Kitten Working Group
Internet-Draft
Obsoletes: RFC 2831 (if approved)
(if approved)
Intended status: Informational
Expires: October 24, 2011

A. Melnikov
Isode Limited
April 22, 2011

Moving DIGEST-MD5 to Historic
draft-ietf-kitten-digest-to-historic-04

Abstract

This memo describes problems with the DIGEST-MD5 Simple Authentication and Security Layer (SASL) mechanism as specified in RFC 2831. It marks DIGEST-MD5 as OBSOLETE in the IANA Registry of SASL mechanisms, and moves RFC 2831 to Historic. status.

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 24, 2011.

Copyright Notice

Table of Contents

1. Overview

[RFC2831] defined how HTTP Digest Authentication [RFC2617] can be used as a Simple Authentication and Security Layer (SASL) [RFC4422] mechanism for any protocol that has a SASL profile. It was intended both as an improvement over CRAM-MD5 [RFC2195] and as a convenient way to support a single authentication mechanism for web, email, LDAP, and other protocols. While it can be argued that it was an improvement over CRAM-MD5, many implementors commented that the additional complexity of DIGEST-MD5 made it difficult to implement fully and securely.

Below is an incomplete list of problems with DIGEST-MD5 mechanism as specified in RFC 2831:

1. The mechanism had too many options and modes. Some of them were not well described and were not widely implemented. For example, DIGEST-MD5 allowed the "qop" directive to contain multiple values, but it also allowed for multiple qop directives to be specified. The handling of multiple options was not specified, which resulted in minor interoperability problems. Some implementations amalgamated multiple qop values into one, while others treated multiple qops as an error. Another example is the use of an empty authorization identity. In SASL an empty authorization identity means that the client is willing to authorize as the authentication identity. The document was not clear on whether the authzid must be omitted or can be specified with the empty value to convey this. The requirement for backward compatibility with HTTP Digest meant that the situation was even worse. For example DIGEST-MD5 required all usernames/ passwords which can be entirely represented in ISO-8859-1 charset to be down converted from UTF-8 to ISO-8859-1. Another example is use of quoted strings. Handling of characters that needed escaping was not properly described and the DIGEST-MD5 document had no examples to demonstrate correct behavior.

2. The document used ABNF from RFC 822 [RFC0822], which allows an extra construct and allows for "implied folding whitespace" to be inserted in many places. The difference from ABNF [RFC5234] was confusing for some implementors. As a result, many implementations didn’t accept folding whitespace in many places where it was allowed.

3. The DIGEST-MD5 document uses the concept of a "realm" to define a collection of accounts. A DIGEST-MD5 server can support one or more realms. The DIGEST-MD5 document didn’t provide any guidance on how realms should be named, and, more importantly, how they can be entered in User Interfaces (UIs). As the result many

DIGEST-MD5 clients had confusing UIs, didn't allow users to enter a realm and/or didn't allow users to pick one of the server supported realms.

4. Use of username in the inner hash. The inner hash of DIGEST-MD5 is an MD5 hash of colon separated username, realm and password. Implementations may choose to store inner hashes instead of clear text passwords. While this has some useful properties, such as protection from compromise of authentication databases containing the same username and password on other servers, if a server with the username and password is compromised, however this was rarely done in practice. Firstly, the inner hash is not compatible with widely deployed Unix password databases, and second, changing the username would invalidate the inner hash.

5. Description of DES/3DES [DES] and RC4 security layers are inadequate to produce independently-developed interoperable implementations. In the DES/3DES case this was partly a problem with existing DES APIs.

6. DIGEST-MD5 outer hash (the value of the "response" directive) didn't protect the whole authentication exchange, which made the mechanism vulnerable to "man in the middle" (MITM) attacks, such as modification of the list of supported qops or ciphers.

7. The following features are missing from DIGEST-MD5, which make it insecure or unsuitable for use in protocols:

   A. Lack of channel bindings [RFC5056].

   B. Lack of hash agility (i.e. no easy way to replace the MD5 hash function with another one).

   C. Lack of support for SASLPrep [RFC4013] or any other type of Unicode character normalization of usernames and passwords. The original DIGEST-MD5 document predates SASLPrep and doesn't recommend any Unicode character normalization.

8. The cryptographic primitives in DIGEST-MD5 are not up to today's standards, in particular:

   A. The MD5 hash is sufficiently weak to make a brute force attack on DIGEST-MD5 easy with common hardware [RFC6151].

   B. Using the RC4 algorithm for the security layer without discarding the initial key stream output is prone to attack [RC4].

C. The DES cipher for the security layer is considered insecure due to its small key space [RFC3766].

Note that most of the problems listed above are already present in the HTTP Digest authentication mechanism.

Because DIGEST-MD5 was defined as an extensible mechanism, it would be possible to fix most of the problems listed above. However this would increase implementation complexity of an already complex mechanism even further, so the effort would not be worth the cost. In addition, an implementation of a "fixed" DIGEST-MD5 specification would likely either not interoperate with any existing implementation of RFC 2831, or would be vulnerable to various downgrade attacks.

Note that despite DIGEST-MD5 seeing some deployment on the Internet, this specification recommends obsoleting DIGEST-MD5 because DIGEST-MD5, as implemented, is not a reasonable candidate for further standardization and should be deprecated in favor of one or more new password-based mechanisms currently being designed.

The SCRAM family of SASL mechanisms [RFC5802] has been developed to provide similar features as DIGEST-MD5 but with a better design.

2. Security Considerations

Security issues are discussed through out this document.

3. IANA Considerations

IANA is requested to change the "Intended usage" of the DIGEST-MD5 mechanism registration in the SASL mechanism registry to OBSOLETE. The SASL mechanism registry is specified in [RFC4422] and is currently available at:

http://www.iana.org/assignments/sasl-mechanisms

4. Acknowledgements

The author gratefully acknowledges the feedback provided by Chris Newman, Simon Josefsson, Kurt Zeilenga, Sean Turner and Abhijit Menon-Sen. Various text was copied from other RFCs, in particular from RFC 2831.

5. References

5.1. Normative References

[RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

[RFC2831] Leach, P. and C. Newman, "Using Digest Authentication as a SASL Mechanism", RFC 2831, May 2000.

5.2. Informative References

[DES] National Institute of Standards and Technology, "Data Encryption Standard (DES)", FIPS PUB 46-3, October 1999.

[RC4] Strombergson, J. and S. Josefsson, "Test vectors for the stream cipher RC4", draft-josefsson-rc4-test-vectors-02.txt (work in progress), June 2010.

[RFC0822] Crocker, D., "Standard for the format of ARPA Internet text messages", STD 11, RFC 822, August 1982.

[RFC2195] Klensin, J., Catoe, R., and P. Krumviede, "IMAP/POP AUTHorize Extension for Simple Challenge/Response", RFC 2195, September 1997.

[RFC3766] Orman, H. and P. Hoffman, "Determining Strengths For Public Keys Used For Exchanging Symmetric Keys", BCP 86, RFC 3766, April 2004.

[RFC4013] Zeilenga, K., "SASLprep: Stringprep Profile for User Names and Passwords", RFC 4013, February 2005.

[RFC4422] Melnikov, A. and K. Zeilenga, "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. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC5802] Newman, C., Menon-Sen, A., Melnikov, A., and N. Williams, "Salted Challenge Response Authentication Mechanism (SCRAM) SASL and GSS-API Mechanisms", RFC 5802, July 2010.

[RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, March 2011.

Author's Address

Alexey Melnikov
Isode Limited
5 Castle Business Village
36 Station Road
Hampton, Middlesex TW12 2BX
UK

Email: Alexey.Melnikov@isode.com
URI: http://www.melnikov.ca/

NETWORK WORKING GROUP
Internet-Draft
Intended status: Standards Track
Expires: October 1, 2017

N. Williams
Cryptonector LLC
A. Melnikov
Isode Ltd

March 30, 2017

Namespace Considerations and Registries for GSS-API Extensions
draft-ietf-kitten-gssapi-extensions-iana-11.txt

Abstract

This document describes the ways in which the GSS-API may be extended and directs the creation of an IANA registry for various GSS-API namespaces.

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 1, 2017.

Copyright Notice

Table of Contents

1. Conventions used in this document

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].

2. Introduction

There is a need for private-use and mechanism-specific extensions to the Generic Security Services Application Programming Interface (GSS-API). As such extensions are designed and standardized (or not), both at the IETF and elsewhere, there is a non-trivial risk of namespace pollution and conflicts. To avoid this we set out guidelines for extending the GSS-API and direct the creation of an IANA registry for GSS-API namespaces.

Registrations of individual items and sub-namespaces are allowed. Each sub-namespace may provide different rules for registration, e.g., for mechanism-specific and private-use extensions.

3. Extensions to the GSS-API

Extensions to the GSS-API can be categorized as follows:

- Abstract API extensions

- Implementation-specific
- Mechanism-specific
- Language binding-specific

Extensions to the GSS-API may be purely semantic, without effect on the GSS-API's namespaces. Or they may introduce new functions, constants, types, etc...; these clearly affect the GSS-API namespaces.

Extensions that affect the GSS-API namespaces should be registered with the IANA as described herein.

4. Generic GSS-API Namespaces

The abstract API namespaces for the GSS-API are:

- Type names
- Function names
- Constant names for various types
- Constant values for various types
- Name types (OID, type name and syntaxes)

Additionally we have namespaces associates with the OBJECT IDENTIFIER (OID) type. The IANA already maintains a registry of such OIDs:

- Mechanism OIDs
- Name Type OIDs

5. Language Binding-Specific GSS-API Namespaces

Language binding specific namespaces include, among others:

- Header/interface module names
- Object classes and/or types
- Methods and/or functions
- Constant names
- Constant values

6. Extension-Specific GSS-API Namespaces

Extensions to the GSS-API may create additional namespaces. See Section 8.2.

7. Registration Form

Registrations for GSS-API namespaces SHALL take the following form:

Registration Field	Possible Values	Description
Bindings	'Generic', 'C-bindings', 'Java', 'C#', <programming language name>	Indicates the name of the programming language that this registration involves, or, if 'Generic', that this is an entry for the generic abstract GSS-API (i.e., not specific to any programming language).
Registration type	'Instance', 'Sub-Namespace'	Indicates whether this entry reserves a given symbol name (and possibly, constant value), or whether it reserves an entire sub-namespace (the name is a pattern) or constant value range.
Object Type	<Symbol> defined by the binding language (for example 'Data-Type', 'Function', 'Method', 'Integer', 'String', 'OID', 'Context-Flag', 'Name-Type', 'Macro', 'Header-File-Name', 'Module-Name', 'Class')	Indicates the type of the object whose symbolic name or constant value this entry registers. The possible values of this field depend on the programming language in question, therefore they are not all specified here.
Symbol Name/Prefix	<Symbol name or name pattern>	The name of a symbol or symbol sub-namespace being

		registered. See Section 8.2.1
Binding of	<Name of abstract API element of which this object is a binding>	If the registration is for a specific language binding of the GSS-API, then this names the abstract API element of which it is a binding (OPTIONAL).
Constant Value/Range	<Constant value> or <constant value range>	The value of the constant named by the <Symbol Name/Prefix>. This field is present only for Instance and Sub-namespace registrations of Constant object types.
Description	<Text>	Description of the registration. Multiple instances of this field may result (see Section 8.2.3).
Registration Rules	<Reference> to an IANA registration Policy defined in [RFC5226] (or an RFC that updates it), for instance 'IESG Approval', 'Expert Review', 'First Come First Served', 'Private Use'.	Describes the rules for allocation of items that fall in this sub-namespace, for entries with Registration Type of Sub-namespace (OPTIONAL). For private use sub-namespaces the submitter MUST provide the e-mail address of a responsible contact. If this field is not specified for a sub-namespace, the default registration rules specified in Section 8.2 apply.
Reference	<Reference>	Reference to a document that describes the registration, if any (OPTIONAL). Multiple instances of this field are allowed, with one reference each.
Expert Reviewer	<Name of expert reviewers, possibly	OPTIONAL, see Section 8.2.2. Multiple instances of this

	WG names>	field are allowed, with one expert reviewer per-instance. Leave this field blank when requesting a registration. It will be filled in by the Expert who reviews the registration.
Expert Review Notes	<Notes from the expert review>	Expert reviewers may request that some comments be included with the registration, e.g., regarding security considerations of the registered extension.
Status	'Registered' or 'Obsoleted'	Status of the registration.
Obsoleting Reference	<Reference>	Reference to a document, if any, that obsoletes this registration. Multiple instances of this field are allowed, with one reference each. (OPTIONAL)

The IANA should create a single GSS-API namespace registry, or multiple registries, one for symbolic names and one for constant values, and/or it may create a registry per-programming language, at its convenience.

Entries in these registries should consist of all the fields from their corresponding registration entries.

Entries should be sorted by: programming language, registration type, object type, and symbol name/pattern.

8. IANA Considerations

This document deals with IANA considerations throughout. Specifically it creates a single registry of various kinds of things, though the IANA may instead create multiple registries, each for one of those kinds of things. Of particular interest may be that IANA will now be the registration authority for the GSS-API name type OID space.

8.1. Initial Namespace Registrations

Initial registry content corresponding to the items defined in [RFC2743], [RFC2744], [RFC2853], [RFC1964] and [RFC4121] and others will be supplied during the IANA review portion of the RFC publishing process. [[Note to RFC Editor: Delete the following sentence before publication:]] The KITTEN WG chairs MUST indicate that such content has been reviewed by the WG and that there is WG consensus that the entries are in agreement with those RFCs.

8.1.1. Example registrations

In order to sanity check recommended IANA registration templates, this section registers several entries.

Registration Field	Possible Values
Bindings	C-bindings
Registration type	Instance
Object Type	Function
Symbol Name	gss_init_sec_context
Binding of	GSS_Init_sec_context
Constant Value/Range	N/A
Description	Create a security context by initiator
Registration Rules	N/A
Reference	RFC 2744
Expert Reviewer	Kitten WG
Expert Review Notes	
Status	Registered
Obsoleting Reference	N/A

Registration Field	Possible Values
Bindings	C-bindings
Registration type	Instance
Object Type	Function
Symbol Name	gss_accept_sec_context
Binding of	GSS_Accept_sec_context
Constant Value/Range	N/A
Description	Accept a security context from initiator
Registration Rules	N/A
Reference	RFC 2744
Expert Reviewer	Kitten WG
Expert Review Notes	
Status	Registered
Obsoleting Reference	N/A

Registration Field	Possible Values
Bindings	C-bindings
Registration type	Instance
Object Type	Context-Flag
Symbol Name	GSS_C_DELEG_FLAG
Binding of	deleg_state or deleg_req_flag
Constant Value/Range	1
Description	On output (if set): Delegated credentials are available via the delegated_cred_handle parameter of GSS_Accept_sec_context. On input (if set): With the call to GSS_Init_sec_context, delegate credentials to the acceptor.
Registration Rules	N/A
Reference	RFC 2744
Expert Reviewer	Kitten WG
Expert Review Notes	
Status	Registered
Obsoleting Reference	N/A

8.2. Registration Maintenance Guidelines

Standards-Track RFCs can create new items with any non-conflicting Symbol Name/Prefix value for this registry by virtue of IESG approval to publish as a Standards-Track RFC -- that is, without additional expert review.

Standards-Track RFCs can mark existing entries as obsolete, and can even create conflicting entries if explicitly stated (the IESG, of course, should review conflicts carefully, and may reject them).

IANA shall also consider submissions from individuals, and via Informational and Experimental RFCs, subject to Expert Review. IANA SHALL allow such registrations if a) they are not conflicting, b) provided that the registration is for object types other than Context-Flags, and c) subject to expert review. Guidelines for expert reviews are given below.

8.2.1. Sub-Namespace Symbol Pattern Matching

Sub-namespace registrations must provide a pattern for matching symbols for which the sub-namespace's registration rules apply. The pattern consists of a string with the following special tokens:

- '*', meaning "match any string."
- "%m", meaning "match any mechanism family short-hand name."
- "%i", meaning "match any implementor vanity short-hand name."

For example, "GSS_%m_*" matches "GSS_krb5_foo" since "krb5" is a common short-hand for the Kerberos V GSS-API mechanism [RFC1964]. But "GSS_%m_*" does not match "GSS_foo_bar" unless "foo" is asserted to be a short-hand for some mechanism.

8.2.2. Expert Reviews of Individual Submissions

[[The following paragraph should be deleted from the document before publication, as it will not age well. It should be moved to the shepherding write-up.]]

Expert review selection SHALL be done as follows. If, at the time that the IANA receives an individual submission for registration in this registry, there are any IETF Working Groups chartered to produce GSS-API-related documents, then the IANA SHALL ask the chairs of such WGs to be expert reviewers or to name one. If there are no such WGs at that time, then the IANA SHALL ask past chairs of the KITTEN WG and the author/editor of this RFC to act as expert reviewers or name an alternate.

Expert reviewers of individual registration submissions with Registration Type == Sub-namespace should check that the registration request has a suitable description (which doesn't need to be sufficiently detailed for others to implement) and that the Symbol Name/Prefix is sufficiently descriptive of the purpose of the sub-namespace or reflective of the name of the submitter or associated company.

Expert reviewers of individual registration submissions with

Registration Type == Instance should check that the Symbol Name falls under a sub-namespace controlled by the submitter. Registration of such entries which do not fall under such a sub-namespace may be allowed provided that they correspond to long existing non-standard extensions to the GSS-API and this can be easily checked or demonstrated, otherwise IESG Protocol Action is REQUIRED (see previous section). Also, reviewers should check that any registration of constant values have a detailed description that is suitable for other implementors to reproduce, and that they don't conflict with other usages or are otherwise dangerous in the reviewers estimation.

Expert reviewers should review impact on mechanisms, security and interoperability, and may reject or annotate registrations which can have mechanism impact that requires IESG protocol action. Consider, for example, new versions of GSS_Init_sec_context() and/or GSS_Accept_sec_context which have new input and/or output parameters which imply changes on the wire or in behaviour that may result in interoperability issues. A reviewer could choose to add notes to the registration describing such issues, or the reviewer might conclude that the danger to Internet interoperability is sufficient to warrant rejecting the registration.

8.2.3. Change Control

Registered entries may be marked obsoleted using the same expert review process as for registering new entries. Obsoleted entries are not, however, to be deleted, but merely marked having Obsoleted Status. Note that entries may be created as obsoleted to record the fact that the given symbol(s) have been used before, even though continued use of them is discouraged.

Registered entries may also be updated in two other ways: additional references, obsoleting references, and descriptions may be added.

All changes are subject to expert review, except for changes to registrations in a sub-namespace which are subject to the rules of the relevant sub-namespace. The submitter of a change request need not be the same as the original submitter.

Registrations may be modified by addition, but under no circumstance may any fields be modified except for the Status field or Contact Address, or to correct for transcription errors in filing or processing registration requests.

The IANA SHALL add a field describing the date that a an addition or modification was made, and a description of the change.

9. Security Considerations

General security considerations relating to IANA registration services apply; see [RFC5226].

Also, expert reviewers should look for and may document security related issues with submitters' GSS-API extensions, to the best of the reviewers' ability given the information furnished by the submitter. Reviewers may add comments regarding their limited ability to review a submission for security problems if the submitter is unwilling to provide sufficient documentation.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2743] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, DOI 10.17487/RFC2743, January 2000, <http://www.rfc-editor.org/info/rfc2743>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

10.2. Informative References

[RFC1964] Linn, J., "The Kerberos Version 5 GSS-API Mechanism", RFC 1964, DOI 10.17487/RFC1964, June 1996, <http://www.rfc-editor.org/info/rfc1964>.

[RFC2744] Wray, J., "Generic Security Service API Version 2 : C-bindings", RFC 2744, DOI 10.17487/RFC2744, January 2000, <http://www.rfc-editor.org/info/rfc2744>.

[RFC2853] Kabat, J. and M. Upadhyay, "Generic Security Service API Version 2 : Java Bindings", RFC 2853, DOI 10.17487/RFC2853, June 2000, <http://www.rfc-editor.org/info/rfc2853>.

[RFC4121] Zhu, L., Jaganathan, K., and S. Hartman, "The Kerberos Version 5 Generic Security Service Application Program Interface (GSS-API) Mechanism: Version 2", RFC 4121, DOI 10.17487/RFC4121, July 2005, <http://www.rfc-editor.org/info/rfc4121>.

Authors' Addresses

Nicolas Williams
Cryptonector LLC

Email: nico@cryptonector.com

Alexey Melnikov
Isode Ltd
5 Castle Business Village
36 Station Road
Hampton, Middlesex TW12 2BX
UK

Email: Alexey.Melnikov@isode.com

KITTEN WORKING GROUP
Internet-Draft
Intended status: Standards Track
Expires: December 2, 2012

N. Williams
Cryptonector, LLC
L. Johansson
SUNET
S. Hartman
Painless Security
S. Josefsson
SJD AB

May 31, 2012

GSS-API Naming Extensions
draft-ietf-kitten-gssapi-naming-exts-15

Abstract

The Generic Security Services API (GSS-API) provides a simple naming architecture that supports name-based authorization. This document introduces new APIs that extend the GSS-API naming model to support name attribute transfer between GSS-API peers.

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 December 2, 2012.

Copyright Notice

Table of Contents

1. Conventions used in this document

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] .

2. Introduction

As described in [RFC4768] the GSS-API's naming architecture suffers from certain limitations. This document defines concrete GSS-API extensions.

A number of extensions to the GSS-API [RFC2743] and its C Bindings [RFC2744] are described herein. The goal is to make information modeled as "name attributes" available to applications. Such information MAY for instance be used by applications to make authorization-decisions. For example, Kerberos V authorization data elements, both in their raw forms, as well as mapped to more useful value types, can be made available to GSS-API applications through these interfaces.

The model is that GSS names have attributes. The attributes of a name may be authenticated (e.g., an X509 attribute certificate or signed SAML attribute assertion), or may have been set on a GSS name for the purpose of locally "asserting" the attribute during credential acquisition or security context exchange. Name attributes' values are network representations thereof (e.g., the actual value octets of the contents of an X.509 certificate extension, for example) and are intended to be useful for constructing portable access control facilities. Applications may often require language- or platform-specific data types, rather than network representations of name attributes, so a function is provided to obtain objects of such types associated with names and name attributes.

Future updates of this specification may involve adding an attribute namespace for attributes that only have application-specific semantics. Note that mechanisms will still need to know how to transport such attributes. The IETF may also wish to add functions by which to inquire whether a mechanism(s) understands a given attribute name or namespace, and to list which attributes or attribute namespaces a mechanism understands. Finally, the IETF may want to consider adding a function by which to determine the name of the issuer of a name attribute.

3. Name Attribute Authenticity

An attribute is 'authenticated' if and only if there is a secure association between the attribute (and its values) and the trusted source of the peer credential. Examples of authenticated attributes are (any part of) the signed portion of an X.509 certificate or AD-KDCIssued authorization-data elements in Kerberos V Tickets provided of course that the authenticity of the respective security associations (e.g., signatures) have been verified.

Note that the fact that an attribute is authenticated does not imply anything about the semantics of the attribute nor that the trusted credential source was authorized to assert the attribute. Such interpretations SHOULD be the result of applying local policy to the attribute.

An un-authenticated attribute is called _asserted_ in what follows. This is not to be confused with other uses of the word asserted or assertion such as "SAML attribute assertion", the attributes of which may be authenticated in the sense of this document for instance if the SAML attribute assertion was signed by a key trusted by the peer.

4. Name Attributes/Values as ACL Subjects

To facilitate the development of portable applications that make use of name attributes to construct and evaluate portable ACLs the GSS-API makes name attribute values available in canonical network encodings thereof.

5. Naming Contexts

Several factors influence the context in which a name attribute is interpreted. One is the trust context.

As discussed previously, applications apply local policy to determine whether a particular peer credential issuer is trusted to make a given statement. Different GSS-API mechanisms and deployments have different trust models surrounding attributes they provide about a name.

For example, Kerberos deployments in the enterprise typically trust a KDC to make any statement about principals in a realm. This includes attributes such as group membership.

In contrast, in a federated SAML environment, the identity provider typically exists in a different organization than the acceptor. In this case, the set of group memberships or entitlements that the IDP is permitted to make needs to be filtered by the policy of the

acceptor and federation.

So even an attribute containing the same information such as e-mail address would need to be treated differently by the application in the context of an enterprise deployment from the context of a federation.

Another aspect related to trust is the role of the credential issuer in providing the attribute. Consider Kerberos PKINIT [RFC4556]. In this protocol, a public key and associated certificate are used to authenticate to a Kerberos KDC. Consider how attributes related to a pkinit certificate should be made available in GSS-API authentications based on the Kerberos ticket. In some deployments the certificate may be fully trusted; in including the certificate information in the ticket, the KDC permits the acceptor to trust the information in the certificate just as if the KDC itself had made these statements. In other deployments, the KDC may have authorized a hash of the certificate without evaluating the content of the certificate or generally trusting the issuing certification authority. In this case, if the certificate were included in the issued ticket, the KDC would only be making the statement that the certificate was used in the authentication. This statement would be authenticated, but would not imply that the KDC stated particular attributes of the certificate described the initiator.

Another aspect of context is encoding of the attribute information. An attribute containing an ASCII [ANSI.X3-4.1986] or UTF-8 [RFC3629] version of an e-mail address could not be interpreted the same as a ASN.1 Distinguished Encoding Rules e-mail address in a certificate.

All of these contextual aspects of a name attribute affect whether two attributes can be treated the same by an application and thus whether they should be considered the same name attribute. In the GSS-API naming extensions, attributes that have different contexts MUST have different names so they can be distinguished by applications. As an unfortunate consequence of this requirement, multiple attribute names will exist for the same basic information. That is, there is no single attribute name for the e-mail address of an initiator. Other aspects of how mechanisms describe information about subjects would already make this true. For example, some mechanisms use OIDs to name attributes; others use URIs.

Local implementations or platforms are likely to have sufficient policy and information to know when contexts can be treated as the same. For example the GSS-API implementation may know that a particular certification authority can be trusted in the context of a pkinit authentication. The local implementation may have sufficient policy to know that a particular credential issuer is trusted to make

a given statement. In order to take advantage of this local knowledge within the GSS-API implementation, naming extensions support the concept of local attributes in addition to standard attributes. For example, an implementation might provide a local attribute for e-mail address. The implementation would specify the encoding and representation of this attribute; mechanism-specific standards attributes would be re-encoded if necessary to meet this representation. Only e-mail addresses in contexts that meet the requirements of local policy would be mapped into this local attribute.

Such local attributes inherently expose a tradeoff between interoperability and usability. Using a local attribute in an application requires knowledge of the local implementation. However using a standardized attribute in an application requires more knowledge of policy and more validation logic in the application. Sharing this logic in the local platform provides more consistency across applications as well as reducing implementation costs. Both options are needed.

6. Representation of Attribute Names

Different underlying mechanisms (e.g., SAML or X.509 certificates) provide different representations for the names of their attribute. In X.509 certificates, most objects are named by object identifiers (OIDs). The type of object (certificate extension, name constraint, keyPurposeID, etc) along with the OID is sufficient to identify the attribute. By contrast, according to Section 8.2 and 2.7.3.1 of [OASIS.saml-core-2.0-os], the name of an attribute has two parts. The first is a URI describing the format of the name. The second part, whose form depends on the format URI, is the actual name. In other cases an attribute might represent a certificate that plays some particular role in a GSS-API mechanism; such attributes might have a simple mechanism-defined name.

Attribute names MUST support multiple components. If there are more than one component in an attribute name, the more significant components define the semantics of the less significant components.

Attribute names are represented as OCTET STRING elements in the API described below. These attribute names have syntax and semantics that are understood by the application and by the lower-layer implementations (some of which are described below).

If an attribute name contains a space (ASCII 0x20), the first space separates the most significant or primary component of the name from the remainder. We may refer to the primary component of the

attribute name as the attribute name's "prefix". If there is no space, the primary component is the entire name, otherwise it defines the interpretation of the remainder of the name.s

If the primary component contains an ASCII : (0x3a), then the primary component is a URI. Otherwise, the attribute is a local attribute and the primary component has meaning to the implementation of GSS-API or to the specific configuration of the application. Local attribute names with an at-sign ('@') in them are reserved for future allocation by the IETF.

Since attribute names are split at the first space into prefix and suffix, there is a potential for ambiguity if a mechanism blindly passes through a name attribute whose name it does not understand. In order to prevent such ambiguities the mechanism MUST always prefix raw name attributes with a prefix that reflects the context of the attribute.

Local attribute names under the control of an administrator or a sufficiently trusted part of the platform need not have a prefix to describe context.

7. API

7.1. SET OF OCTET STRING

The construct SET OF OCTET STRING occurs once in RFC 2743 [RFC2743] where it is used to represent a set of status strings in the GSS_Display_status call. The Global Grid Forum has defined SET OF OCTET STRING as a buffer-set type in GFD.024 [GFD.024] which also provides one API for memory management of these structures. The normative reference to GFD.024 [GFD.024] is for the buffer set functions defined in section 2.5 and the associated buffer set C types defined in section 6 (namely gss_buffer_set_desc, gss_buffer_set_t, gss_create_empty_buffer_set, gss_add_buffer_set_member, gss_release_buffer_set). Nothing else from GFD.024 is required to implement this document. In particular, that document specify changes in behaviour existing GSS-API functions in section 3: implementing those changes are not required to implement this document. Any implementation of SET OF OCTET STRING for use by this specification MUST preserve order.

7.2. Const types

The C bindings for the new APIs uses some types from [RFC5587] to avoid issues with the use of "const". The normative reference to [RFC5587] is for the C types specified in Figure 1 of 3.4.6, nothing

else from that document is required to implement this document.

7.3. GSS_Display_name_ext()

Inputs:

- name INTERNAL NAME,
- display_as_name_type OBJECT IDENTIFIER

Outputs:

- major_status INTEGER,
- minor_status INTEGER,
- display_name OCTET STRING -- caller must release with GSS_Release_buffer()

Return major_status codes:

- GSS_S_COMPLETE indicates no error.
- GSS_S_UNAVAILABLE indicates that the given name could not be displayed using the syntax of the given name type.
- GSS_S_FAILURE indicates a general error.

This function displays a given name using the given name syntax, if possible. This operation may require mapping Mechanism Names (MNs) to generic name syntaxes or generic name syntaxes to mechanism-specific name syntaxes; such mappings may not always be feasible and MAY be inexact or lossy, therefore this function may fail.

7.3.1. C-Bindings

The display_name buffer is de-allocated by the caller with gss_release_buffer.

```
OM_uint32 gss_display_name_ext(
  OM_uint32                    *minor_status,
  gss_const_name_t             name,
  gss_const_OID                display_as_name_type,
  gss_buffer_t                 display_name
);
```

7.4. GSS_Inquire_name()

Inputs:

- name INTERNAL NAME

Outputs:

- major_status INTEGER,
- minor_status INTEGER,
- name_is_MN BOOLEAN,
- mn_mech OBJECT IDENTIFIER,
- attrs SET OF OCTET STRING -- the caller is responsible for de-allocating memory using GSS_Release_buffer_set

Return major_status codes:

- GSS_S_COMPLETE indicates no error.
- GSS_S_FAILURE indicates a general error.

This function outputs the set of attributes of a name. It also indicates if a given name is an Mechanism Name (MN) or not and, if it is, what mechanism it's an MN of.

7.4.1. C-Bindings

```
OM_uint32 gss_inquire_name(
  OM_uint32                    *minor_status,
  gss_const_name_t             name,
  int                          *name_is_MN,
  gss_OID                      *MN_mech,
  gss_buffer_set_t             *attrs
);
```

The gss_buffer_set_t is used here as the C representation of SET OF OCTET STRING. This type is used to represent a set of attributes and is a NULL-terminated array of gss_buffer_t. The gss_buffer_set_t type and associated API is defined in GFD.024 [GFD.024]. The "attrs" buffer set is de-allocated by the caller using gss_release_buffer_set().

7.5. GSS_Get_name_attribute()

Inputs:

- name INTERNAL NAME,
- attr OCTET STRING

Outputs:

- major_status INTEGER,
- minor_status INTEGER,
- authenticated BOOLEAN, -- TRUE if and only if authenticated by the trusted peer credential source.
- complete BOOLEAN -- TRUE if and only if this represents a complete set of values for the name.
- values SET OF OCTET STRING -- the caller is responsible for de-allocating memory using GSS_Release_buffer_set.
- display_values SET OF OCTET STRING -- the caller is responsible for de-allocating memory using GSS_Release_buffer_set

Return major_status codes:

- GSS_S_COMPLETE indicates no error.
- GSS_S_UNAVAILABLE indicates that the given attribute OID is not known or set.
- GSS_S_FAILURE indicates a general error.

This function outputs the value(s) associated with a given GSS name object for a given name attribute.

The complete flag denotes that (if TRUE) the set of values represents a complete set of values for this name. The peer being an authoritative source of information for this attribute is a sufficient condition for the complete flag to be set by the peer.

In the federated case when several peers may hold some of the attributes about a name this flag may be highly dangerous and SHOULD NOT be used.

NOTE: This function relies on the GSS-API notion of "SET OF" allowing for order preservation; this has been discussed on the KITTEN WG mailing list and the consensus seems to be that, indeed, that was always the intention. It should be noted however that the order presented does not always reflect an underlying order of the mechanism specific source of the attribute values.

7.5.1. C-Bindings

The C-bindings of GSS_Get_name_attribute() requires one function call per-attribute value, for multi-valued name attributes. This is done by using a single gss_buffer_t for each value and an input/output integer parameter to distinguish initial and subsequent calls and to indicate when all values have been obtained.

The 'more' input/output parameter should point to an integer variable whose value, on first call to gss_get_name_attribute() MUST be -1, and whose value upon function call return will be non-zero to indicate that additional values remain, or zero to indicate that no values remain. The caller should not modify this parameter after the initial call. The status of the complete and authenticated flags MUST NOT change between multiple calls to iterate over values for an attribute.

The output buffers "value" and "display_value" are de-allocated by the caller using gss_release_buffer().

```
OM_uint32 gss_get_name_attribute(
  OM_uint32                    *minor_status,
  gss_const_name_t             name,
  gss_const_buffer_t           attr,
  int                          *authenticated,
  int                          *complete,
  gss_buffer_t                 value,
  gss_buffer_t                 display_value,
  int                          *more
);
```

7.6. GSS_Set_name_attribute()

Inputs:

- name INTERNAL NAME,
- complete BOOLEAN, -- TRUE if and only if this represents a complete set of values for the name.

- attr OCTET STRING,
- values SET OF OCTET STRING

Outputs:

- major_status INTEGER,
- minor_status INTEGER

Return major_status codes:

- GSS_S_COMPLETE indicates no error.
- GSS_S_UNAVAILABLE indicates that the given attribute NAME is not known or could not be set.
- GSS_S_FAILURE indicates a general error.

When the given NAME object is an MN this function MUST fail (with GSS_S_FAILURE) if the mechanism for which the name is an MN does not recognize the attribute name or the namespace it belongs to. This is because name attributes generally have some semantics that mechanisms must understand.

On the other hand, when the given name is not an MN this function MAY succeed even if none of the available mechanisms understand the given attribute, in which subsequent credential acquisition attempts (via GSS_Acquire_cred() or GSS_Add_cred()) with the resulting name MUST fail for mechanisms that do not understand any one or more name attributes set with this function. Applications may wish to use a non-MN, then acquire a credential with that name as the desired name. The acquired credentials will have elements only for the mechanisms that can carry the name attributes set on the name.

Note that this means that all name attributes are locally critical: the mechanism(s) must understand them. The reason for this is that name attributes must necessarily have some meaning that the mechanism must understand, even in the case of application-specific attributes (in which case the mechanism must know to transport the attribute to any peer). However, there is no provision to ensure that peers understand any given name attribute. Individual name attributes may be critical with respect to peers, and the specification of the attribute will have to indicate which of the mechanism’s protocol or the application is expected to enforce criticality.

The complete flag denotes that (if TRUE) the set of values represents

a complete set of values for this name. The peer being an authoritative source of information for this attribute is a sufficient condition for the complete flag to be set by the peer.

In the federated case when several peers may hold some of the attributes about a name this flag may be highly dangerous and SHOULD NOT be used.

NOTE: This function relies on the GSS-API notion of "SET OF" allowing for order preservation; this has been discussed on the KITTEN WG mailing list and the consensus seems to be that, indeed, that was always the intention. It should be noted that underlying mechanisms may not respect the given order.

7.6.1. C-Bindings

The C-bindings of GSS_Set_name_attribute() requires one function call per-attribute value, for multi-valued name attributes -- each call adds one value. To replace an attribute's every value delete the attribute's values first with GSS_Delete_name_attribute().

```
OM_uint32 gss_set_name_attribute(
  OM_uint32                    *minor_status,
  gss_const_name_t             name,
  int                          complete,
  gss_const_buffer_t           attr,
  gss_const_buffer_t           value
);
```

7.7. GSS_Delete_name_attribute()

Inputs:

- name INTERNAL NAME,
- attr OCTET STRING,

Outputs:

- major_status INTEGER,
- minor_status INTEGER

Return major_status codes:

- GSS_S_COMPLETE indicates no error.

- GSS_S_UNAVAILABLE indicates that the given attribute NAME is not known.
- GSS_S_UNAUTHORIZED indicates that a forbidden delete operation was attempted, such as deleting a negative attribute.
- GSS_S_FAILURE indicates a general error.

Deletion of negative authenticated attributes from NAME objects MUST NOT be allowed and must result in a GSS_S_UNAUTHORIZED.

7.7.1. C-Bindings

```
OM_uint32 gss_delete_name_attribute(
  OM_uint32                    *minor_status,
  gss_const_name_t             name,
  gss_const_buffer_t           attr
);
```

7.8. GSS_Export_name_composite()

Inputs:

- name INTERNAL NAME

Outputs:

- major_status INTEGER,
- minor_status INTEGER,
- exp_composite_name OCTET STRING -- the caller is responsible for de-allocating memory using GSS_Release_buffer

Return major_status codes:

- GSS_S_COMPLETE indicates no error.
- GSS_S_FAILURE indicates a general error.

This function outputs a token which can be imported with GSS_Import_name(), using GSS_C_NT_COMPOSITE_EXPORT as the name type and which preserves any name attribute information (including the authenticated/complete flags) associated with the input name (which GSS_Export_name() may well not). The token format is not specified here as this facility is intended for inter-process communication

only; however, all such tokens MUST start with a two-octet token ID, hex 04 02, in network byte order.

The OID for GSS_C_NT_COMPOSITE_EXPORT is <TBD>.

7.8.1. C-Bindings

The "exp_composite_name" buffer is de-allocated by the caller with gss_release_buffer.

```
OM_uint32 gss_export_name_composite(
  OM_uint32                    *minor_status,
  gss_const_name_t             name,
  gss_buffer_t                 exp_composite_name
);
```

8. IANA Considerations

This specification has no actions for IANA.

This document creates a namespace of GSS-API name attributes. Attributes are named by URIs, so no single authority is technically needed for allocation. However future deployment experience may indicate the need for an IANA registry for URIs used to reference names specified by IETF standards. It is expected that this will be a registry of URNs but this document provides no further guidance on this registry.

9. Security Considerations

This document extends the GSS-API naming model to include support for name attributes. The intention is that name attributes are to be used as a basis for (among other things) authorization decisions or personalization for applications relying on GSS-API security contexts.

The security of the application may be critically dependent on the security of the attributes. This document classifies attributes as asserted or authenticated. Asserted (non-authenticated) attributes MUST NOT be used if the attribute has security implications for the application (e.g., authorization decisions) since asserted attributes may easily be controlled by the peer directly.

It is important to understand the meaning of ’authenticated’ in this setting. Authenticated does not imply that any semantic of the attribute is claimed to be true. The only implication is that a

trusted third party has asserted the attribute as opposed to the attribute being asserted by the peer itself. Any additional semantics are always the result of applying policy. For instance in a given deployment the mail attribute of the subject may be authenticated and sourced from an email system where 'authoritive' values are kept. In another situation users may be allowed to modify their mail addresses freely. In both cases the 'mail' attribute may be authenticated by virtue of being included in signed SAML attribute assertions or by other means authenticated by the underlying mechanism.

When the underlying security mechanism does not provide a permanent unique identity (e.g., anonymous kerberos), GSS-API naming extensions may be used to provide a permanent unique identity attribute. This may be a globally unique identifier, a value unique within the namespace of the attribute issuer, or a "directed" identifier that is unique per peer acceptor identity. SAML, to use one example technology, offers a number of built-in constructs for this purpose, such as a <NameID> with a Format of "urn:oasis:names:tc:SAML:2.0:nameid-format:persistent". SAML deployments also typically make use of domain-specific attribute types that can serve as identifiers.

10. References

10.1. Normative References

[GFD.024] Argonne National Laboratory, National Center for Supercomputing Applications, Argonne National Laboratory, and Argonne National Laboratory, "GSS-API Extensions", GFD GFD.024, June 2004.

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

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

[RFC2744] Wray, J., "Generic Security Service API Version 2 : C-bindings", RFC 2744, January 2000.

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

10.2. Informative References

[ANSI.X3-4.1986]

American National Standards Institute, "Coded Character Set - 7-bit American Standard Code for Information Interchange", ANSI X3.4, 1986.

[OASIS.saml-bindings-2.0-os]
Cantor, S., Hirsch, F., Kemp, J., Philpott, R., and E. Maler, "Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS Standard saml-bindings-2.0-os, March 2005.

[OASIS.saml-core-2.0-os]
Cantor, S., Kemp, J., Philpott, R., and E. Maler, "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS Standard saml-core-2.0-os, March 2005.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC4556] Zhu, L. and B. Tung, "Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)", RFC 4556, June 2006.

[RFC4768] Hartman, S., "Desired Enhancements to Generic Security Services Application Program Interface (GSS-API) Version 3 Naming", RFC 4768, December 2006.

Authors' Addresses

Nicolas Williams
Cryptonector, LLC

Email: nico@cryptonector.com

Leif Johansson
Swedish University Network
Thulegatan 11
Stockholm
Sweden

Email: leifj@sunet.se
URI: http://www.sunet.se

Sam Hartman
Painless Security

Phone:
Fax:
Email: hartmans-ietf@mit.edu
URI:

Simon Josefsson
SJD AB
Hagagatan 24
Stockholm 113 47
SE

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 25, 2012

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

February 24, 2012

A SASL & GSS-API Mechanism for OpenID
draft-ietf-kitten-sasl-openid-08

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 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 August 25, 2012.

Copyright Notice

Table of Contents

1. Introduction

OpenID [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.

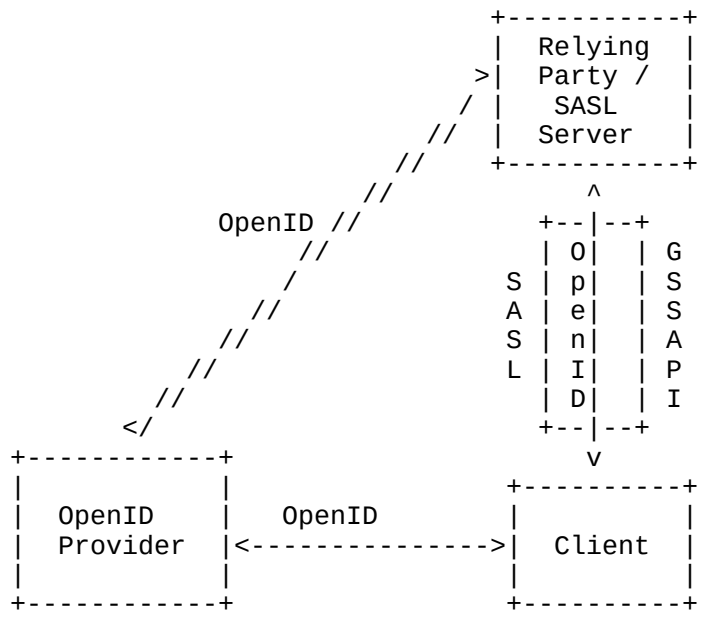
Simple Authentication and Security Layer (SASL) [RFC4422] (SASL) is used by application protocols such IMAP [RFC3501], POP [RFC1939] and XMPP [RFC6120], with the goal of modularizing authentication and security layers, so that newer mechanisms can be added as needed. This memo specifies just such a mechanism.

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 re-use 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.



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 [W3C .REC-html401-19991224] 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 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 the 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 authentiction 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 an 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. 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] 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 an response consisting of "=", to indicate that authentication continues via the normal OpenID flow.

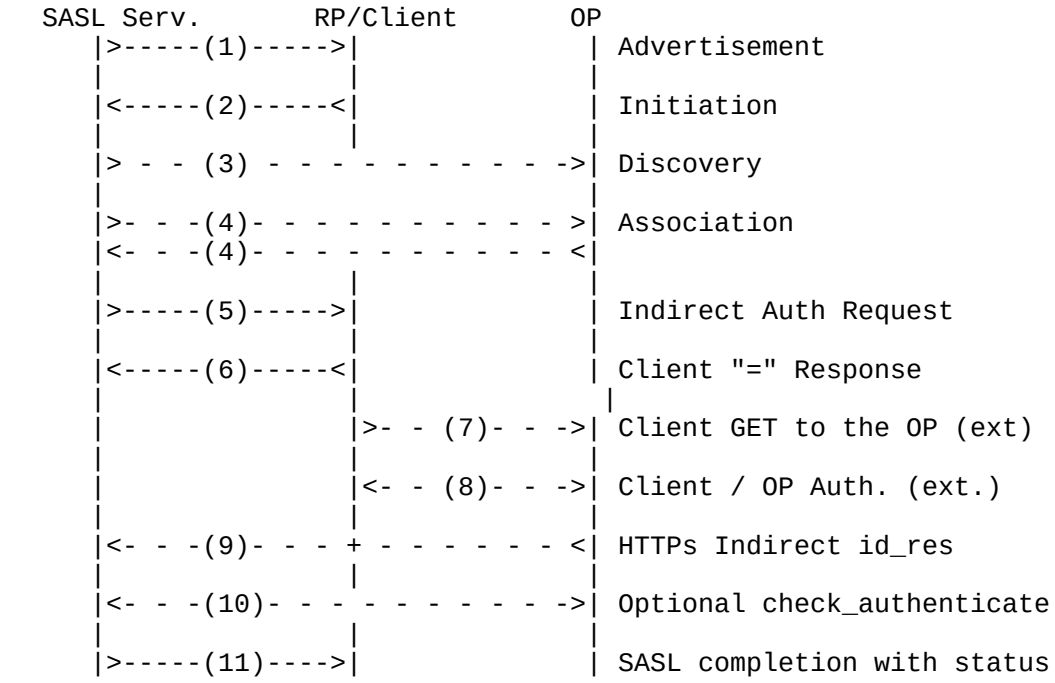
7. At this point the client application MUST construct a URL containing the content received in the previous message from the RP. This URL is transmitted to the OP either by the SASL client application or an appropriate handler, such as a browser.

8. Next the client optionally authenticates to the OP and then approves or disapproves authentication to the Relying Party. 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 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/TLS 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 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 interprocess communication, the relying party MUST encode a nonce or transaction-id in the URIs it transmits through the client for success or failure, either as 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 large 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, fancy style-sheets, etc. Because of this lack of structure, implementations will need to invoke a fairly 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 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 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 "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 from the client to the server 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, the "authentication_request" described below.

The third mechanism message is from client to the server, and 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.

```
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]. XRIs MUST NOT be used. [XRI2.0]

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.

```
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 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 authentiation 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 previsiously in the transaction.

The response by the Relying Party constitutes a SASL mechanism outcome, and 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. 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 concatentation 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

[RFC4422] Section 3.6 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 lock step, 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 and its sub-sections and appropriate references of it not referenced elsewhere in this document are not required for SASL implementors, but this section MUST be observed to implement the GSS-API mechanism discussed 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 is excluded when OpenID is used as a GSS-API mechanism, and b) the RFC2743 section 3.1 initial context token header is prefixed to the client's first authentication message (context token).

The GSS-API mechanism OID for OpenID is OID-TBD (IANA to assign: see IANA considerations).

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 [RFC2743], Section 4.1).

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 [OpenID] section 7.2.

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 [RFC2743], Section 4). The exported name token does, of course, conform to [RFC2743], Section 3.2, 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 one 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 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 readibility.
      ]
      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 who 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_authenticate 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 clarity. ]
      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 and to other literature [1]. 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 pre-arranged 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 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

The IANA is requested to update the SASL Mechanism 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: IETF

Note: None

The IANA is further requested to assign an OID for this GSS mechanism in the SMI numbers registry, with the prefix of iso.org.dod.internet.security.mechanisms (1.3.6.1.5.5) and to reference 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.

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

[RFC2606] Eastlake, D.E. 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.

[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.

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

9.2. Informative References

[RFC1939] Myers, J.G. and M.T. 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>.

Appendix A. Changes

This section to be removed prior to publication.

- 04 - 07 04 - 07 address LC and review comments, including those of Stephen Farrell, Steve Kent, and Brian Carpenter.
- 03 Clarifies messages and ordering, and replace the empty message with a "=" message.
- 02 Address all WGLC comments.
- 01 Specific text around possible improvements for OOB browser control in security considerations. Also talk about transaction id.
- 00 WG -00 draft. Slight wording modifications abou design constraints per Alexey.
- 02 Correct single (significant) error on mechanism name.
- 01 Add nonce discussion, add authorized identity, explain a definition. Add gs2 support.
- 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

Simon Josefsson
SJD AB
Hagagatan 24
Stockholm, 113 47
SE

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 23, 2012

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

February 20, 2012

A SASL and GSS-API Mechanism for SAML
draft-ietf-kitten-sasl-saml-09.txt

Abstract

Security Assertion Markup Language (SAML) 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 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 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 August 23, 2012.

Copyright Notice

Table of Contents

1. Introduction

Security Assertion Markup Language (SAML) 2.0 [OASIS.saml-core-2.0-os] 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.saml-bindings-2.0-os], and interoperability profiles [OASIS.saml-profiles-2.0-os] designed for different use cases.

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], POP [RFC1939] and 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 relying parties. As such, while servers (as relying parties) 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 re-use the Web Browser SSO profile defined in section 4.1 of the SAML profiles 2.0 specification [OASIS.saml-profiles-2.0-os] to the maximum extent and therefore does not establish a separate authentication, integrity and confidentiality mechanism. The mechanism assumes 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 the SAML profiles 2.0 specification [OASIS.saml-profiles-2.0-os] 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 Relying Party (the SASL server) and to the Client, as the two SASL communication end points, but no changes to the SAML Identity Provider are necessary. To accomplish this goal some indirect messaging is tunneled within SASL, and some use of external methods is made.

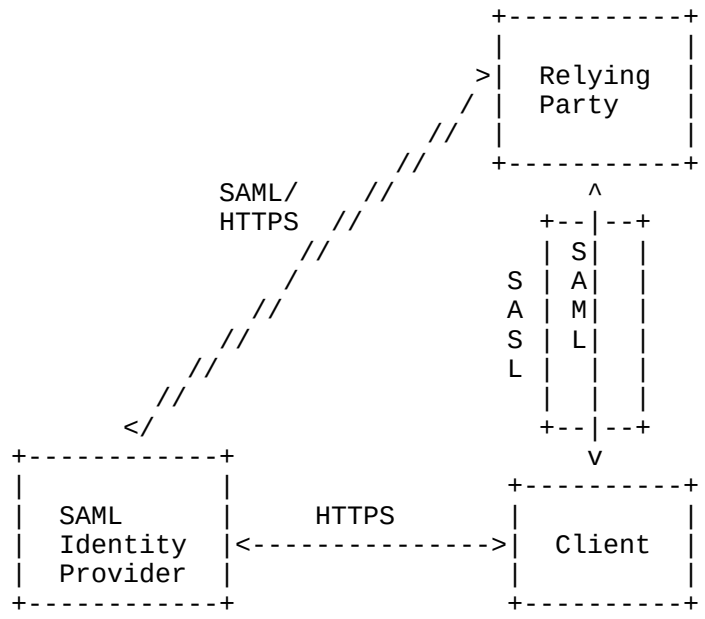


Figure 1: Interworking Architecture

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 specification [OASIS.saml-core-2.0-os].

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 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-html401-19991224]. What follows is a typical flow:

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

2. The Relying Party redirects the browser via an HTTP redirect (as described in Section 10.3 of [RFC2616]) to the Identity Provider (IdP) or an IdP discovery service. When it does so, it includes the following parameters: (1) an authentication request that contains the name of resource being requested, (2) a browser cookie, and (3) a return URL as specified in Section 3.1 of the SAML profiles 2.0 specification [OASIS.saml-profiles-2.0-os].

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

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 Relying Party 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 Relying Party and the client both must change their code to implement this SASL mechanism, the IdP can remain untouched. The Relying Party 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 (Relying Party) 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 an empty response, as 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 is out of scope for SAML and hence for this draft. 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 the SASL server (Relying Party) in the form of an Authentication Statement or failure, using a 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, along with an optional list of attributes

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:

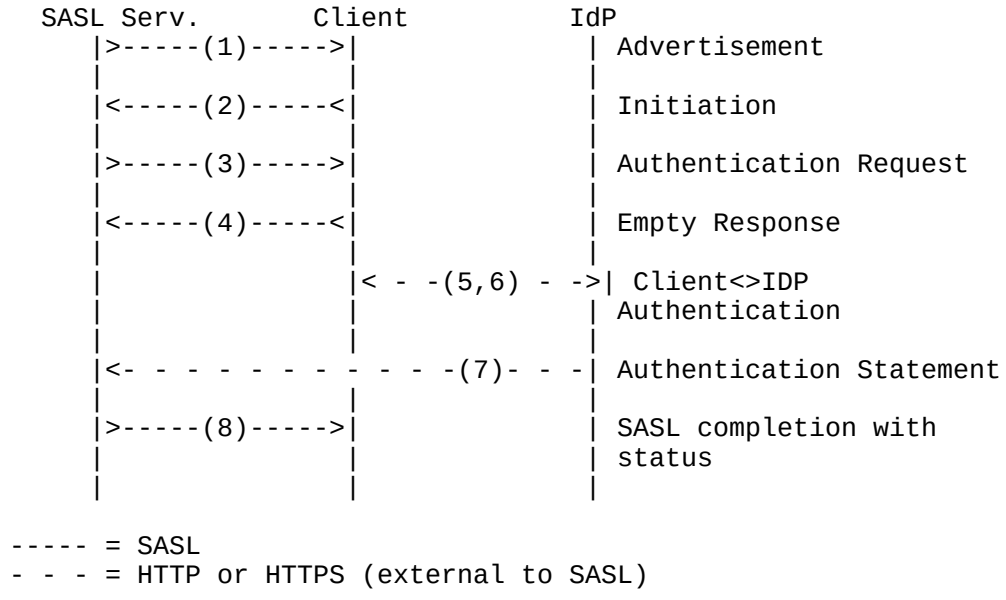


Figure 2: Authentication flow

3. SAML SASL Mechanism Specification

This section specifies the details of the SAML SASL mechanism. See section 5 of [RFC4422] for what is described here.

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 client to the server, and is the fixed message consisting of "=" (i.e., an empty response). 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 authentication identifier (message 2 in Figure 2) and is defined 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).

Domain name is specified in [RFC1035]. 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) in the following way:

```
authentication-request = URI
```

URI is specified in [RFC3986] and is encoded according to Section 3.4 (HTTP Redirect) of the SAML bindings 2.0 specification [OASIS.saml-bindings-2.0-os]. The SAML authentication request is encoded according to Section 3.4 (Authentication Request) of the SAML core 2.0 specification [OASIS.saml-core-2.0-os]. Should the client support Internationalized Resource Identifiers (IRIs) [RFC3987] it MUST first convert the IRI to a URI before transmitting it to the server [RFC5890].

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 which 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 3.1 of SAML profiles 2.0 specification [OASIS.saml-profiles-2.0-os] (message 5 and 6 in Figure 2).

The client handles both user authentication to the IdP and confirmation or rejection of the authentiation 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 Relying Party 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 Relying Party. 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 Relying Party) 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 specify 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 Relying Party takes the role of the GSS-API Acceptor. The SAML Identity Provider 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 is excluded when SAML is used as a GSS-API mechanism, and

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

The GSS-API mechanism OID for SAML is OID-TBD (IANA to assign: see IANA considerations).

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 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]. More precisely, to pass identity validation the client uses the securely negotiated targ_name as the reference identifier and match it to the DNS-ID of the server certificate, and MUST reject the connection if there is a mismatch. For compatibility with deployed certificate hierarchies, the client MAY also perform a comparison with the 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 GSS_Pseudo_random.

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 does, 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 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 according to the definition in Section 4 ofBASE64 [RFC4648] encoded gs2-header and domain:

```
<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:

```
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:

```
 <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 empty response to the challenge encoded as a single =:

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

The following steps between brackets are out of scope for this

document but 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 Identity Provider (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 who sends the AuthN response to the server, containing the subject-identifier as an attribute. 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 (alt): 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 for clarity.

5.2. IMAP

The following 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
CUWNtOTBiMk52YkVKcA0KYm1ScGJtYzlJblZ5YmpwdllYTnBjenB1WVcxbGN6
cDBZenBUUVUxTU9qSXVNRHBpYVc1a2FXNW5jenBJVg0KRlJRTFZCUFUxUWlEU
W9nSUNBZ1FYTnpaWEowYVc5dVEyOXVjM1Z0WlhKVFpYSjJhV05sVlZKTVBRME
tJQw0KQWdJQ0FnSUNBaWFIUjBjSE02THk5dFlXbHNMbVY0WVcxd2JHVXVZMjl
0TDFOQlRVd3ZRWE56WlhKMGFXOQ0KdVEyOXVjM1Z0WlhKVFpYSjJhV05sSWo0
TkNpQThjMkZ0YkRwSmMzTjFaWElnZUcxc2JuTTZjMkZ0YkQwaQ0KZFhKdU9tO
WhjMmx6T201aGJXVnpPblJqT2xOQlRVdzZNaTR3T21GemMyVnlkR2x2YmlJK0
RRb2dJQ0FnSQ0KR2gwZEhCek9pOHZlRzF3Y0M1bGVHRnRjR3hsTG1OdmJRMEt
JRHd2YzJGdGJEcEpjM04xWlhJK0RRb2dQSA0KTmhiV3h3T2s1aGJXVkpSRkJ2
YkdsamVTQjRiV3h1Y3pwellXMXNjRDBpZFhKdU9tOWhjMmx6T201aGJXVg0Ke
k9uUmpPbE5CVFV3Nk1pNHdPbkJ5YjNSdlkyOXNJZzBLSUNBZ0lDQkdiM0p0WV
hROUluVnlianB2WVhOcA0KY3pwdVlXMWxjenAwWXpwVFFVMU1Pakl1TURwdVl
XMWxhV1F0Wm05eWJXRjBPbkJsY25OcGMzUmxiblFpRA0KUW9nSUNBZ0lGTlFU
bUZ0WlZGMVlXeHBabWxsY2owaWVHMXdjQzVsZUdGdGNHeGxMbU52YlNJZ1FXe
HNiMw0KZERjbVZoZEdVOUluUnlkV1VpSUM4K0RRb2dQSE5oYld4d09sSmxjWF
ZsYzNSbFpFRjFkR2h1UTI5dWRHVg0KNGRBMEtJQ0FnSUNCNGJXeHVjenB6WVc
xc2NEMGlkWEp1T205aGMybHpPbTVoYldek9uUmpPbE5CVFV3Ng0KTWk0d09u
QnliM1J2WTI5c0lpQU5DaUFnSUNBZ0lDQWdRMjl0Y0dGeWFYTnZiajBpWlhoa
FkzUWlQZzBLSQ0KQ0E4YzJGdGJEcEJkWFJvYmtOdmJuUmxlSFJEYkdGemMxSm
xaZzBLSUNBZ0lDQWdlRzFzYm5NNmMyRnRiRA0KMGlkWEp1T205aGMybHpPbTV
oYldWek9uUmpPbE5CVFV3Nk1pNHdPbUZ6YzJWeWRHbHZiaUkrRFFvZ0lDQQ0K
Z0lDQjFjbTQ2YjJGemFYTTZibUZ0WlhNNmRHTTZVMEZOVERveUxqQTZZV002W
TJ4aGMzTmxjenBRWVhOeg0KZDI5eVpGQnliM1JsWTNSbFpGUnlZVzV6Y0c5eW
RBMEtJQ0E4TDNOaGJXdzZRWFYwYUc1RGIyNTBaWGgwUQ0KMnhoYzNOU1pXWSt
EUW9nUEM5ellXMXNjRHBTWlhGMVpYTjBaV1JCZFhSb2JrTnZiblJsZUhRK0lB
MEtQQw0KOXpZVzFzY0RwQmRYUm9ibEpsY1hWbGMzUSs=
C:
S: . OK Success (tls protection)
```

The decoded challenge is:

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
Mi4wOnByb3RvY29sIiANCiAgICAgQ29tcGFyaXNvbj0iZXhhY3QiPg0KI
CA8c2FtbDpBdXRobkNvbnRleHRDbGFzc1JlZg0KICAgICAgeG1sbnM6c2FtbD
0idXJuOm9hc2lzOm5hbWVzOnRjOlNBTUw6Mi4wOmFzc2VydGlvbiI+DQogICA
gICB1cm46b2FzaXM6bmFtZXM6dGM6U0FNTDoyLjA6YWM6Y2xhc3NlczpQYXNz
d29yZFByb3RlY3RlZFRyYW5zcG9ydA0KICA8L3NhbWw6QXV0aG5Db250ZXh0Q
2xhc3NSZWY+DQogPC9zYW1scDpSZXF1ZXN0ZWRBdXRobkNvbnRleHQ+IA0KPC
9zYW1scDpBdXRoblJlcXVlc3Q+

Where the decoded SAMLRequest looks like:

```
 <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, the reader is referred to the SAML specification and to other literature. Similarly, for general SASL Security Considerations, the reader is referred to that specification.

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 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 typical part of SAML trust establishment between Relying Parties and IdP.

6.3. User Privacy

The IdP is aware of each Relying Party 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 Relying Parties to link data that they have collected on the users. By using the same identifier to log into every Relying Party, collusion between Relying Parties is possible. In SAML, targeted identity was introduced. Targeted identity allows the IdP to transform the identifier the user typed in to a Relying Party specific opaque identifier. This way the Relying Party would never see the actual user identifier, but a randomly generated identifier.

6.5. GSS-API specific security considerations

Security issues inherent in GSS-API (RFC 2743) and GS2 (RFC 5801) 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 is requested to register 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 is further requested to assign a new entry for this GSS mechanism in the sub-registry for SMI Security for Mechanism Codes, whose prefix is iso.org.dod.internet.security.mechanisms (1.3.6.1.5.5) and to reference this specification in the registry.

8. References

8.1. Normative References

[OASIS.saml-bindings-2.0-os]
Cantor, S., Hirsch, F., Kemp, J., Philpott, R., and E. Maler, "Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS Standard saml-bindings-2.0-os, March 2005.

[OASIS.saml-core-2.0-os]
Cantor, S., Kemp, J., Philpott, R., and E. Maler, "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS Standard saml-core-2.0-os, March 2005.

[OASIS.saml-profiles-2.0-os]
Hughes, J., Cantor, S., Hodges, J., Hirsch, F., Mishra, P., Philpott, R., and E. Maler, "Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS Standard OASIS.saml-profiles-2.0-os, March 2005.

[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. and K. Zeilenga, "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.

[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-html401-19991224] 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>.

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.

[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.

Appendix B. Changes

This section to be removed prior to publication.

- 09 Fixed text per IESG review
- 08 Fixed text per Gen-Art review
- 07 Fixed text per comments Alexey Melnikov
- 06 Fixed text per AD comments
- 05 Fixed references per ID-nits
- 04 Added request for IANA assignment, few text clarifications
- 03 Number of cosmetic changes, fixes per comments Alexey Melnikov
- 02 Changed IdP URI to domain per Joe Hildebrand, fixed some typos
- 00 WG -00 draft. Updates GSS-API section, some fixes per Scott Cantor
- 01 Added authorization identity, added GSS-API specifics, added client supplied IdP
- 00 Initial Revision.

Authors' Addresses

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

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

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

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

Simon Josefsson
SJD AB
Hagagatan 24
Stockholm 113 47
SE

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 26, 2011

S. Josefsson
SJD AB

February 22, 2011

SASL and GSS-API Mechanism for Two Factor Authentication based on a Password and a One-Time Password (OTP): CROTP draft-josefsson-kitten-crotp-00

Abstract

The CROTP mechanism family provide support for two-factor authentication using a static long-term password and a single use changing one-time password (OTP) in the SASL and GSS-API frameworks. The design of CROTP is based on SCRAM described in RFC 5802. CROTP works with several OTP system, including the Open AuTHentication HOTP algorithm described in RFC 4226.

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 August 26, 2011.

Copyright Notice

Table of Contents

1. Introduction

Simple Authentication and Security Layer (SASL) [RFC4422] is a framework that provide user authentication for connection-based protocols. The SCRAM [RFC5802] mechanism family provides username/password based authentication. Several systems for One-Time Password exists, including S/KEY [RFC1760] (revised into OTP [RFC2289]) and OATH HOTP [RFC4226].

The CROTP mechanism extends SCRAM by adding a new mandatory field to transfer a One-Time Password (OTP) and specifying some additional requirements on implementations. CROTP is defined as a SASL mechanism that is wire compatible with a GSS-API mechanism used through the GS2 [RFC5801] framework, which effectively specifies a new GSS-API [RFC2743] mechanism of CROTP.

Because CROTP is specified as SCRAM with some modification, familiarity with SCRAM is required by the reader.

2. Conventions Used in This Document

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].

3. Mechanism Name

The CROTP mechanism name is constructed from the SCRAM mechanism name by replacing "SCRAM" with "CROTP". The length of the complete CROTP mechanism name will thus be the same as the complete "SCRAM" mechanism name. This permit use of any hash function name as per the discussion in section 4 of [RFC5802].

For example, the CROTP mechanism names that correspond to SCRAM-SHA-1 and SCRAM-SHA-1-PLUS are CROTP-SHA-1 and CROTP-SHA-1-PLUS respectively.

For interoperability, all CROTP clients and servers MUST implement the CROTP-SHA-1 authentication mechanism, i.e., an authentication mechanism from the CROTP family that uses the SHA-1 hash function as defined in [RFC3174].

Generally, the usage of PLUS vs non-PLUS names are the same as for SCRAM.

(If it weren’t for the maximum name length of 20 characters imposed

by the SASL framework, we would have used a more illustrative name such as "OTP-SCRAM" rather than "CROTP".)

4. Protocol

CROTP is defined as a set of modifications of the SCRAM protocol. We provide a new field to transfer the OTP from the client to the server. The OTP is sent in unencrypted/unhashed form, to allow for OTP systems where the server is cannot calculate OTPs locally but require the OTP in clear text from the client to be able to validate it. In the wire protocol the "client-final-message-without-proof" field is replaced as follows:

```
;; from RFC 5802
client-final-message-without-proof =
                  channel-binding "," nonce [","
                  extensions]

;; variant used by CROTP
otp                                = "o=" saslname
client-final-message-without-proof =
                  channel-binding "," nonce "," otp
                  ["," extensions]
```

[Design discussion: The OTP had to go either in "client-first-message-bare" or "client-final-message-without-proof" and we chose the latter to align the OTP prompt with where a client (at latest) queries the user for a password.]

The hashing performed by CROTP/SCRAM covers the entire "client-final-message-without-proof" field, thus the OTP value will be bound to the authentication attempt.

The optional error codes are extended with a new error code to indicate OTP related problems, as follows:

```
server-error-value-ext = "replayed-otp" /
                "invalid-otp" /
                value
```

[TODO: Require TLS for OTP protection, or derive keys from the SCRAM negotiation and GSS_Wrap the OTP?]

5. CROTP as a GSS-API Mechanism

CROTP is identical to SCRAM as a GSS-API except that the GSS-API OID

for CROTP-SHA-1 is 1.3.6.1.4.1.11591.4.9.

6. Implementation guidelines

Servers that support both SCRAM and CROTP may use the same password-equivalent for both mechanisms since they are compatible.

Clients that already have a SCRAM credential (i.e., the "ClientKey") can use it as the "ClientKey" in CROTP without modification.

When a server advertises support for both SCRAM and CROTP, the client may modify the password prompt to include an optional OTP field. If the user does not provide an OTP the client can proceed with SCRAM, but if the user provides an OTP the client would select CROTP.

Clients MUST NOT use CROTP with an empty OTP. If the server supports SCRAM, it could be used instead, or authentication could fail.

7. IANA Considerations

IANA has added the following family of SASL mechanisms to the SASL Mechanism registry established by [RFC4422]:

```
Subject: Registration of SASL mechanism family CROTP-
SASL family name (or prefix for the family): CROTP-
Security considerations: This document
Published specification: This document
Person & email address to contact for further information:
Simon Josefsson <simon@josefsson.org>
Intended usage: COMMON
Owner/Change controller:
Simon Josefsson <simon@josefsson.org>
Note: The family names are intended to match the SCRAM names,
  and uses the same (but separate) registration policy.
  Registration of CROTP names should be reviewed for alignment
  with similar SCRAM names.  XXX: registration policy?
```

8. Security Considerations

The CROTP mechanism is based on [RFC5802] and inherits all its security considerations

The OTP is sent in unencrypted/unhashed form from the client to the server, which allows an attacker to read the OTP value and perform a race with the server to validate the OTP.

TBA.

9. Acknowledgements

TBA.

10. References

10.1. Normative References

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

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

[RFC5802] Newman, C., Menon-Sen, A., Melnikov, A., and N. Williams, "Salted Challenge Response Authentication Mechanism (SCRAM) SASL and GSS-API Mechanisms", RFC 5802, July 2010.

10.2. Informative References

[RFC1760] Haller, N., "The S/KEY One-Time Password System", RFC 1760, February 1995.

[RFC2289] Haller, N., Metz, C., Nesser, P., and M. Straw, "A One-Time Password System", RFC 2289, February 1998.

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

[RFC3174] Eastlake, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, September 2001.

[RFC4226] M'Raihi, D., Bellare, M., Hoornaert, F., Naccache, D., and O. Ranen, "HOTP: An HMAC-Based One-Time Password Algorithm", RFC 4226, December 2005.

[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.

Author's Address

Simon Josefsson
SJD AB
Hagagatan 24
Stockholm 113 47
SE

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

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 13, 2011

T. Yu
MIT Kerberos Consortium

July 12, 2010

Desired changes to the GSS-API
draft-yu-kitten-api-wishlist-01

Abstract

Feedback from GSS-API implementors and application developers suggests that the API as it currently exists would benefit from improvements. This memo collects some specific suggestions of KITTEN WG participants.

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 January 13, 2011.

Copyright Notice

Table of Contents

1. Introduction

Experiences of GSS-API implementors and GSS-API application developers, particularly with the C bindings, suggest that the GSS-API would benefit from certain improvements. Some of these suggestions collected from the KITTEN working group include:

1. initialization/new credentials
2. listing/iterating credentials
3. exporting/importing credentials
4. error message reporting
5. asynchronous calls
6. security strength reporting
7. programmer friendliness

This summary is not complete; it is meant as a starting point.

2. Asynchronous calls

The desire for supporting asynchronous calls is a specific case of a generally accepted goal of increasing concurrency. Proponents of this goal typically note that new computers appear to be gaining more processor cores faster than they are gaining computing speed per core. Asynchronous calls (or event-based solutions) are an alternative to the traditional multi-threading model for increasing concurrency.

The existing C bindings say nothing about thread safety for the GSS-API. Implementors have considered various interpretations of thread safety, including using internal mutex locks within a GSS-API implementation to provide thread safety for callers. While the existing C bindings allow for such an approach, the traditional threaded programming model has its drawbacks.

In addition, some GSS-API mechanisms are nearly impossible to implement in a way that prevents gss_init_sec_context and such from blocking on I/O operations, particularly network I/O. An application that attempts to achieve high concurrency must dedicate a thread for each context establishment operation, for example. This can be problematic on platforms where threads are expensive.

Forcing callers to call into an event loop provided by GSS-API is not desirable.

3. Error message reporting

Existing GSS-API facilities for obtaining error information are limited to 32-bit major and minor status codes. This prevents callers from obtaining detailed (perhaps textual) information that may assist in troubleshooting. In addition, the existing GSS-API specifications do not have provisions for gracefully dealing with potentially conflicting minor status codes in multi-mechanism implementations, particularly ones that allow for runtime loading of GSS-API mechanisms.

Concrete approaches for improving GSS-API error reporting appear to be somewhat lacking, apart from the "PGSSAPI" proposal by Nico Williams, which adds semantics to the actual pointer value passed as the minor_context argument. Several working group participants find the "PGSSAPI" approach distasteful.

4. Security strength reporting

There is some interest in adding an interface to report the security strength of the established context, for use with implementations of protocols such as SASL. Some debate has taken place about whether a numeric report of security strength is an appropriate means of communicating this information to an application.

5. Programmer friendliness

In the GSS-API C bindings, the gss_accept_sec_context function takes 11 parameters, and the gss_init_sec_context function takes 13. Many of these parameters accept a default value, and in fact application developers sometimes unnecessarily provide non-default values, which often unintentionally results in reduced functionality.

Some programmers find that needing to explicitly loop over gss_init_sec_context and gss_accept_sec_context (as is currently required by a conforming GSS-API application during context establishment) is cumbersome. It may be beneficial to define a simpler interface for programmers who do not require the additional control afforded by explicitly calling the context establishment functions in a loop.

6. Security Considerations

Addition of an interface to report security strength of a GSS-API context enables applications to make better-informed decisions about security policy.

Author's Address

Tom Yu
MIT Kerberos Consortium
77 Massachusetts Ave
Building W92 Room 145
Cambridge, MA 02139
US

Phone: +1 617 253 1753
Email: tlyu@mit.edu