web Host Metadata",
draft-hammer-hostmeta-13 (work in progress), June 2010.

Authors’ Addresses

William Mills
Yahoo! Inc.

Phone:
Email: wmills@yahoo-inc.com

Tim Showalter
Yahoo! Inc.

Phone:
Email: timshow@yahoo-inc.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