NFSv4 Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 14, 2011

W. Adamson
NetApp
K. Coffman
CITI, University of Michigan
N. Williams
Oracle

March 13, 2011

NFSv4 Multi-Domain Access
draft-adamson-nfsv4-multi-domain-access-04

Abstract

The Network File System, version 4 (NFSv4) uses a representation of identity that allows the use of users and groups from multiple, distinct administrative domains, and NFSv4 allows the use of security mechanisms that authenticate principals from multiple, distinct administrative domains. This document describes methods by which NFSv4 clients and servers can handle principals, users, groups from multiple administrative domains.

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 September 14, 2011.

Table of Contents

1. Requirements notation

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

The NFS Version 4 [RFC3530] protocol enables the construction of a distributed file system which can join NFSv4.0 or NFSv4.1 [I-D.ietf-nfsv4-minorversion1] servers from multiple administrative domains, each potentially using separate name resolution services and separate security services, into a common multi-domain name space.

NFSv4 deals with two kinds of identities: authentication identities (referred to here as "principals") and authorization identities ("users" and "groups" of users). NFSv4 supports multiple authentication methods, each authenticating an "initiator principal" (typically representing a user) to an "acceptor principals" (always corresponding to the server). NFSv4 does not prescribe how to represent authorization identities on file systems. All file access decisions constitute "authorization" and are made by servers using information about client principals (such as username, group memberships, and so on) and file metadata related to authorization, such as a file's access control list (ACL).

Authentication in NFSv4 occurs at the the RPCSEC_GSS [RFC2203] layer where GSS-API mechanisms [RFC2743] are used to authenticate users on NFSv4 clients to NFSv4 servers, and to provide security services such as confidentiality and integrity protection for the protocol's messages. The NFSv4 protocol specifies no particular representation for authentication identities as these are entirely GSS-API mechanism-specific.

Authorization for file object access is done at the NFSv4 protocol layer (i.e., above the RPCSEC_GSS layer), on the server side, based on an authenticated client principal's authorization context and the authorization meta-data of the file system objects that the client wishes to access. File authorization meta-data is set and retrieved in the NFSv4 RPC [RFC1831] layer, specifically via the object's owner, owner_group and acl, dacl and sacl attributes (the last three being ACLs). ACLs are lists of ACL entries (ACEs). Each ACE has a "who" field identifying a subject to whom some permission is granted or denied. The owner and owner_group attributes and the who ACE field, all reference users and groups. On the wire, the protocol represents users and groups as strings of characters with this form: name@domain, where <name> is a user or group name, and <domain> is a the name of an administrative domain, more specifically a DNS [RFC1034] domainname.

NFSv4 server implementations usually do not, and really ought not, store authorization identities on disk in the same form as is used on the wire. The reason is that users' and groups' names change all too often, while searching for and updating file authorization meta-data

after a user/group name change is not trivial, particularly in a global namespace spanning multiple administrative domains.

NFSv4 servers therefore must perform two kinds of mappings:

1. Between the authentication identity and the authorization context (a principal's user ID, group memberships, etcetera)

2. Between the on-the-wire authorization identity representation and the on-disk authorization identity representation.

Many aspects of these mappings are entirely implementation-specific, but some require name resolution services, and in order to interoperate servers must use such services in compatible ways. Many implementations are limited to being able to represent users and groups from a single domain.

In this document we address both of those kind of mappings, describing possible implementation strategies, and specifying a name service for interoperation in a global namespace [I-D.ietf-nfsv4-federated-fs-reqts].

3. Background

NFSv4 uses a syntax of the form "name@domain" to represent, on the wire, the NFSv4 ACL name for users and groups. This design provides a level of indirection that allows a client and server with different internal representations of authorization identity to interoperate even when referring to authorization identities from different administrative domains.

Multi-domain capable sites need to meet certain requirements in order to ensure that clients and servers can map name@domain to internal representations reliably:

- The name portion of name@domain MUST be unique within the specified DNS domain.

- Every local representation of a user and of a group MUST have a canonical name@domain, and it must be possible to return the canonical name@domain for any identity stored on disk, at least when required infrastructure servers (such as name services) are online.

As described in [I-D.ietf-nfsv4-minorversion1] section 2.2.1.1 "RPC Security Flavors":

> NFSv4.1 clients and servers MUST implement RPCSEC_GSS. (This requirement to implement is not a requirement to use.) Other flavors, such as AUTH_NONE, and AUTH_SYS, MAY be implemented as well.

The AUTH_NONE security flavor can be useful to the multi-domain NFSv4 or federated name space to grant universal access to public data without any credentials.

The AUTH_SYS security flavor uses a host-based authentication model where the [weakly-authenticated] client asserts the user's authorization identities using small integers as user and group identity representations. Because of the small integer authorization ID representation, AUTH_SYS can only be used in a name space where all clients and servers share a uidNumber and gidNumber translation service. A shared translation service is required because uidNumbers and gidNumbers are passed in the RPC credential; there is no negotiation of namespace in AUTH_SYS. Collisions can occur if multiple translation services are used. These and other issues are addressed in [I-D.williams-rpcsecgssv3] which describes a new version of RPCSEC_GSS that includes a modernized replacement for AUTH_SYS.

4. Terminology

Identity: a way to refer to a user or group.

Principal: an entity that is authenticated by RPCSEC_GSS (usually, but not always, a user; rarely, if ever, a group; sometimes a host).

Authorization context: the set of user and group IDs, privileges, labels, and other items relevant to authorization, corresponding to a subject (user or principal).

Domain-local ID: Most installations assign numeric, local identifiers to users and groups, using a namespace local to their domain. We call this a domain-local ID.

Local representation of identity: an item such as a POSIX user IDentifier (UID) or group ID (GID), or a Windows Security IDentifier (SID), or other such representation of a user or a group of users. These can be local to a single host.

Global representation of identity: a tuple consisting of a domain identifier (possibly the domain's name itself) and a domain-local user/group ID. We do not propose a standard global representation of identity, but the concept is useful. [NEEDSWORK: we refer to the global representation form in the RPCSEC_GSS PAC]

Group: a security entity representing zero, one or more users and, possibly, other groups. Can appear as the subject of an ACE.

Domain: a set of users, groups and computers administered by a single entity, and identified by a DNS domain name.

POSIX IDs: small non-negative integer (typically 0..2^31 or 0..2^32) identifiers. The namespace of user IDs (UIDs) is distinct from the namespace of group IDs (GIDs).

Windows SIDs: an identifier of security entities, including users and groups. The form of a SID is: S-1-<authority>-<RID_0>-<RID_1> -<RID_n> By convention some authority numbers denote security entities, identified by RID_n, local to a domain identified by RIDs 0..n-1. Domain RIDs are usually generated randomly within a "forest" of domains.

Name resolution: mapping from {domain, name} to {domain, ID}, and vice-versa via lookups. Can be applied to local or remote domains.

ID mapping: {remote domain, remote domain-local ID} to {local representation of ID} mappings.

5. Local Representations of Global Identity

Multi-domain support starts at the fileserver where local ID forms need to be able to represent global identities from both local and remote domains. Local representation of global identity also applies to clients, particularly clients with local filesystems. There's a range of local solutions to this multi-domain ID representation problem. In this section we describe several approaches to representing a <name>@<local or remote domain> on disk. None of these approaches are REQUIRED; all are INFORMATIVE. However, conventions relating to the use of name services are NORMATIVE.

5.1. Storing Name@Domainname

One simple approach to the multiple domain problem is to store the name@domain on disk.

This approach imposes a severe constraint on the administrators of these domains: user and group names must never be reused, as there is also no realistic way to keep the name@domain on disk representation up to date with user, group or domain renames and removals. Consider a remote domain's NFSv4 servers where real-time employee join/leave data may be (typically is!) considered privileged, and remote servers may not be sufficiently privileged to access it [NEEDSWORK].

5.2. Storing Remote-ID@Domainname

Most installations assign numeric, local identifiers to users and groups, using a namespace local to their domain. We call this a domain-local ID. We can then construct a global identity form consisting of a domain name and a domain-local user/group ID.

The user or group renaming issue can be addressed by using a global identity form where domain-local IDs are required. I.e., use name resolution to lookup name@domain to find the ID local to the specified domain, and join the ID with the specified domain name. This function still has a renaming problem with respect to DNS domain renames, but that is a more realistically manageable problem than the user/group renaming problem.

5.2.1. Storing Remote-ID@Domain-ID

The DNS domain renaming issue in the previous section can be addressed by assigning and publishing a unique ID to each DNS domain. I.e., use name resolution to lookup name@domain to find some ID local to the domain, lookup the domain ID and store <remote-ID,domain-ID>. The Windows Security Identifier (SID) is an example of this form.

5.3. ID Mapping

Many file systems exported by NFS only store 32-bit user and group IDs which limit their ability to utilize the on disk representation described in Section 5.2. Such systems may need to use an additional service to map between <remote user ID, local user IDs> and <remote group IDs, local group IDs>. We call this an "ID mapping service".

The use of an ID mapping service is not strictly necessary if the system operates on IDs large enough and in a known format such that <user/group ID, domain ID> can be parsed and encoded into a native ID. However, a large class of operating systems, those which are Unix or Unix-like operating systems, such as Solaris and Linux, use 32-bit UIDs and GIDs in many interfaces and therefore need mapping for backwards compatibility reasons.

One example of such a service is to keep a local or distributed database for dynamically assigning a local 32 bit ID to every <ID>@<domain-ID>, or one could do that only for remote domains, reserving only a small part of the local 32-bit ID namespace for remote domains' users/groups.

The remote ID and remote domain are then used as inputs to a name resolution service which contacts the remote domain name service to resolve the remote name.

5.4. Use of Name Services

File systems often use a distributed directory service for resolving domain local 32 bit IDs to users and groups. The Network Information Service [NIS] and the Lightweight Directory Access Protocol [RFC4511] are the two broadly deployed distributed directory service protocols used for this purpose. LDAP is used instead of NIS in environments where scalabilty, security and/or extensibility are desired. Section 8.2 expands the LDAP protocol to include mappings between principals and local user and group IDs.

Support for LDAP [RFC4511] with the RFC2307 schema [RFC2307] is REQUIRED.

5.4.1. Using LDAP with RFC2307 Schema

Name resolution consists of searches with scope 'sub', a base DN corresponding to a domain (more on this below) and a filter of either of these forms, with matching on objectClass being optional:

- (objectClass=posixAccount)(uid="<username>)
- (objectClass=posixGroup)(cn="<groupname>)
- (objectClass=posixAccount)(uidNumber="<UID>)
- (objectClass=posixGroup)(gidNumber="<GID>)

The base DN SHOULD be formatted from a domain's DNS domainname as follows. First format the domainname as a string, then strip the trailing dot ('.'), if any, then replace all dots ('.') with ",DC=", then prepend "DC=" to the resulting string. For example, foo.bar.example becomes "DC=foo,DC=bar,DC=example". This convention is REQUIRED to be implemented. Domains with base DNs that do not match this convention MAY be used, but their domainname-to-base-DN mappings must be published where NFSv4 clients and servers may find them; we provide no conventions for publishing such mappings. We RECOMMEND that LDAP referrals be used to publish such mappings (e.g., the client does an LDAP search using "DC=foo,DC=example" as the base DN and gets a referral that includes the correct non-standard base DN for "foo.example").

Client and server implementations MUST support the use of LDAP referrals to find LDAP servers authoritative for any given base DN.

For example, to resolve a user named joe@foo.bar.example to a remote ID a system would do an LDAP search with DC=foo,DC=bar,DC=example as the base DN, scope='sub' and with a filter of (objectClass=posixAccount)(uid="<username>) looking for the uidNumber attribute.

5.4.2. Using Active Directory LDAP

[NEEDSWORK: Add text describing searches by which to resolve name@domain to SIDs and vice versa.]

5.4.3. Mapping Domain Names to Domain IDs

[NEEDSWORK: Add text on mapping domainnames to domain IDs. Note that Windows SID does this.]

We need to have a common way to map Domain Names to Domain IDs to enable mult-domain numeric IDs as described in Section 5.2.1. Currently we have two suggestions:

1. Just use SIDs, first asking MSFT to allocate a suitable authority for non-Windows domain SIDs.

2. - Store 96-bit numeric IDs which means we:

   * cast those to domain SIDs later

   * define a non-SID large ID format. This is a fine fallback should MSFT be unwilling to assign authority numbers for this purpose

6. Resolving Cross-Domain Authorization Information

In order to authorize client principal access to files, the NFS server must map the RPCSEC_GSS client principal name or the underlying GSS-API security context to authorization information including a local ID, a set of local group IDs and other local user privileges meaningful to the file system being exported.

In the cross-domain case where a client principal is seeking access to files on a server in a different NFSv4 domain, the NFS server needs to obtain, in a secure manner, the authorization information from an authoritative source: e.g. a directory service in the client principals NFS domain.

There are several methods the cross-domain authoritative authorization information can be obtained:

1. A mechanism specific GSS-API authorization payload containing credential authorization data such as a "privilege attribute certificate" or PAC.

2. An NFS server local domain directory query when there is a security agreement between the two cross-domain directory services plus regular update data feeds so that the NFS server local domain directory service is authoritative for the client prinicpal domain.

3. A direct query from the NFS server to the client principal authoritative directory service.

The authorization data information SHOULD be obtained via the GSS-API name attribute interface [I-D.ietf-kitten-gssapi-naming-exts] either via a single attribute for the credential authorization data or via discrete GSS-API name attributes corresponding to the authorization data elements described in Section 6.1. Details for those attributes are TBD.

Note that the retrieval of attribute values used by the GSS-API name attribute interface implementation could utilize any of the above mentioned methods of obtaining the authorization information.

If the named attribute interface is not available, or the attributes are not available, other means of determining a principal’s authorization data SHOULD be used, such as those described in Section 6.2 and Section 6.3.

6.1. Credential Authorization Data

Here we list in more detail the authorization information that an NFSv4 server needs in order to make a file access decision. The credential authorization data contains the user and group IDs corresponding to the client principal, in global representation of identity form. Note that the server may need to map the global IDs to local IDs as described in Section 5.3.

The ability to map IDs to the name@domain form is required for the NFSv4 server to be able to respond to file authorization meta-data (ACL) set and retrieve requests.

Credential authorization data consists of:

- UserID: This field contains the principal's global ID and/or local ID mapping thereof, and the name@domain form thereof.
- PrimaryGroupID: This field contains the global ID and/or local ID mapping thereof for the principal's primary group, and the name@domain form thereof.
- Groups: This field contains an array of group IDs for the groups that the user is a member of, in global ID form and/or local ID mappings thereof, as well as in name@domain forms.
- Optional field(s) for privileges and authorizations granted to the principal, if any.
- Optional field(s) for other privilege information such as the multi-level security label range/set of the principal.
- Optional implementation-specific items relevant to authorization.

6.2. Using Credential Authorization Data

Authorization context information can sometimes be obtained from the credentials authenticating a principal; the GSS-API represents such information as attributes of the initiator prinicpal name. For example: Kerberos 5 [RFC4120] has a method for conveying "authorization data", both client-asserted as well as KDC-authenticated authorization data, and one KDC implementation uses this feature to convey a "privilege attribute certificate" (PAC) listing the principal's user and group "security identifiers" (SIDs). Another example is the Kerberos General PAC [I-D.sorce-krbwg-general-pac] which lists the principal's user and group "universal user identifiers" (UUIDs) as well as their string representations and DNS domains. PKIX [RFC5280] certificates allow

for extensions that could be used similarly.

6.2.1. Using a PAC

The Windows operating system uses an authorization context called a "PAC" [PAC], which contains a user Security IDentifier (SID) and a list of group SIDs. Some Kerberos Key Distribution Centers (KDCs), notably Windows KDCs, issue Kerberos Tickets with PACs as Kerberos authorization data.

Some KDCs (will) issue Kerberos Tickets with the General PAC [I-D.sorce-krbwg-general-pac] as authorization data. The General PAC authorization data MUST be authenticated in the sense that its contents must come from an authenticated, trusted source, such as a directory server or the issuer of the client principal's credential.

When a client principal is authenticated using such a ticket, the server SHOULD extract the PAC from the client's ticket and map, if need be, the SIDs or UUIDs in the PAC to local ID representations.

The authorization context information in a PAC can be considered a single, authenticated, discrete GSS-API name attribute, in which case the server must parse it into its individual elements.

6.3. Using Directory Services

If suitable and sufficient authenticated GSS-API name attributes for the client principal are not available, then the server may try to map the client principal name to a local notion of user account, and then lookup that user account's authorization context information through authenticated name service lookups.

6.3.1. Mapping Principal Names to Username

One simple method for Kerberos principal-to-username mapping is to first apply an algorithmic or table-based Kerberos client principal realm name to domain name mapping, then a client principal name to username mapping. Finally, the server can look up the user's authorization context using the user's domain's name services.

A trivial Kerberos realm-name-to-domainname mapping consists of using the realm name as the domainname. [NEEDSWORK: Add notes about internationalization.] Servers SHOULD implement this mapping as an option, possibly as a default option.

A trivial Kerberos principal name to username mapping for 1-component principal names is to use the principal name, unmodified, as the username. Servers SHOULD implement this mapping as an option,

possibly as a default option.

6.3.2. Using a Name Service to Map Principal Names to User Accounts

Name services such as the Solaris gsscred database where the local identity is looked up in a database keyed by the GSS exported name token, or LDAP with the extension described in Section 8.2, can be used to map principal names to user accounts.

7. Multi Domain User Group Membership Determination

User group membership is easy to determine for users in a system's local domain: the operating system will already know how to do that. For users in remote domains, the authentication service may provide group membership information. If not, we need methods for group memebership determination.

[NEEDSWORK:

1. provide a[n obviously limited] mode of operation that depends only on RFC2307 and therefore does not support group nesting;

2. provide a more full-fledged mode of operation that depends RFC2307bis;

3. provide a more full-fledged mode of operation that depends AD's schema.]

User group membership in remote domain's groups, and/or for remote users, may be determined using LDAP with the RFC2307 schema. The RFC2307 schema does not define the values of the 'memberUid' attribute, but in practice it seems that those are expected to be the names of users as found in the 'uid' attribute of 'posixAccount' entries. There is work in progress to update RFC2307 [I-D.howard-rfc2307bis] to allow the use of DNs in the member attribute. Group nesting is also enabled.

Assuming the ability to store DNs in the member attribute, then, group membership determination can be done as follows. Given a user ID whose DN has been determined:

1. Search the user's domain for groups that the user is a member of in the user's home domain. Since we cannot assume a domain is authoritative for another domains group membership, filter out groups that are not local to the user's home domain.

2. Search the server's domain for groups that the user is a member of in the server's home domain.

3. Search the server's domain for groups that the user's group memberships determined in steps 1 and 2 are members of.

4. Continue searching for nested group memberships given the list of groups from steps 2 and 3 while being careful to detect or prevent loops.

However, the above procedure has the same user/group name renaming

issue. By skipping step 2 we can get down to just a group renaming issue. To fully address the rename issue we need either a new attribute or value type for memberUid, storing user/group IDs in some global ID representation.

[NEEDSWORK: Add text defining such a new attribute/value type.]

8. LDAP and Multi-Domain NFSv4

Each of the three methods of retrieving cross-domain authorization information described in section Section 6 can require a directory service query. Cross-domain querys are not only inefficient, but also implies knowledge of multiple systems where two different domains rely on completely different infrastructures for user information.

[NEEDSWORK: Describe why LDAP is REQUIRED]

8.1. LDAP Service Discovery

[NEEDSWORK: this is just an idea place holder.]

Two potential methods:

1. Use local methods (configuration, DNS SRV RR lookups, ...) to discover local domain's servers, then depend on LDAP referrals for discovering all other domains servers.

2. Use DNS SRV RRs much the way AD does

NICO: I would prefer that we have one REQUIRED to implement service discovery mechanism as follows:

- specify local DS discovery using DNS SRV RR lookups much like AD does (i.e., have a label to indicate the purpose of the LDAP service, not just _ldap). Make this general enough that clients could discover DSes of remote domains on their own.

- use LDAP referrals (and DNS resolution of the host parts of the referrals) to discover DSes of other domains.

The main benefit of this mechanism is that we can leave the work of finding topologically-close caches and/or authoritative servers to the clients' local DSes, thus avoiding the need to deal directly with topology in our spec.

8.2. LDAP Attribute for Principal Name to Local ID Translation

The gSSPrincipal objectclass allows for the use of the gSSAuthName attribute described in the following section.

```
objectclass ( 1.3.6.1.4.1.250.10.7
NAME 'gSSPrincipal'
DESC 'GSS Principal Name'
SUP  posixAccount
```

```
                        MAY ( gSSAuthName ) )
```

The gSSAuthName attribute provides a method for the translations between a posixAccount and (multiple) GSS-API security principals, used as described in Section 6.3.1.

The gSSAuthName attribute stores a user's GSS-API principal name in exported name token form (see [RFC2743]).

```
                        attributetype ( 1.3.6.1.4.1.250.10.6
                        NAME ( 'gSSAuthName')
                        DESC 'GSS-API exported principal name
                        exported token'
                        EQUALITY bitStringMatch
                        SYNTAX 1.3.6.1.4.1.1466.115.121.1.6)
```

8.3. Name Resolution and LDAP Caching

As noted in Section 5.2, most local representations require a name service to perform ID to name translations. Implementations are REQUIRED to support the use of LDAP as a name service, relying on LDAP referrals for federated namespace construction.

Note that in a topographically widely separated set of domains the need to do name service lookups in various domains' name services may prove brittle, resulting in non-deterministic server behavior (e.g., sometimes a user can access share, sometimes they cannot; sometimes they appear to be members of some group, sometimes they do not). To avoid this, site administrators may wish to maintain local caches of remote domains' name services such that LDAP searches for users/ groups in remote domains can be satisfied locally for some set of key attributes (such as naming and ID attributes), with referrals used in all other cases.

Domains in a federated namespace may provide each other with LDAP LDIF delta feeds by which to maintain cached LDAP contents up to date.The LDAP DN hierarchy described in Section 5.4.1 has the advantage of aiding delta feeds from remote domains where each domain's information is in its own DN subtree.

9. Security Considerations

Caching of remote domains' LDAP search results persents some security considerations. For example, some attributes' values may not be visible unless a user's credentials are used. Some attributes' values may not be intended to be visible to users, but to hosts. Caching servers MUST be capable of issuing referrals as needed for attributes whose values they may not read. Some domain federations will want to have their domains trust each others' caching servers.

More considerations to come

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.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC3530] Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M., and D. Noveck, "Network File System (NFS) version 4 Protocol", RFC 3530, April 2003.

[RFC1831] Srinivasan, R., "RPC: Remote Procedure Call Protocol Specification Version 2", RFC 1831, August 1995.

[RFC2203] Eisler, M., Chiu, A., and L. Ling, "RPCSEC_GSS Protocol Specification", RFC 2203, September 1997.

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

[RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005.

[RFC4511] Sermersheim, J., "Lightweight Directory Access Protocol (LDAP): The Protocol", RFC 4511, June 2006.

[RFC2307] Howard, L., "An Approach for Using LDAP as a Network Information Service", RFC 2307, March 1998.

[I-D.howard-rfc2307bis] Howard, L. and H. Chu, "An Approach for Using LDAP as a Network Information Service", draft-howard-rfc2307bis-02 (work in progress), August 2009.

[I-D.sorce-krbwg-general-pac] Sorce, S., Yu, T., and T. Hardjono, "A Generalized PAC for Kerberos V5", draft-sorce-krbwg-general-pac-01 (work in progress), December 2010.

[I-D.ietf-nfsv4-federated-fs-reqts] Lentini, J., Everhart, C., Ellard, D., Tewari, R., and M. Naik, "Requirements for Federated File Systems", draft-ietf-nfsv4-federated-fs-reqts-06 (work in progress), October 2009.

[I-D.ietf-kitten-gssapi-naming-exts]
Williams, N. and L. Johansson, "GSS-API Naming Extensions", draft-ietf-kitten-gssapi-naming-exts-09 (work in progress), February 2011.

[I-D.ietf-nfsv4-minorversion1]
Shepler, S., Eisler, M., and D. Noveck, "NFS Version 4 Minor Version 1", draft-ietf-nfsv4-minorversion1-29 (work in progress), December 2008.

[PAC] Brezak, J., "Utilizing the Windows 2000 Authorization Data in Kerberos Tickets for Access Control to Resources", October 2002.

10.2. Informative References

[I-D.williams-rpcsecgssv3]
Haynes, T. and N. Williams, "Remote Procedure Call (RPC) Security Version 3", draft-williams-rpcsecgssv3-01 (work in progress), March 2011.

[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.

[NIS] Sun Microsystems, "System and Network Administration", March 1990.

Authors' Addresses

William A. (Andy) Adamson
NetApp

Email: andros@netapp.com

Kevin Coffman
CITI, University of Michigan

Email: kwc@umich.edu

Nicolas Williams
Oracle

Email: nicolas.williams@oracle.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 23, 2012

J. Howlett
JANET(UK)
S. Hartman
Painless Security
October 21, 2011

Key Negotiation Protocol (KNP)
draft-howlett-radsec-knp-02

Abstract

The Key Negotiation Protocol enables an untrusting RADIUS client and RADIUS server to derive a key by reference to a mutually trusted actor called the Introducer. This key may subsequently be used for one of two purposes. First, it can credential a TLS PSK ciphersuite applied to a RadSec connection between the RADIUS client and RADIUS server; or secondly, to establish a trust relationship between the RADIUS client and a second Introducer that is trusted by the first Introducer.

The composition of these capabilities enables a RADIUS client to establish a RadSec connection with any RADIUS server with whom it shares a direct or indirect trust relationship via one or more Introducers.

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 April 23, 2012.

Copyright Notice

Table of Contents

1. Introduction

TLS encryption for RADIUS (RadSec) [I-D.ietf-radext-radsec] provides a mechanism for securing the communication between a RADIUS [RFC2865] client and server on the transport layer by using TLS [RFC5246].

RadSec mandates the use of one of the [RFC5246] ciphersuites and recommends the use of two other ciphersuites specified in that document. However any ciphersuite, including the TLS Pre-Shared Key (PSK) ciphersuites [RFC4279], may be used providing that it supports encryption.

The Key Negotiation Protocol enables an untrusting RADIUS client and RADIUS server to derive a key by means of a mutually trusted actor called the Introducer. This key may subsequently for two purposes.

First, the key can be used to credential a TLS PSK ciphersuite when applied to a RadSec connection between the RADIUS client and RADIUS server, permitting a trusted exchange of RADIUS messages in the absence of a pre-existing relationship between the RADIUS client and RADIUS server. This is described as "Credentialing".

Secondly, the key can used as a credential by a RADIUS client to establish a trust relationship with a second Introducer that happens to be trusted by the first Introducer. This is described as "Introduction".

The composition of Credentialing and Introduction enables a RADIUS client to establish a RadSec connection with any RADIUS server with whom it shares an indirect trust relationship via one or more Introducers.

2. Motivation

The KNP is motivated by the following requirements:

- In the case of a non-federated RADIUS environment where a RADIUS client and RADIUS AS.server shares a direct trust relationship, a shared secret credential is used as the trust anchor between these systems. In transitioning to the use of RadSec, it may be more convenient if these systems are able to continue using the existing credential technology rather than introduce a new credential technology (such as X.509 certificates), as this may impose significant changes to operational practices (such as deploying a Public Key Infrastructure).

- In the case of a federated RADIUS environment where RADIUS clients and RADIUS servers are associated with different domains,

transitioning to the use of RadSec may impose a requirement to distribute and manage multiple trust anchors. It may be more convenient if the systems within these domains were able to use a single trust anchor for RADIUS systems in all other domains, in addition to those systems within its own domain. This may facilitate the scaling of large heterogeneous RADIUS environments where it may be difficult - for technical and/or administrative reasons - to impose support for even a small set of trust anchors.

- o The use of multiple trust anchors within complex federated environments may impede essential trust management functions such as timely revocation. Reducing the number of trust anchors may therefore improve trust management within these environments, particularly if it can be reduced to a single trust anchor.

3. Overview

The Key Negotiation Protocol (KNP) enables a RADIUS client and RADIUS server that do not share a direct trust relationship to derive a shared key by virtue of both systems having a trust relationship with an EAP server called the Introducer. This key may be used for the following purposes:

1. Credentialing: the RADIUS client and RADIUS server can use the key to credential a TLS PSK ciphersuite applied to a RadSec connection.

2. Introduction: a credential can be derived from the key that can be used to authenticate the RADIUS client against a second Introducer that is trusted by the first Introducer.

The composition of these capabilities enables a RADIUS client to derive a key that can be used to credential a RadSec connection with any other RADIUS server with whom it shares a common Introducer and, through transitivity, any number of intermediate Introducers.

This transitivity of trust between a RADIUS client and RADIUS server across a chain of intermediate Introducers may appear very similar to the use of RADIUS proxies to establish a chain of trust between a RADIUS client and RADIUS server. There is however a very significant difference:

- o In the case of RADIUS proxy, the RADIUS messages emitted by the RADIUS client and RADIUS server must transit through the intermediate RADIUS proxy(ies). There is no end-to-end relationship between the RADIUS client and RADIUS server, either in terms of connectivity or trust.

- o In the case of KNP, the RADIUS messages are able to transit directly between RADIUS client and RADIUS server. The path of transmissions between these systems is therefore entirely decoupled from the path of trust . There is an end-to-end relationship between the RADIUS client and RADIUS server, both in terms of connectivity and trust.

The use of RADIUS Proxies and Introducers are not mutually exclusive; deployers may choose to use both.

4. 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 RFC 2119 [RFC2119].

5. The KNP Actors

In the KNP, the RADIUS client and RADIUS server do not initially share a trust relationship. Instead, these actors share a trust relationship with a mutually trusted third party known as the "Introducer".

Figure 1 below depicts the trust relationships for a RADIUS client, RADIUS server and Introducer before the KNP has been invoked.

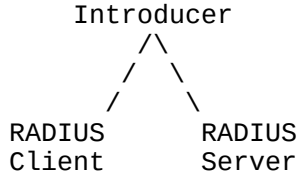


Figure 1

Figure 2 below depicts the new trust relationship between the RADIUS client, RADIUS server and Introducer after the KNP has been invoked.

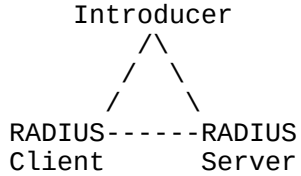


Figure 2

Figure 3 below depicts the flow of RADIUS packets from the RADIUS

client to the RADIUS server using the new trust relationship.

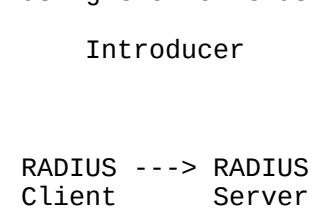


Figure 3

Note that the RADIUS messages are not routed by the Introducer, as they would in the case of a RADIUS Proxy. Instead, they flow directly from RADIUS client to RADIUS server.

6. Relationships With Other Protocols

The KNP builds on a variety of protocols. This section describes the relationship of KNP to these.

6.1. Relationship to EAP

In the KNP the RADIUS client assumes the role of an EAP peer. In this role, it attempts to authenticate against a RADIUS server that assumes the role of a pass-through EAP authenticator. An EAP server acts as the Introducer.

The KNP enables all three actors - RADIUS client (EAP peer), RADIUS server (EAP authenticator) and Introducer (EAP server) - to establish a common view of their mutual relationships as described by the EAP names and keys that the EAP exchange yields, using the norms established by the EAP Key Management Framework [RFC5247].

The RADIUS client must possess an EAP credential for the Introducer, allowing mutual authentication of both parties.

Figure 4 below depicts the relationships between these actors:

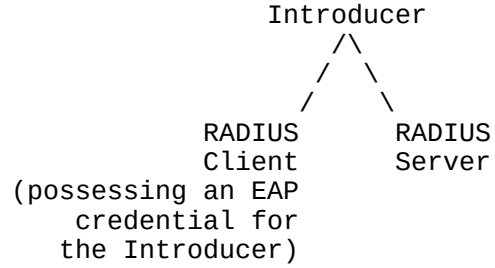


Figure 4

6.2. Relationship to RADIUS

The RADIUS server uses the RADIUS protocol to forward the EAP transaction to the Introducer.

The RADIUS server must share a RADIUS secret with the Introducer, allowing mutual authentication of both actors.

Figure 5 below depicts the relationships between these actors:

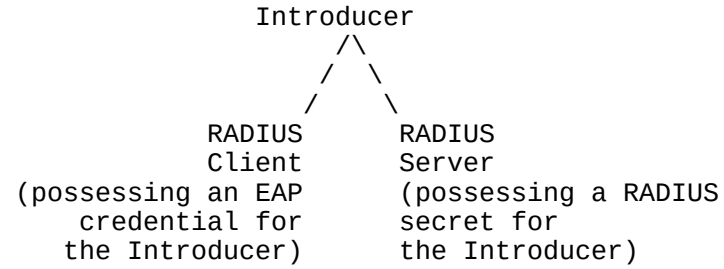


Figure 5

6.3. Relationship to the GSS API

The KNP builds on the GSS API [RFC2743] framework by using the GSS EAP mechanism [I-D.ietf-abfab-gss-eap] and EAP [RFC3748]. The GSS EAP tokens are transported between the RADIUS client and RADIUS server using the HTTP Negotiate authentication scheme [RFC4559].

6.4. Relationship to the HTTP

The KNP uses HTTP to transport its request and response protocol messages between the RADIUS Client and RADIUS server.

7. Key Negotiation Protocol

As described previously, the KNP provides two operations: Credentialing and Introduction.

The KNP provides these operations using a common protocol pattern. This pattern is applied against two types of Target actor, depending on the operation in question:

- In the case of Credentialing, the Target actor is a RADIUS server. If Credentialing is successful, the RADIUS client and RADIUS server will derive a common PSK that can be applied with a TLS-PSK

ciphersuite and RadSec.

- In the case of Introduction, the Target actor is an Introducer. If an Introduction is successful, the RADIUS client and Introducer will derive an EAP credential that can subsequently be used for subsequent Credentialing or Introduction operations.

For both operations it is essential that all three actors - RADIUS Client, Introducer and Target (whether a RADIUS server, in the case of Credentialing, or another Introducer, in the case of Introduction) - are able to authorise the claims that the other actors make about their respective names. These claims are validated using different processes for each relationship; these are summarised in Figure 6 below.

Subject	Relying Party	Process	Evidence from
RADIUS Client	Introducer	GSS EAP authentication	EAP method w/ qualifying Security Claims
Introducer	RADIUS Client		
Introducer	Target	RADIUS authentication	RADIUS shared secret
Target	Introducer		
Target	RADIUS Client	Channel bindings	Assertion by Introducer
RADIUS Client	Target	RADIUS attribute	Assertion by Introducer

Figure 6

7.1. Operation Independent Flow

The RADIUS Client invokes the KNP by establishing an HTTP connection with the Target’s KNP end-point.

The RADIUS Client MUST use the GSS EAP mechanism [I-D.ietf-abfab-gss-eap] to authenticate to the Introducer, requesting mutual authentication from the GSS layer.

The RADIUS Client, Target and Introducer MUST support EAP channel bindings [I-D.ietf-emu-chbind]. The Introducer MUST use validate the EAP channel bindings [I-D.ietf-emu-chbind] provided by the RADIUS Client. If the channel binding verification fails, the Introducer MUST reject the authentication.

The completion of the EAP method exchange results in the derivation of an EAP MSK known only to the RADIUS Client and Introducer and Peer-Id(s) and Server-Id(s) identifying these respectively. The Introducer MUST replicate the keying material and Server-Id to the Target.

The RADIUS Client and Target, in possession of the EAP MSK, create a GSS-API security context as described in section 6 of [I-D.ietf-abfab-gss-eap].

The RADIUS Client POSTs a key negotiation request, encoded as an HTML form dataset, to the Target. This request contains a set of operation-specific controls that specifies key negotiation parameters. A key negotiation request MUST contain the following controls:

- Version: the version of the KNP.
- Request-Identifier: a unique alphanumeric identifier for the request.
- Requestor-Name: the requestor’s GSS EAP initiator name.
- Operation-Type: the type of operation.
- Authenticator-Type: message authentication algorithm.
- Authenticator-Value: message authenticator value.

The Target extracts the key negotiation parameters and assesses their compliance to the Target’s key negotiation policies. The Target MUST return an operation-specific document providing information about the resulting key negotiation context.

- Version: the version of the KNP.
- Request-Identifier: the identifier for the request that this is a reponse to.
- Responder-Name: the requestor’s GSS EAP acceptor name.
- Operation-Type: the type of operation.
- Status-Code: a status code.
- Expires-After: a timestamp indicating the time of expiration.

- Authenticator-Type: message authentication algorithm.

- Authenticator-Value: message authenticator value.

TODO: consider use of SAML authentication assertion instead?

The RADIUS server and client SHOULD cache the GSS context until expiry of the GSS context. However, either party MAY delete a GSS context at any time. If a GSS context is deleted, any operation-specific derived materials SHOULD also be deleted, although such materials MAY be retained for auditing purposes.

7.2. The Credentialing Operation

This section describes the Credentialing operation-specific extensions to the general KNP flow.

The RADIUS Client MUST specify the following control values within the key negotiation request:

- Operation-Type: Credentialing

The PSK identity and value shall be outputs of GSS_Pseudo_random() [RFC4401] using the Pseudo-Random Function defined for the GSS EAP mechanism [I-D.ietf-abfab-gss-eap].

For the PSK identity, the prf_in input string MUST be prepended with the string "tls-psk-knp-identity"; desired_out_len MUST be set to 128 octets.

For the PSK value, the prf_in input string MUST be prepended with the string "tls-psk-knp-value"; desired_out_len MUST be set to 64 octets.

Note: these output values should use base64 encoding

7.3. The Introduction Operation

This section describes the Introduction operation-specific extensions to the general KNP flow.

The RADIUS Client MUST specify the following control values within the key negotiation request:

- Operation-Type: Introduction"

The EAP identity and credential shall be outputs of GSS_Pseudo_random() [RFC4401] using the Pseudo-Random Function defined for the GSS EAP mechanism [I-D.ietf-abfab-gss-eap].

For the EAP identity, the prf_in input string MUST be prepended with the string "tls-psk-eap-identity"; desired_out_len MUST be set to 128 octets. The output string MUST be appended with the realm of the Introducer to form an NAI.

For the EAP credential, the prf_in input string MUST be prepended with the string "tls-psk-eap-psk"; desired_out_len MUST be set to 64 octets.

Note: these output values should use base64 encoding.

8. Security Considerations

TODO

9. IANA Considerations

TODO

10. Acknowledgements

TODO

11. Normative References

[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.

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[RFC4279] Eronen, P. and H. Tschofenig, "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, December 2005.

[RFC4401] Williams, N., "A Pseudo-Random Function

(PRF) API Extension for the Generic Security Service Application Program Interface (GSS-API)", RFC 4401, February 2006.

[RFC4559] Jaganathan, K., Zhu, L., and J. Brezak, "SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows", RFC 4559, June 2006.

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

[RFC5247] Aboba, B., Simon, D., and P. Eronen, "Extensible Authentication Protocol (EAP) Key Management Framework", RFC 5247, August 2008.

[I-D.ietf-abfab-gss-eap] Hartman, S. and J. Howlett, "A GSS-API Mechanism for the Extensible Authentication Protocol", draft-ietf-abfab-gss-eap-03 (work in progress), October 2011.

[I-D.ietf-radext-radsec] Winter, S., McCauley, M., Venaas, S., and K. Wierenga, "TLS encryption for RADIUS", draft-ietf-radext-radsec-09 (work in progress), July 2011.

[I-D.ietf-emu-chbind] Hartman, S., Clancy, T., and K. Hoeper, "Channel Binding Support for EAP Methods", draft-ietf-emu-chbind-10 (work in progress), October 2011.

Authors' Addresses

Josh Howlett
JANET(UK)
Lumen House, Library Avenue, Harwell
Oxford OX11 0SG
UK

Phone: +44 1235 822363
EMail: Josh.Howlett@ja.net

Sam Hartman
Painless Security

EMail: hartmans-ietf@mit.edu

ABFAB
Internet-Draft
Intended status: Standards Track
Expires: July 14, 2016

J. Howlett
Janet
S. Hartman
Painless Security
A. Perez-Mendez, Ed.
University of Murcia
January 11, 2016

A RADIUS Attribute, Binding, Profiles, Name Identifier Format, and Confirmation Methods for SAML
draft-ietf-abfab-aaa-saml-14

Abstract

This document describes the use of the Security Assertion Mark-up Language (SAML) with RADIUS in the context of the ABFAB architecture. It defines two RADIUS attributes, a SAML binding, a SAML name identifier format, two SAML profiles, and two SAML confirmation methods. The RADIUS attributes permit encapsulation of SAML assertions and protocol messages within RADIUS, allowing SAML entities to communicate using the binding. The two profiles describe the application of this binding for ABFAB authentication and assertion query/request, enabling a Relying Party to request authentication of, or assertions for, users or machines (Clients). These Clients may be named using a NAI name identifier format. Finally, the subject confirmation methods allow requests and queries to be issued for a previously authenticated user or machine without needing to explicitly identify them as the subject. The use of the artifacts defined in this document is not exclusive to ABFAB. They can be applied in any AAA scenario, such as the network access control.

Status of This Memo

This Internet-Draft will expire on July 14, 2016.

Copyright Notice

Table of Contents

1. Introduction

Within the ABFAB (Application Bridging for Federated Access Beyond web) architecture [I-D.ietf-abfab-arch] it is often desirable to convey Security Assertion Mark-up Language (SAML) assertions and protocol messages.

SAML typically only considers the use of HTTP-based transports, known as bindings [OASIS.saml-bindings-2.0-os], which are primarily intended for use with the SAML V2.0 Web Browser Single Sign-On Profile [OASIS.saml-profiles-2.0-os]. However the goal of ABFAB is to extend the applicability of federated identity beyond the Web to other applications by building on the AAA framework. Consequently there exists a requirement for SAML to integrate with the AAA framework and protocols such as RADIUS [RFC2865] and Diameter [RFC6733], in addition to HTTP.

In summary this document specifies:

- Two RADIUS attributes to encapsulate SAML assertions and protocol messages respectively.
- A SAML RADIUS binding that defines how SAML assertions and protocol messages can be transported by RADIUS within a SAML exchange.
- A SAML name identifier format in the form of a Network Access Identifier.
- A profile of the SAML Authentication Request Protocol that uses the SAML RADIUS binding to effect SAML-based authentication and authorization.
- A profile of the SAML Assertion Query And Request Protocol that uses the SAML RADIUS binding to effect the query and request of SAML assertions.
- Two SAML Subject Confirmation Methods for indicating that a user or machine client is the subject of an assertion.

This document adheres to the guidelines stipulated by [OASIS.saml-bindings-2.0-os] and [OASIS.saml-profiles-2.0-os] for defining new SAML bindings and profiles respectively, and other conventions applied formally or otherwise within SAML. In particular, this document provides a 'Required Information' section for the binding and profiles that enumerate:

- A URI that uniquely identifies the protocol binding or profile.
- Postal or electronic contact information for the author.
- A reference to previously defined bindings or profiles that the new binding updates or obsoletes.
- In the case of a profile, any SAML confirmation method identifiers defined and/or utilized by the profile.

1.1. Terminology

This document uses terminology from a number of related standards, which tend to adopt different terms for similar or identical concepts. In general the document uses, when possible, the ABFAB term for the entity, as described in [I-D.ietf-abfab-arch]. For reference we include this table which maps the different terms into a single view.

Protocol	Client	Relying Party	Identity Provider
ABFAB	Client	Relying Party	Identity Provider
SAML	Subject Principal	Service Provider Requester Consumer	Identity Provider Responder Issuer
RADIUS	User	NAS RADIUS client	AS RADIUS server

Table 1. Terminology

2. Conventions

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].

3. RADIUS SAML Attributes

The RADIUS SAML binding defined in Section 4 of this document uses two attributes to convey SAML assertions and protocol messages [OASIS.saml-core-2.0-os]. Owing to the typical size of these structures, these attributes use the Long Extended Type format [RFC6929] to encapsulate their data. RADIUS entities MUST NOT include both attributes in the same RADIUS message, as they represent exclusive alternatives to convey SAML information.

3.1. SAML-Assertion attribute

This attribute is used to encode a SAML assertion. The following figure represents the format of this attribute.

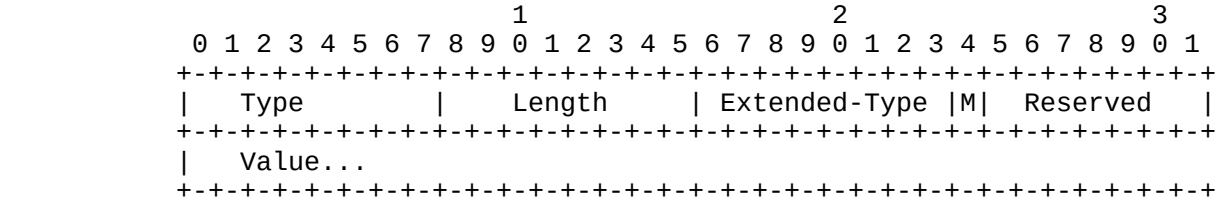


Figure 1: SAML-Assertion format

Type

245 (To be confirmed by IANA)

Length

>= 5

Extended-Type

TBD1

M (More)

As described in [RFC6929].

Reserved

As described in [RFC6929].

Value

One or more octets encoding a SAML assertion.

3.2. SAML-Protocol attribute

This attribute is used to encode a SAML protocol message. The following figure represents the format of this attribute.

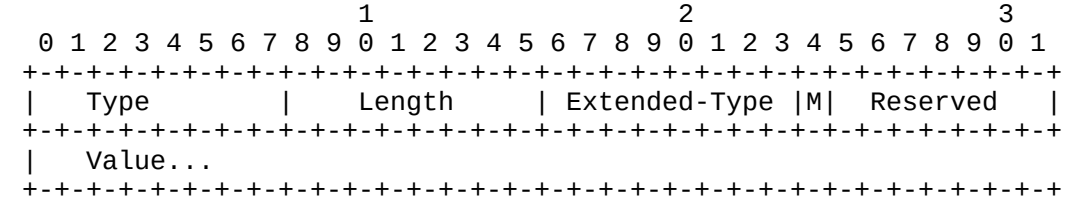


Figure 2: SAML-Protocol format

Type

245 (To be confirmed by IANA)

Length

>= 5

Extended-Type

TBD2

M (More)

As described in [RFC6929].

Reserved

As described in [RFC6929].

Value

One or more octets encoding a SAML protocol message.

4. SAML RADIUS Binding

The SAML RADIUS binding defines how RADIUS [RFC2865] can be used to enable a RADIUS client and server to exchange SAML assertions and protocol messages.

4.1. Required Information

Identification: urn:ietf:params:abfab:bindings:radius

Contact information: iesg@ietf.org

Updates: None.

4.2. Operation

In this specification, the Relying Party MUST trust any statement in the SAML messages from the IdP in the same way that it trusts information contained in RADIUS attributes. These entities MUST trust the RADIUS infrastructure to provide integrity of the SAML messages.

Hence, it is REQUIRED that the RADIUS exchange is protected using TLS encryption for RADIUS [RFC6614] to provide confidentiality and integrity protection, unless alternative methods to ensure them are used, such as IPSEC tunnels or a sufficiently secure internal network.

Implementations of this profile can take advantage of mechanisms to permit the transport of longer SAML messages over RADIUS transports, such as the Support of fragmentation of RADIUS packets [RFC7499] or Larger Packets for RADIUS over TCP [I-D.ietf-radext-bigger-packets].

There are two system models for the use of SAML over RADIUS. The first is a request-response model, using the RADIUS SAML-Protocol

attribute defined in Section 3 to encapsulate the SAML protocol messages.

1. The RADIUS client, acting as a Relying Party (RP), transmits a SAML request element within a RADIUS Access-Request message. This message MUST include a single instance of the RADIUS User-Name attribute whose value MUST conform to the Network Access Identifier [RFC7542] scheme. The Relying Party MUST NOT include more than one SAML request element.

2. The RADIUS server, acting as an Identity Provider (IdP), returns a SAML protocol message within a RADIUS Access-Accept or Access-Reject message. These messages necessarily conclude a RADIUS exchange and therefore this is the only opportunity for the Identity Provider to send a response in the context of this exchange. The Identity Provider MUST NOT include more than one SAML response. An IdP that refuses to perform a message exchange with the Relying Party can silently discard the SAML request (this could subsequently be followed by a RADIUS Access-Reject, as the same conditions that cause the IdP to discard the SAML request may also cause the RADIUS server to fail to authenticate).

The second system model permits a RADIUS server acting as an Identity Provider to use the RADIUS SAML-Assertion attribute defined in Section 3 to encapsulate an unsolicited SAML assertion. This attribute MUST be included in a RADIUS Access-Accept message. When included, the attribute MUST contain a single SAML assertion.

RADIUS servers MUST NOT include both the SAML-Protocol and the SAML-Assertion attribute in the same RADIUS message. If an IdP is producing a response to a SAML request, then the first system model is used. An IdP MAY ignore a SAML request and send an unsolicited assertion using the second system model using the RADIUS SAML-Assertion attribute.

In either system model, Identity Providers SHOULD return a RADIUS state attribute as part of the Access-Accept message so that future SAML queries or requests can be run against the same context of an authentication exchange.

This binding is intended to be composed with other uses of RADIUS, such as network access. Therefore, other arbitrary RADIUS attributes MAY be used in either the request or response.

In the case of a SAML processing error, the RADIUS server MAY include a SAML response message with an appropriate value for the <samlp:Status> element within the Access-Accept or Access-Reject

packet to notify the client. Alternatively, the RADIUS server can respond without a SAML-Protocol attribute.

4.3. Processing of names

SAML entities using profiles making use of this binding will typically possess both the SAML and AAA names of their correspondents. Frequently these entities will need to apply policies using these names; for example, when deciding to release attributes. Often these policies will be security-sensitive, and so it is important that policy is applied on these names consistently.

4.3.1. AAA names

These rules relate to the processing of AAA names by SAML entities using profiles making use of this binding.

- Identity Providers SHOULD apply policy based on the Relying Party's identity associated with the RADIUS Access-Request.

- Relying Parties SHOULD apply policy based on the NAI realm associated with the RADIUS Access-Accept.

4.3.2. SAML names

These rules relate to the processing of SAML names by SAML entities using profiles making use of this binding.

Identity Providers MAY apply policy based on the Relying Party's SAML entityId. In such cases, at least one of the following methods is required in order to establish a relation between the SAML name and the AAA name of the Relying Party:

- RADIUS client identity in trusted SAML metadata (as described in section Section 4.3.3).

- RADIUS client identity in trusted digitally signed SAML request.

A digitally signed SAML request without the RADIUS client identity is not sufficient, since a malicious RADIUS entity can observe a SAML message and include it in a different RADIUS message without the consent of the issuer of that SAML message. If an Identity Provider were to process the SAML message without confirming that it applied to the RADIUS message, inappropriate policy would be used.

Relying Parties MAY apply policy based on the SAML issuer's <entityId>. In such cases, at least one of the following methods is

required in order to establish a relationship between the SAML name and the AAA name of the Identity Provider:

- RADIUS realm in trusted SAML metadata (as described in section Section 4.3.3).
- RADIUS realm in trusted digitally signed SAML response or assertion.

A digitally signed SAML response alone is not sufficient for the same reasons described above for SAML requests.

4.3.3. Mapping of AAA names in SAML metadata

This section defines extensions to the SAML metadata schema [OASIS.saml-metadata-2.0-os] that are required in order to represent AAA names associated with a particular <EntityDescriptor> element.

In SAML metadata, a single entity may act in many different roles in the support of multiple profiles. This document defines two new roles: RADIUS IDP and RADIUS RP, requiring the declaration of two new subtypes of RoleDescriptorType: RADIUSIDPDescriptorType and RADIUSRPDescriptorType. These subtypes contain the additional elements required to represent AAA names for IDP and RP entities respectively.

4.3.3.1. RADIUSIDPDescriptorType

The RADIUSIDPDescriptorType complex type extends RoleDescriptorType with elements common to IdPs that support RADIUS. It contains the following additional elements:

<RADIUSIDPService> [Zero or More] Zero or more elements of type EndpointType that describe RADIUS endpoints that are associated with the entity.

<RADIUSRealm> [Zero or More] Zero or more elements of type string that represent the acceptable values of the RADIUS realm associated with the entity, obtained from the realm part of RADIUS User-Name attribute.

The following schema fragment defines the RADIUSIDPDescriptorType complex type:

```
        <complexType name="RADIUSIDPDescriptorType">
          <complexContent>
            <extension base="md:RoleDescriptorType">
              <sequence>
                <element ref="abfab:RADIUSIDPService" minOccurs="0" maxOccurs="u
nbounded"/>
                <element ref="abfab:RADIUSRealm" minOccurs="0" maxOccurs="unboun
ded"/>
              </sequence>
            </extension>
          </complexContent>
        </complexType>
        <element name="RADIUSIDPService" type="md:EndpointType"/>
        <element name="RADIUSRealm" type="string"/>
```

Figure 3: RADIUSIDPDescriptorType schema

4.3.3.2. RADIUSRPDescriptorType

The RADIUSRPDescriptorType complex type extends RoleDescriptorType with elements common to RPs that support RADIUS. It contains the following additional elements:

<RADIUSRPService> [Zero or More] Zero or more elements of type EndpointType that describe RADIUS endpoints that are associated with the entity.

<RADIUSNasIpAddress> [Zero or More] Zero or more elements of type string that represent the acceptable values of the RADIUS NAS-IP-Address or NAS-IPv6-Address attributes associated with the entity.

<RADIUSNasIdentifier> [Zero or More] Zero or more elements of type string that represent the acceptable values of the RADIUS NAS-Identifier attribute associated with the entity.

<RADIUSGssEapName> [Zero or More] Zero or more elements of type string that represent the acceptable values of the GSS-EAP acceptor name associated with the entity. The format for this name is described in section 3.1 of [RFC7055], while section 3.4 describes how that name is decomposed and transported using RADIUS attributes.

The following schema fragment defines the RADIUSRPDescriptorType complex type:

```
    <complexType name="RADIUSRPDescriptorType">
      <complexContent>
        <extension base="md:RoleDescriptorType">
          <sequence>
            <element ref="md:RADIUSRPService" minOccurs="0" maxOccurs="unbounded
"/>
            <element ref="md:RADIUSNasIpAddress" minOccurs="0" maxOccurs="unboun
ded"/>
            <element ref="md:RADIUSNasIdentifier" minOccurs="0" maxOccurs="unbou
nded"/>
            <element ref="md:RADIUSGssEapName" minOccurs="0" maxOccurs="unbounde
d"/>
          </sequence>
        </extension>
      </complexContent>
    </complexType>
    <element name="RADIUSRPService" type="md:EndpointType"/>
    <element name="RADIUSNasIpAddress" type="string"/>
    <element name="RADIUSNasIdentifier" type="string"/>
    <element name="RADIUSGssEapName" type="string"/>
```

Figure 4: RADIUSRPDescriptorType schema

4.3.4. Example of SAML metadata including AAA names

The following figures illustrate an example of metadata including AAA names for and IDP and a RP respectively. The IDP's SAML name is "https://IdentityProvider.com/", whereas its RADIUS realm is "idp.com". The RP's SAML name is "https://RelyingParty.com/SAML", being its GSS-EAP acceptor name "nfs/fileserver.rp.com@RP.COM".

```
    <EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
                      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                      xmlns:abfab="urn:ietf:params:xml:ns:abfab"
                      entityID="https://IdentityProvider.com/SAML">
        <RoleDescriptor xsi:type="abfab:RADIUSIDPDescriptorType"
              protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
            <RADIUSRealm>idp.com</RADIUSRealm>
        </RoleDescriptor>
    </EntityDescriptor>
```

Figure 5: Metadata for the IDP

```
    <EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
                      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                      xmlns:abfab="urn:ietf:params:xml:ns:abfab"
                      entityID="https://RelyingParty.com/SAML">
        <RoleDescriptor xsi:type="abfab:RADIUSRPDescriptorType"
              protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
            <RADIUSGssEapName>nfs/fileserver.rp.com@RP.COM</RADIUSGssEapName>
        </RoleDescriptor>
    </EntityDescriptor>
```

Figure 6: Metadata for the RP

4.4. Use of XML Signatures

This binding calls for the use of SAML elements that support XML signatures. To promote interoperability, implementations of this binding MUST support a default configuration that does not require the use of XML signatures. Implementations MAY choose to use XML signatures.

4.5. Metadata Considerations

These binding and profiles are mostly intended to be used without metadata. In this usage, RADIUS infrastructure is used to provide integrity and naming of the SAML messages and assertions. RADIUS configuration is used to provide policy, including which attributes are accepted from a Relying Party and which attributes are sent by an Identity Provider.

Nevertheless, if metadata is used, the roles describe in section Section 4.3.3 MUST be present.

5. Network Access Identifier Name Identifier Format

URI: urn:ietf:params:abfab:nameid-format:nai

Indicates that the content of the element is in the form of a Network Access Identifier (NAI) using the syntax described by [RFC7542].

6. RADIUS State Confirmation Method Identifiers

URI: urn:ietf:params:abfab:cm:user

URI: urn:ietf:params:abfab:cm:machine

Indicates that the Subject is the system entity (either the user or machine) authenticated by a previously transmitted RADIUS Access-

Accept message, as identified by the value of that RADIUS message's State attribute.

7. ABFAB Authentication Profile

In the scenario supported by the ABFAB Authentication Profile, a Client controlling a User Agent requests access to a Relying Party. The Relying Party uses RADIUS to authenticate the Client. In particular, the Relying Party, acting as a RADIUS client, attempts to validate the Client's credentials against a RADIUS server acting as the Client's Identity Provider. If the Identity Provider successfully authenticates the Client, it produces an authentication assertion which is consumed by the Relying Party. This assertion MAY include a name identifier that can be used between the Relying Party and the Identity Provider to refer to the Client.

7.1. Required Information

Identification: urn:ietf:params:abfab:profiles:authentication

Contact information: iesg@ietf.org

SAML Confirmation Method Identifiers: The SAML V2.0 "RADIUS State" confirmation method identifiers, either urn:ietf:params:abfab:cm:user or urn:ietf:params:abfab:cm:machine, are used by this profile.

Updates: None.

7.2. Profile Overview

To implement this scenario, this profile of the SAML Authentication Request protocol MUST be used in conjunction with the SAML RADIUS binding defined in Section 4.

This profile is based on the SAML V2.0 Web Browser Single Sign-On Profile [OASIS.saml-profiles-2.0-os]. There are some important differences, specifically:

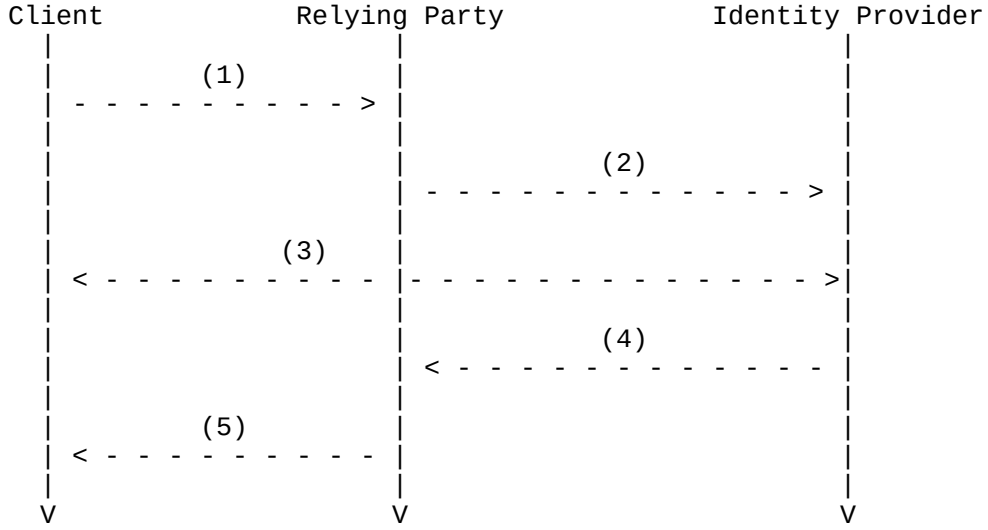
Authentication: This profile does not require the use of any particular authentication method. The ABFAB architecture does require the use of EAP [RFC3579], but this specification may be used in other non-ABFAB scenarios.

Bindings: This profile does not use HTTP-based bindings. Instead all SAML protocol messages are transported using the SAML RADIUS binding defined in Section 4. This is intended to reduce the number of bindings that implementations must support to be interoperable.

Requests: The profile does not permit the Relying Party to name the <saml:Subject> of the <samlp:AuthnRequest>. This is intended to simplify implementation and interoperability.

Responses: The profile only permits the Identity Provider to return a single SAML message or assertion that MUST contain exactly one authentication statement. Other statements may be included within this assertion at the discretion of the Identity Provider. This is intended to simplify implementation and interoperability.

Figure 7 below illustrates the flow of messages within this profile.



The following steps are described by the profile. Within an individual step, there may be one or more actual message exchanges.

Figure 7

1. Client request to Relying Party (Section 7.3.1): In step 1, the Client, via a User Agent, makes a request for a secured resource at the Relying Party. The Relying Party determines that no security context for the Client exists and initiates the authentication process.

2. Relying Party issues <samlp:AuthnRequest> to Identity Provider (Section 7.3.2). In step 2, the Relying Party may optionally issue a <samlp:AuthnRequest> message to be delivered to the Identity Provider using the SAML-Protocol RADIUS attribute.

3. Identity Provider identifies Client (Section 7.3.3). In step 3, the Client is authenticated and identified by the Identity Provider, while honoring any requirements imposed by the Relying Party in the <samlp:AuthnRequest> message if provided.

4. Identity Provider issues <samlp:Response> to Relying Party (Section 7.3.4). In step 4, the Identity Provider issues a <samlp:Response> message to the Relying Party using the SAML RADIUS binding. The response either indicates an error or includes a SAML Authentication Statement in exactly one SAML Assertion. If the RP did not send an <samlp:AuthnRequest>, the IdP issues an unsolicited <samlp:Assertion>, as described in Section 7.4.4.

5. Relying Party grants or denies access to Client (Section 7.3.5). In step 5, having received the response from the Identity Provider, the Relying Party can respond to the Client with its own error, or can establish its own security context for the Client and return the requested resource.

7.3. Profile Description

The ABFAB Authentication Profile is a profile of the SAML V2.0 Authentication Request Protocol [OASIS.saml-core-2.0-os]. Where both specifications conflict, the ABFAB Authentication Profile takes precedence.

7.3.1. Client Request to Relying Party

The profile is initiated by an arbitrary Client request to the Relying Party. There are no restrictions on the form of the request. The Relying Party is free to use any means it wishes to associate the subsequent interactions with the original request. The Relying Party, acting as a RADIUS client, attempts to authenticate the Client.

7.3.2. Relying Party Issues <samlp:AuthnRequest> to Identity Provider

The Relying Party uses RADIUS to communicate with the Client’s Identity Provider. The Relying Party MAY include a <samlp:AuthnRequest> within this RADIUS Access-Request message using the SAML-Protocol RADIUS attribute. The next hop destination MAY be the Identity Provider or alternatively an intermediate RADIUS proxy.

Profile-specific rules for the contents of the <samlp:AuthnRequest> element are given in Section 7.4.1.

7.3.3. Identity Provider Identifies Client

The Identity Provider MUST establish the identity of the Client using a RADIUS authentication method, or else it will return an error. If the ForceAuthn attribute on the <samlp:AuthnRequest> element (if sent by the Relying Party) is present and true, the Identity Provider MUST freshly establish this identity rather than relying on any existing session state it may have with the Client (for example, TLS state that may be used for session resumption). Otherwise, and in all other respects, the Identity Provider may use any method to authenticate the Client, subject to the constraints called out in the <samlp:AuthnRequest> message.

7.3.4. Identity Provider Issues <samlp:Response> to Relying Party

The Identity Provider MUST conclude the authentication in a manner consistent with the RADIUS authentication result. The IdP MAY issue a <samlp:Response> message to the Relying Party that is consistent with the authentication result, as described in [OASIS.saml-core-2.0-os]. This SAML response is delivered to the Relying Party using the SAML RADIUS binding described in Section 4.

Profile-specific rules regarding the contents of the <samlp:Response> element are given in Section 7.4.2.

7.3.5. Relying Party Grants or Denies Access to Client

If a <samlp:Response> message is issued by the Identity Provider, the Relying Party MUST process that message and any enclosed assertion elements as described in [OASIS.saml-core-2.0-os]. Any subsequent use of the assertion elements is at the discretion of the Relying Party, subject to any restrictions contained within the assertions themselves or from any previously established out-of-band policy that governs the interaction between the Identity Provider and the Relying Party.

7.4. Use of Authentication Request Protocol

This profile is based on the Authentication Request Protocol defined in [OASIS.saml-core-2.0-os]. In the nomenclature of actors enumerated in section 3.4 of that document, the Relying Party is the requester, the User Agent is the attesting entity and the Client is the Requested Subject.

7.4.1. <samlp:AuthnRequest> Usage

The Relying Party MUST NOT include a <saml:Subject> element in the request. The authenticated RADIUS identity identifies the Client to the Identity Provider.

A Relying Party MAY include any message content described in [OASIS.saml-core-2.0-os], section 3.4.1. All processing rules are as defined in [OASIS.saml-core-2.0-os].

If the Relying Party wishes to permit the Identity Provider to establish a new identifier for the Client if none exists, it MUST include a <saml:NameIDPolicy> element with the AllowCreate attribute set to "true". Otherwise, only a Client for whom the Identity Provider has previously established an identifier usable by the Relying Party can be authenticated successfully.

The <samlp:AuthnRequest> message MAY be signed. Authentication and integrity are also provided by the SAML RADIUS binding.

7.4.2. <samlp:Response> Message Usage

If the Identity Provider cannot or will not satisfy the request, it MUST either respond with a <samlp:Response> message containing an appropriate error status code or codes and/or respond with a RADIUS Access-Reject message.

If the Identity Provider wishes to return an error, it MUST NOT include any assertions in the <samlp:Response> message. Otherwise, if the request is successful (or if the response is not associated with a request), the <samlp:Response> element is subject to the following constraints:

- It MAY be signed.
- It MUST contain exactly one assertion. The <saml:Subject> element of this assertion MUST refer to the authenticated RADIUS user.
- The assertion MUST contain a <saml:AuthnStatement>. Besides, the assertion MUST contain a <saml:Subject> element with at least one <saml:SubjectConfirmation> element containing a Method of urn:ietf:params:abfab:cm:user or urn:ietf:params:abfab:cm:machine that reflects the authentication of the Client to the Identity Provider. Since the containing message is in response to an <samlp:AuthnRequest>, the InResponseTo attribute (both in the <saml:SubjectConfirmationData> and in the <saml:Response> elements) MUST match the request’s ID. The <saml:Subject> element

MAY use the NAI Name Identifier Format described in Section 5 to establish an identifier between the Relying Party and the IdP.

- Other conditions MAY be included as requested by the Relying Party or at the discretion of the Identity Provider. The Identity Provider is NOT obligated to honor the requested set of conditions in the <samlp:AuthnRequest>, if any.

7.4.3. <samlp:Response> Message Processing Rules

The Relying Party MUST do the following:

- Assume that the Client's identifier implied by a SAML <Subject> element, if present, takes precedence over an identifier implied by the RADIUS User-Name attribute.

- Verify that the InResponseTo attribute in the "RADIUS State" <saml:SubjectConfirmationData> equals the ID of its original <samlp:AuthnRequest> message, unless the response is unsolicited, in which case the attribute MUST NOT be present.

- If a <saml:AuthnStatement> used to establish a security context for the Client contains a SessionNotOnOrAfter attribute, the security context SHOULD be discarded once this time is reached, unless the Relying Party reestablishes the Client's identity by repeating the use of this profile.

- Verify that any assertions relied upon are valid according to processing rules in [OASIS.saml-core-2.0-os].

- Any assertion which is not valid, or whose subject confirmation requirements cannot be met MUST be discarded and MUST NOT be used to establish a security context for the Client.

7.4.4. Unsolicited Responses

An Identity Provider MAY initiate this profile by delivering an unsolicited assertion to a Relying Party. This MUST NOT contain any <saml:SubjectConfirmationData> elements containing an InResponseTo attribute.

7.4.5. Use of the SAML RADIUS Binding

It is RECOMMENDED that the RADIUS exchange is protected using TLS encryption for RADIUS [RFC6614] to provide confidentiality and integrity protection.

7.4.6. Use of XML Signatures

This profile calls for the use of SAML elements that support XML signatures. To promote interoperability implementations of this profile MUST NOT require the use of XML signatures. Implementations MAY choose to use XML signatures.

7.4.7. Metadata Considerations

There are no metadata considerations particular to this profile, aside from those applying to the use of the RADIUS binding.

8. ABFAB Assertion Query/Request Profile

This profile builds on the SAML V2.0 Assertion Query/Request Profile defined by [OASIS.saml-profiles-2.0-os]. That profile describes the use of the Assertion Query and Request Protocol defined by section 3.3 of [OASIS.saml-core-2.0-os] with synchronous bindings, such as the SOAP binding defined in [OASIS.saml-bindings-2.0-os].

While the SAML V2.0 Assertion Query/Request Profile is independent of the underlying binding, it is nonetheless useful to describe the use of the SAML RADIUS binding defined in Section 4 of this document, in the interests of promoting interoperable implementations, particularly as the SAML V2.0 Assertion Query/Request Profile is most frequently discussed and implemented in the context of the SOAP binding.

8.1. Required Information

Identification: urn:ietf:params:abfab:profiles:query

Contact information: iesg@ietf.org

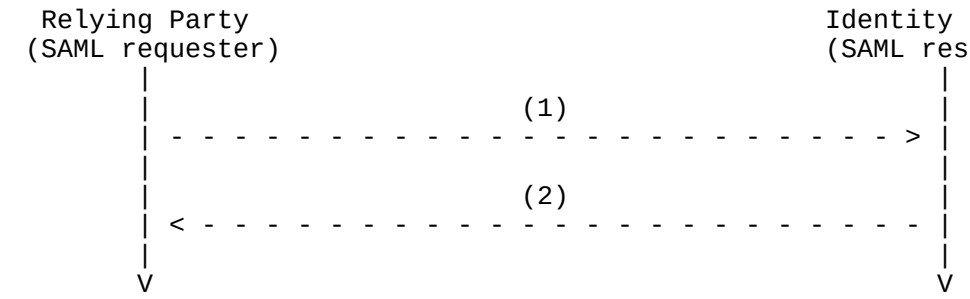
Description: Given below.

Updates: None.

8.2. Profile Overview

As with the SAML V2.0 Assertion Query/Request Profile defined by [OASIS.saml-profiles-2.0-os] the message exchange and basic processing rules that govern this profile are largely defined by Section 3.3 of [OASIS.saml-core-2.0-os] that defines the messages to be exchanged, in combination with the binding used to exchange the messages. The SAML RADIUS binding described in this document defines the binding of the message exchange to RADIUS. Unless specifically noted here, all requirements defined in those specifications apply.

Figure 8 below illustrates the basic template for the query/request profile.



The following steps are described by the profile.

Figure 8

1. Query/Request issued by Relying Party: In step 1, a Relying Party initiates the profile by sending an <AssertionIDRequest>, <SubjectQuery>, <AuthnQuery>, <AttributeQuery>, or <AuthzDecisionQuery> message to a SAML authority.

2. <Response> issued by SAML Authority: In step 2, the responding SAML authority (after processing the query or request) issues a <Response> message to the Relying Party.

8.3. Profile Description

8.3.1. Differences from the SAML V2.0 Assertion Query/Request Profile

This profile is identical to the SAML V2.0 Assertion Query/Request Profile, with the following exceptions:

- When processing the SAML request, the IdP MUST give precedence to the Client's identifier implied by RADIUS State attribute, if present, over the identifier implied by the SAML request's <Subject>, if any.

- In respect to sections 6.3.1 and 6.5 of [OASIS.saml-profiles-2.0-os], this profile does not consider the use of metadata (as in [OASIS.saml-metadata-2.0-os]). See Section 8.3.4.

- In respect to sections 6.3.2, 6.4.1, and 6.4.2 of [OASIS.saml-profiles-2.0-os], this profile additionally stipulates that implementations of this profile MUST NOT require the use of XML signatures. See Section 8.3.3.

8.3.2. Use of the SAML RADIUS Binding

The RADIUS Access-Request sent by the Relying Party:

- MUST include an instance of the RADIUS Service-Type attribute, having a value of Authorize-Only.
- SHOULD include the RADIUS State attribute, where this Query/ Request pertains to previously authenticated Client.

When processing the SAML request, the IdP MUST give precedence to the Client's identifier implied by RADIUS State attribute over the identifier implied by the SAML request's <Subject>, if any.

It is RECOMMENDED that the RADIUS exchange is protected using TLS encryption for RADIUS [RFC6614] to provide confidentiality and integrity protection.

8.3.3. Use of XML Signatures

This profile calls for the use of SAML elements that support XML signatures. To promote interoperability implementations of this profile MUST NOT require the use of XML signatures. Implementations MAY choose to use XML signatures.

8.3.4. Metadata Considerations

There are no metadata considerations particular to this profile, aside from those applying to the use of the RADIUS binding.

9. Privacy considerations

The profiles defined in this document allow a Relying Party to request specific information about the Client, and allow an IdP to disclose information about that Client. In this sense, Identity Providers MUST apply policy to decide what information is released to a particular Relying Party. Moreover, the identity of the Client is typically hidden from the Relying Party unless informed by the Identity Provider. Conversely, the Relying Party does typically know the realm of the IdP, as it is required to route the RADIUS packets to the right destination.

The kind of information that is released by the IdP can include generic attributes such as affiliation shared by many Clients. But even these generic attributes can help to identify a specific Client. Other kinds of attributes may also provide a Relying Party with the ability to link the same Client between different sessions. Finally, other kind of attributes might provide a group of Relying Parties

with the ability to link the Client between them or with personally identifiable information about the Client.

These profiles do not directly provide a Client with a mechanism to express preferences about what information is released. That information can be expressed out-of-band, for example as part of the enrollment process.

The Relying Party may disclose privacy-sensitive information about itself as part of the request, although this is unlikely in typical deployments.

If RADIUS proxies are used and encryption is not used, the attributes disclosed by the IdP are visible to the proxies. This is a significant privacy exposure in some deployments. Ongoing work is exploring mechanisms for creating TLS connections directly between the RADIUS client and the RADIUS server to reduce this exposure. If proxies are used, the impact of exposing SAML assertions to the proxies needs to be carefully considered.

The use of TLS to provide confidentiality for the RADIUS exchange is strongly encouraged. Without this, passive eavesdroppers can observe the assertions.

10. Security Considerations

In this specification, the Relying Party MUST trust any statement in the SAML messages from the IdP in the same way that it trusts information contained in RADIUS attributes. These entities MUST trust the RADIUS infrastructure to provide integrity of the SAML messages.

Furthermore, the Relying Party MUST apply policy and filter the information based on what information the IdP is permitted to assert and on what trust is reasonable to place in proxies between them.

XML signatures and encryption are provided as an OPTIONAL mechanism for end-to-end security. These mechanism can protect SAML messages from being modified by proxies in the RADIUS infrastructure. These mechanisms are not mandatory-to-implement. It is believed that ongoing work to provide direct TLS connections between a RADIUS client and RADIUS server will provide similar assurances but better deployability. XML security is appropriate for deployments where end-to-end security is required but proxies cannot be removed or where SAML messages need to be verified at a later time or by parties not involved in the authentication exchange.

11. IANA Considerations

11.1. RADIUS Attributes

The authors request that Attribute Types and Attribute Values defined in this document be registered by the Internet Assigned Numbers Authority (IANA) from the RADIUS namespaces as described in the "IANA Considerations" section of [RFC3575], in accordance with BCP 26 [RFC5226]. For RADIUS packets, attributes and registries created by this document IANA is requested to place them at http://www.iana.org/assignments/radius-types.

In particular, this document defines two new RADIUS attributes, entitled "SAML-Assertion" and "SAML-Protocol" (see Section 3), with assigned values of 245.TBD1 and 245.TBD2 from the Long Extended Space of [RFC6929]:

Type	Ext. Type	Name	Length	Meaning
245	TBD1	SAML-Assertion	>=5	Encodes a SAML assertion
245	TBD2	SAML-Protocol	>=5	Encodes a SAML protocol message

11.2. ABFAB Parameters

A new top-level registry is created titled "ABFAB Parameters".

In this top-level registry, a sub-registry titled "ABFAB URN Parameters" is created. Registration in this registry is by the IETF review or expert review procedures [RFC5226].

This paragraph gives guidance to designated experts. Registrations in this registry are generally only expected as part of protocols published as RFCs on the IETF stream; other URIs are expected to be better choices for non-IETF work. Expert review is permitted mainly to allow early registration related to specifications under development when the community believes they have reached sufficient maturity. The expert SHOULD evaluate the maturity and stability of such an IETF-stream specification. Experts SHOULD review anything not from the IETF stream for consistency and consensus with current practice. Today such requests would not typically be approved.

If a parameter named "paramname" is to be registered in this registry, then its URN will be "urn:ietf:params:abfab:paramname". The initial registrations are as follows:

Parameter	Reference
bindings:radius	Section 4
nameid-format:nai	Section 5
profiles:authentication	Section 7
profiles:query	Section 8
cm:user	Section 6
cm:machine	Section 6

ABFAB Parameters

11.3. Registration of the ABFAB URN Namespace

IANA is requested to register the "abfab" URN sub-namespace in the IETF URN sub-namespace for protocol parameters defined in [RFC3553].

Registry Name: abfab

Specification: draft-ietf-abfab-aaa-saml

Repository: ABFAB URN Parameters (Section Section 11.2)

Index Value: Sub-parameters MUST be specified in UTF-8 using standard URI encoding where necessary.

12. Acknowledgements

The authors would like to acknowledge the OASIS Security Services (SAML) Technical Committee, and Scott Cantor in particular, for their help with the SAML-related material.

The authors would also like to acknowledge the collaboration of Jim Schaad, Leif Johansson, Klaas Wierenga, Stephen Farell, Gabriel Lopez, and Rafael Marin, who have provided valuable comments on this document.

13. References

13.1. 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>.

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, DOI 10.17487/RFC2865, June 2000, <http://www.rfc-editor.org/info/rfc2865>.

[RFC3579] Aboba, B. and P. Calhoun, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", RFC 3579, DOI 10.17487/RFC3579, September 2003, <http://www.rfc-editor.org/info/rfc3579>.

[RFC6614] Winter, S., McCauley, M., Venaas, S., and K. Wierenga, "Transport Layer Security (TLS) Encryption for RADIUS", RFC 6614, DOI 10.17487/RFC6614, May 2012, <http://www.rfc-editor.org/info/rfc6614>.

[RFC6929] DeKok, A. and A. Lior, "Remote Authentication Dial In User Service (RADIUS) Protocol Extensions", RFC 6929, DOI 10.17487/RFC6929, April 2013, <http://www.rfc-editor.org/info/rfc6929>.

[RFC3575] Aboba, B., "IANA Considerations for RADIUS (Remote Authentication Dial In User Service)", RFC 3575, DOI 10.17487/RFC3575, July 2003, <http://www.rfc-editor.org/info/rfc3575>.

[RFC7542] DeKok, A., "The Network Access Identifier", RFC 7542, DOI 10.17487/RFC7542, May 2015, <http://www.rfc-editor.org/info/rfc7542>.

[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.

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

13.2. Informative References

[RFC3553] Mealling, M., Masinter, L., Hardie, T., and G. Klyne, "An IETF URN Sub-namespace for Registered Protocol Parameters", BCP 73, RFC 3553, DOI 10.17487/RFC3553, June 2003, <http://www.rfc-editor.org/info/rfc3553>.

[RFC6733] Fajardo, V., Ed., Arkko, J., Loughney, J., and G. Zorn, Ed., "Diameter Base Protocol", RFC 6733, DOI 10.17487/RFC6733, October 2012, <http://www.rfc-editor.org/info/rfc6733>.

[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>.

[RFC7055] Hartman, S., Ed. and J. Howlett, "A GSS-API Mechanism for the Extensible Authentication Protocol", RFC 7055, DOI 10.17487/RFC7055, December 2013, <http://www.rfc-editor.org/info/rfc7055>.

[RFC7499] Perez-Mendez, A., Ed., Marin-Lopez, R., Pereniguez-Garcia, F., Lopez-Millan, G., Lopez, D., and A. DeKok, "Support of Fragmentation of RADIUS Packets", RFC 7499, DOI 10.17487/RFC7499, April 2015, <http://www.rfc-editor.org/info/rfc7499>.

[I-D.ietf-abfab-arch]
Howlett, J., Hartman, S., Tschofenig, H., Lear, E., and J. Schaad, "Application Bridging for Federated Access Beyond Web (ABFAB) Architecture", draft-ietf-abfab-arch-13 (work in progress), July 2014.

[I-D.ietf-radext-bigger-packets]
Hartman, S., "Larger Packets for RADIUS over TCP", draft-ietf-radext-bigger-packets-05 (work in progress), December 2015.

[W3C.REC-xmlschema-1]
Thompson, H., Beech, D., Maloney, M., and N. Mendelsohn, "XML Schema Part 1: Structures", W3C REC-xmlschema-1, May 2001, <http://www.w3.org/TR/xmlschema-1/>.

Appendix A. XML Schema

The following schema formally defines the "urn:ietf:params:xml:ns:abfab" namespace used in this document, in conformance with [W3C.REC-xmlschema-1] While XML validation is optional, the schema that follows is the normative definition of the constructs it defines. Where the schema differs from any prose in this specification, the schema takes precedence.

```
        <schema
          targetNamespace="urn:ietf:params:xml:ns:abfab"
          xmlns="http://www.w3.org/2001/XMLSchema"
          xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
          xmlns:abfab="urn:ietf:params:xml:ns:abfab"
          elementFormDefault="unqualified"
          attributeFormDefault="unqualified"
          blockDefault="substitution"
          version="1.0">

          <import namespace="urn:oasis:names:tc:SAML:2.0:metadata"/>

          <complexType name="RADIUSIDPDescriptorType">
            <complexContent>
              <extension base="md:RoleDescriptorType">
                <sequence>
                  <element ref="abfab:RADIUSIDPService" minOccurs="0" maxOccurs=
"unbounded"/>
                  <element ref="abfab:RADIUSRealm" minOccurs="0" maxOccurs="unbo
unded"/>
                </sequence>
              </extension>
            </complexContent>
          </complexType>
          <element name="RADIUSIDPService" type="md:EndpointType"/>
          <element name="RADIUSRealm" type="string"/>

          <complexType name="RADIUSRPDescriptorType">
            <complexContent>
              <extension base="md:RoleDescriptorType">
                <sequence>
                  <element ref="md:RADIUSRPService" minOccurs="0" maxOccurs="unb
ounded"/>
                  <element ref="md:RADIUSNasIpAddress" minOccurs="0" maxOccurs="
unbounded"/>
                  <element ref="md:RADIUSNasIdentifier" minOccurs="0" maxOccurs=
"unbounded"/>
                  <element ref="md:RADIUSGssEapName" minOccurs="0" maxOccurs="un
bounded"/>
                </sequence>
              </extension>
            </complexContent>
          </complexType>
          <element name="RADIUSRPService" type="md:EndpointType"/>
          <element name="RADIUSNasIpAddress" type="string"/>
          <element name="RADIUSNasIdentifier" type="string"/>
          <element name="RADIUSGssEapName" type="string"/>

        </schema>
```

Authors' Addresses

Josh Howlett
Janet
Lumen House, Library Avenue, Harwell
Oxford OX11 0SG
UK

Phone: +44 1235 822363
EMail: Josh.Howlett@ja.net

Sam Hartman
Painless Security

EMail: hartmans-ietf@mit.edu

Alejandro Perez-Mendez (editor)
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia 30100
Spain

Phone: +34 868 88 46 44
EMail: alex@um.es

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 14, 2013

S. Hartman, Ed.
Painless Security
J. Howlett
JANET

August 13, 2012

A GSS-API Mechanism for the Extensible Authentication Protocol
draft-ietf-abfab-gss-eap-09.txt

Abstract

This document defines protocols, procedures, and conventions to be employed by peers implementing the Generic Security Service Application Program Interface (GSS-API) when using the Extensible Authentication Protocol mechanism. Through the GS2 family of mechanisms defined in RFC 5801, these protocols also define how Simple Authentication and Security Layer (SASL, RFC 4422) applications use the Extensible Authentication Protocol.

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 February 14, 2013.

Copyright Notice

Table of Contents

1. Introduction

ABFAB [I-D.ietf-abfab-arch] describes an architecture for providing federated access management to applications using the Generic Security Services Application Programming Interface (GSS-API) [RFC2743] and Simple Authentication and Security Layers (SASL) [RFC4422]. This specification provides the core mechanism for bringing federated authentication to these applications.

The Extensible Authentication Protocol (EAP) [RFC3748] defines a framework for authenticating a network access client and server in order to gain access to a network. A variety of different EAP methods are in wide use; one of EAP's strengths is that for most types of credentials in common use, there is an EAP method that permits the credential to be used.

EAP is often used in conjunction with a backend Authentication , Authorization and Accounting (AAA) server via RADIUS [RFC3579] or Diameter [RFC4072]. In this mode, the Network Access Server (NAS) simply tunnels EAP packets over the backend authentication protocol to a home EAP/AAA server for the client. After EAP succeeds, the backend authentication protocol is used to communicate key material to the NAS. In this mode, the NAS need not be aware of or have any specific support for the EAP method used between the client and the home EAP server. The client and EAP server share a credential that depends on the EAP method; the NAS and AAA server share a credential based on the backend authentication protocol in use. The backend authentication server acts as a trusted third party enabling network access even though the client and NAS may not actually share any common authentication methods. As described in the architecture document, using AAA proxies, this mode can be extended beyond one organization to provide federated authentication for network access.

The GSS-API provides a generic framework for applications to use security services including authentication and per-message data security. Between protocols that support GSS-API directly or protocols that support SASL [RFC4422], many application protocols can use GSS-API for security services. However, with the exception of Kerberos [RFC4121], few GSS-API mechanisms are in wide use on the Internet. While GSS-API permits an application to be written independent of the specific GSS-API mechanism in use, there is no facility to separate the server from the implementation of the mechanism as there is with EAP and backend authentication servers.

The goal of this specification is to combine GSS-API's support for application protocols with EAP/AAA's support for common credential types and for authenticating to a server without requiring that server to specifically support the authentication method in use. In

addition, this specification supports the architectural goal of transporting attributes about subjects to relying parties. Together this combination will provide federated authentication and authorization for GSS-API applications. This specification meets the applicability requirements for EAP to application authentication [I-D.ietf-abfab-eapapplicability].

This mechanism is a GSS-API mechanism that encapsulates an EAP conversation. From the perspective of RFC 3748, this specification defines a new lower-layer protocol for EAP. From the perspective of the application, this specification defines a new GSS-API mechanism.

Section 1.3 of [RFC5247] outlines the typical conversation between EAP peers where an EAP key is derived:

- Phase 0: Discovery
- Phase 1: Authentication
- 1a: EAP authentication
- 1b: AAA Key Transport (optional)
- Phase 2: Secure Association Protocol
- 2a: Unicast Secure Association
- 2b: Multicast Secure Association (optional)

1.1. Discovery

GSS-API peers discover each other and discover support for GSS-API in an application-dependent mechanism. SASL [RFC4422] describes how discovery of a particular SASL mechanism such as a GSS-API mechanism is conducted. The Simple and Protected Negotiation mechanism (SPNEGO) [RFC4178] provides another approach for discovering what GSS-API mechanisms are available. The specific approach used for discovery is out of scope for this mechanism.

1.2. Authentication

GSS-API authenticates a party called the GSS-API initiator to the GSS-API acceptor, optionally providing authentication of the acceptor to the initiator. Authentication starts with a mechanism-specific message called a context token sent from the initiator to the acceptor. The acceptor responds, followed by the initiator, and so on until authentication succeeds or fails. GSS-API context tokens are reliably delivered by the application using GSS-API. The

application is responsible for in-order delivery and retransmission.

EAP authenticates a party called a peer to a party called the EAP server. A third party called an EAP passthrough authenticator may decapsulate EAP messages from a lower layer and reencapsulate them into an AAA protocol. The term EAP authenticator refers to whichever of the passthrough authenticator or EAP server receives the lower-layer EAP packets. The first EAP message travels from the authenticator to the peer; a GSS-API message is sent from the initiator to acceptor to prompt the authenticator to send the first EAP message. The EAP peer maps onto the GSS-API initiator. The role of the GSS-API acceptor is split between the EAP authenticator and the EAP server. When these two entities are combined, the division resembles GSS-API acceptors in other mechanisms. When a more typical deployment is used and there is a passthrough authenticator, most context establishment takes place on the EAP server and per-message operations take place on the authenticator. EAP messages from the peer to the authenticator are called responses; messages from the authenticator to the peer are called requests.

Because GSS-API applications provide guaranteed delivery of context tokens, the EAP retransmission timeout MUST be infinite and the EAP layer MUST NOT retransmit a message.

This specification permits a GSS-API acceptor to hand-off the processing of the EAP packets to a remote EAP server by using AAA protocols such as RADIUS, RadSec or Diameter. In this case, the GSS-API acceptor acts as an EAP pass-through authenticator. The pass-through authenticator is responsible for retransmitting AAA messages if a response is not received from the AAA server. If a response cannot be recieved, then the authenticator generates an error at the GSS-API level. If EAP authentication is successful, and where the chosen EAP method supports key derivation, EAP keying material may also be derived. If an AAA protocol is used, this can also be used to replicate the EAP Key from the EAP server to the EAP authenticator.

See Section 5 for details of the authentication exchange.

1.3. Secure Association Protocol

After authentication succeeds, GSS-API provides a number of per-message security services that can be used:

GSS_Wrap() provides integrity and optional confidentiality for a message.

GSS_GetMIC() provides integrity protection for data sent independently of the GSS-API

GSS_Pseudo_random [RFC4401] provides key derivation functionality.

These services perform a function similar to secure association protocols in network access. Like secure association protocols, these services need to be performed near the authenticator/acceptor even when a AAA protocol is used to separate the authenticator from the EAP server. The key used for these per-message services is derived from the EAP key; the EAP peer and authenticator derive this key as a result of a successful EAP authentication. In the case that the EAP authenticator is acting as a pass-through it obtains it via the AAA protocol. See Section 6 for details.

2. Requirements notation

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. EAP Channel Binding and Naming

EAP authenticates a user to a realm. The peer knows that it has exchanged authentication with an EAP server in a given realm. Today, the peer does not typically know which NAS it is talking to securely. That is often fine for network access. However privileges to delegate to a chat server seem very different than privileges for a file server or trading site. Also, an EAP peer knows the identity of the home realm, but perhaps not even the visited realm.

In contrast, GSS-API takes a name for both the initiator and acceptor as inputs to the authentication process. When mutual authentication is used, both parties are authenticated. The granularity of these names is somewhat mechanism dependent. In the case of the Kerberos mechanism, the acceptor name typically identifies both the protocol in use (such as IMAP) and the specific instance of the service being connected to. The acceptor name almost always identifies the administrative domain providing service.

An EAP GSS-API mechanism needs to provide GSS-API naming semantics in order to work with existing GSS-API applications. EAP channel binding [I-D.ietf-emu-chbind] is used to provide GSS-API naming semantics. Channel binding sends a set of attributes from the peer to the EAP server either as part of the EAP conversation or as part of a secure association protocol. In addition, attributes are sent in the backend authentication protocol from the authenticator to the EAP server. The EAP server confirms the consistency of these attributes. Confirming attribute consistency also involves checking consistency against a local policy database as discussed in Section 3.5. In particular, the peer sends the name of the acceptor it is authenticating to as part of channel binding. The acceptor sends its full name as part of the backend authentication protocol. The EAP server confirms consistency of the names.

EAP channel binding is easily confused with a facility in GSS-API also called channel binding. GSS-API channel binding provides protection against man-in-the-middle attacks when GSS-API is used as authentication inside some tunnel; it is similar to a facility called cryptographic binding in EAP. See [RFC5056] for a discussion of the differences between these two facilities and Section 6.1 for how GSS-API channel binding is handled in this mechanism.

3.1. Mechanism Name Format

Before discussing how the initiator and acceptor names are validated in the AAA infrastructure, it is necessary to discuss what composes a name for an EAP GSS-API mechanism. GSS-API permits several types of generic names to be imported using GSS_Import_name(). Once a

mechanism is chosen, these names are converted into a mechanism-specific name called a "Mechanism Name". Note that a Mechanism Name is the name of an initiator or acceptor, not of a GSS-API mechanism. This section first discusses the mechanism name form and then discusses what name forms are supported.

The string representation of the GSS-EAP mechanism name has the following ABNF [RFC5234] representation:

```
char-normal = %x00-2E/%x30-3F/%x41-5B/%x5D-FF
char-escaped = "\" %x2F / "\" %x40 / "\" %x5C
name-char = char-normal / char-escaped
name-string = 1*name-char
user-or-service = name-string
host = [name-string]
realm = name-string
service-specific = name-string
service-specifics = service-specific 0*("/" service-specifics)
name = user-or-service ["/" host [ "/" service-specifics]] [ "@"
       realm ]
```

Special characters appearing in a name can be backslash escaped to avoid their special meanings. For example "\\" represents a literal backslash. This escaping mechanism is a property of the string representation; if the components of a name are transported in some mechanism that will keep them separate without backslash escaping, then backslash SHOULD have no special meaning.

The user-or-service component is similar to the portion of a network access identifier (NAI) before the ’@’ symbol for initiator names and the service name from the registry of GSS-API host-based services in the case of acceptor names [GSS-IANA]. The NAI specification provides rules for encoding and string preparation in order to support internationalization of NAIs; implementations of this mechanism MUST NOT prepare the user-or-service according to these rules; see Section 3.2 for internationalization of this mechanism. The host portion is empty for initiators and typically contains the domain name of the system on which an acceptor service is running. Some services MAY require additional parameters to distinguish the entity being authenticated against. Such parameters are encoded in the service-specifics portion of the name. The EAP server MUST reject authentication of any acceptor name that has a non-empty service-specifics component unless the EAP server understands the service-specifics and authenticates them. The interpretation of the service-specifics is scoped by the user-or-service portion. The realm is similar to the the realm portion of a NAI for initiator names; again the NAI specification’s internationalization rules MUST NOT be applied to the realm. The realm is the administrative realm

of a service for an acceptor name.

The string representation of this name form is designed to be generally compatible with the string representation of Kerberos names defined in [RFC1964].

The GSS_C_NT_USER_NAME form represents the name of an individual user. From the standpoint of this mechanism it may take the form either of an undecorated user name or a name semantically similar to a network access identifier (NAI) [RFC4282]. The name is split at the first at-sign ('@') into the part preceeding the realm which is the user-or-service portion of the mechanism name and the realm portion which is the realm portion of the mechanism name.

The GSS_C_NT_HOSTBASED_SERVICE name form represents a service running on a host; it is textually represented as "service@host". This name form is required by most SASL profiles and is used by many existing applications that use the Kerberos GSS-API mechanism. While support for this name form is critical, it presents an interesting challenge in terms of EAP channel binding. Consider a case where the server communicates with a "server proxy," or a AAA server near the server. That server proxy communicates with the EAP server. The EAP server and server proxy are in different administrative realms. The server proxy is in a position to verify that the request comes from the indicated host. However the EAP server cannot make this determination directly. So, the EAP server needs to determine whether to trust the server proxy to verify the host portion of the acceptor name. This trust decision depends both on the host name and the realm of the server proxy. In effect, the EAP server decides whether to trust that the realm of the server proxy is the right realm for the given hostname and then makes a trust decision about the server proxy itself. The same problem appears in Kerberos: there, clients decide what Kerberos realm to trust for a given hostname. The service portion of this name is imported into the user-or-service portion of the mechanism name; the host portion is imported into the host portion of the mechanism name. The realm portion is empty. However, authentication will typically fail unless some AAA component indicates the realm to the EAP server. If the application server knows its realm, then it should be indicated in the outgoing AAA request. Otherwise, a proxy SHOULD add the realm. An alternate form of this name type MAY be used on acceptors; in this case the name form is "service" with no host component. This is imported with the service as user-or-service and an empty host and realm portion. This form is useful when a service is unsure which name an initiator knows it by.

If the null name type or the GSS_EAP_NT_EAP_NAME (OID 1.3.6.1.5.5.15.2.1) (see Section 7.1) is imported, then the string

representation above should be directly imported. Mechanisms MAY support the GSS_KRB5_NT_KRB5_PRINCIPAL_NAME name form with the OID {iso(1) member-body(2) United States(840) mit(113554) infosys(1) gssapi(2) krb5(2) krb5_name(1)}. In many circumstances, Kerberos GSS-API mechanism names will behave as expected when used with the GSS-API EAP mechanism, but there are some differences that may cause some confusion. If an implementation does support importing Kerberos names it SHOULD fail the import if the Kerberos name is not syntactically a valid GSS-API EAP mechanism name as defined in this section.

3.2. Internationalization of Names

For the most part, GSS-EAP names are transported in other protocols; those protocols define the internationalization semantics. For example, if an AAA server wishes to communicate the user-or-service portion of the initiator name to an acceptor, it does so using existing mechanisms in the AAA protocol. Existing internationalization rules are applied. Similarly, within an application, existing specifications such as [RFC5178] define the encoding of names that are imported and displayed with the GSS-API.

This mechanism does introduce a few cases where name components are sent. In these cases the encoding of the string is UTF-8. Senders SHOULD NOT normalize or map strings before sending. These strings include RADIUS attributes introduced in Section 3.4.

When comparing the host portion of a GSS-EAP acceptor name supplied in EAP channel binding by a peer to that supplied by an acceptor, EAP servers SHOULD prepare the host portion according to [RFC5891] prior to comparison. Applications MAY prepare domain names prior to importing them into this mechanism.

3.3. Exported Mechanism Names

GSS-API provides the GSS_Export_name call. This call can be used to export the binary representation of a name. This name form can be stored on access control lists for binary comparison.

The exported name token MUST use the format described in section 3.2 of RFC 2743. The mechanism specific portion of this name token is the string format of the mechanism name described in Section 3.1.

RFC 2744 [RFC2744] places the requirement that the result of importing a name, canonicalizing it to a Mechanism Name and then exporting it needs to be the same as importing that name, obtaining credentials for that principal, initiating a context with those credentials and exporting the name on the acceptor. In practice, GSS

mechanisms often, but not always meet this requirement. For names expected to be used as initiator names, this requirement is met. However, permitting empty host and realm components when importing hostbased services may make it possible for an imported name to differ from the exported name actually used. Other mechanisms such as Kerberos have similar situations where imported and exported names may differ.

3.4. Acceptor Name RADIUS AVP

See Section 7.4 for registrations of RADIUS attribute types to carry the acceptor service name. All the attribute types registered in that section are strings. See Section 3.1 for details of the values in a name.

If RADIUS is used as an AAA transport, the acceptor MUST send the acceptor name in these attribute types. That is, the acceptor decomposes its name and sends any non-empty portion as a RADIUS attribute. With the exception of the service-specifics portion of the name, the backslash escaping mechanism is not used in RADIUS attributes; backslash has no special meaning. In the service-specifics portion, a literal "/" separates components. In this one attribute, "\/" indicates a slash character that does not separate components and "\\" indicates a literal backslash character.

The initiator MUST require that the EAP method in use support channel binding and MUST send the acceptor name as part of the channel binding data. The client MUST NOT indicate mutual authentication in the result of GSS_Init_Sec_Context unless all name elements that the client supplied are in a successful channel binding response. For example, if the client supplied a hostname in channel binding data, the hostname MUST be in a successful channel binding response.

If an empty target name is supplied to GSS_Init_Sec_Context, the initiator MUST fail context establishment unless the acceptor supplies the acceptor name response (Section 5.4.3). If a null target name is supplied, the initiator MUST use this response to populate EAP channel bindings.

3.5. Proxy Verification of Acceptor Name

Proxies may play a role in verification of the acceptor identity. For example, an AAA proxy near the acceptor may be in a position to verify the acceptor hostname, while the EAP server is likely to be too distant to reliably verify this on its own.

The EAP server or some proxy trusted by the EAP server is likely to be in a position to verify the acceptor realm. In effect, this proxy

is confirming that the right AAA credential is used for the claimed realm and thus that the acceptor is in the organization it claims to be part of. This proxy is also typically trusted by the EAP server to make sure that the hostname claimed by the acceptor is a reasonable hostname for the realm of the acceptor.

A proxy close to the EAP server is unlikely to be in a position to confirm that the acceptor is claiming the correct hostname. Instead this is typically delegated to a proxy near the acceptor. That proxy is typically expected to verify the acceptor hostname and to verify the appropriate AAA credential for that host is used. Such a proxy may insert the acceptor realm if it is absent, permitting realm configuration to be at the proxy boundary rather than on acceptors.

Ultimately specific proxy behavior is a matter for deployment. The EAP server MUST assure that the appropriate validation has been done before including acceptor name attributes in a successful channel binding response. If the acceptor service is included the EAP server asserts that the service is plausible for the acceptor. If the acceptor hostname is included the EAP server asserts that the acceptor hostname is verified. If the realm is included the EAP server asserts that the realm has been verified, and if the hostname was also included, that the realm and hostname are consistent. Part of this verification MAY be delegated to proxies, but the EAP server configuration MUST guarantee that the combination of proxies meets these requirements. Typically such delegation will involve business or operational measures such as cross-organizational agreements as well as technical measures.

It is likely that future technical work will be needed to communicate what verification has been done by proxies along the path. Such technical measures will not release the EAP server from its responsibility to decide whether proxies on the path should be trusted to perform checks delegated to them. However technical measures could prevent misconfigurations and help to support diverse environments.

4. Selection of EAP Method

EAP does not provide a facility for an EAP server to advertise what methods are available to a peer. Instead, a server starts with its preferred method selection. If the peer does not accept that method, the peer sends a NAK response containing the list of methods supported by the client.

Providing multiple facilities to negotiate which security mechanism to use is undesirable. Section 7.3 of [RFC4462]describes the problem referencing the SSH key exchange negotiation and the SPNEGO GSS-API mechanism. If a client preferred an EAP method A, a non-EAP authentication mechanism B, and then an EAP method C, then the client would have to commit to using EAP before learning whether A is actually supported. Such a client might end up using C when B is available.

The standard solution to this problem is to perform all the negotiation at one layer. In this case, rather than defining a single GSS-API mechanism, a family of mechanisms should be defined. Each mechanism corresponds to an EAP method. The EAP method type should be part of the GSS-API OID. Then, a GSS-API rather than EAP facility can be used for negotiation.

Unfortunately, using a family of mechanisms has a number of problems. First, GSS-API assumes that both the initiator and acceptor know the entire set of mechanisms that are available. Some negotiation mechanisms are driven by the client; others are driven by the server. With EAP GSS-API, the acceptor does not know what methods the EAP server implements. The EAP server that is used depends on the identity of the client. The best solution so far is to accept the disadvantages of multi-layer negotiation and commit to using EAP GSS-API before a specific EAP method. This has two main disadvantages. First, authentication may fail when other methods might allow authentication to succeed. Second, a non-optimal security mechanism may be chosen.

5. Context Tokens

All context establishment tokens emitted by the EAP mechanism SHALL have the framing described in section 3.1 of [RFC2743], as illustrated by the following pseudo-ASN.1 structures:

```
GSS-API DEFINITIONS ::=
        BEGIN

        MechType ::= OBJECT IDENTIFIER
        -- representing EAP mechanism
        GSSAPI-Token ::=
        -- option indication (delegation, etc.) indicated within
        -- mechanism-specific token
        [APPLICATION 0] IMPLICIT SEQUENCE {
                thisMech MechType,
                innerToken ANY DEFINED BY thisMech
                   -- contents mechanism-specific
                   -- ASN.1 structure not required
                }
        END
```

The innerToken field starts with a 16-bit network byte order token type identifier. The remainder of the innerToken field is a set of type-length-value subtokens. The following figure describes the structure of the inner token:

Octet Position	Description
0..1	token ID
2..5	first subtoken type
6..9	length of first subtoken
10..10+n-1	first subtoken body
10+n..10+n+3	second subtoken type

The inner token continues with length, second subtoken body, and so forth. If a subtoken type is present, its length and body MUST be present.

Structure of Inner Token

The length is a four-octet length of the subtoken body in network

byte order. The length does not include the length of the type field or the length field; the length only covers the body.

Tokens from the initiator to acceptor use an inner token type with ID 06 01; tokens from acceptor to initiator use an inner token type with ID 06 02. These token types are registered in the registry of RFC 4121 token types; see Section 7.2.

See Section 5.7 for the encoding of a complete token. The following sections discuss how mechanism OIDs are chosen and the state machine that defines what subtokens are permitted at each point in the context establishment process.

5.1. Mechanisms and Encryption Types

This mechanism family uses the security services of the Kerberos cryptographic framework [RFC3961]. The root of the OID ARC for mechanisms described in this document is 1.3.6.1.5.5.15.1.1; a Kerberos encryption type number [RFC3961] is appended to that root OID to form a mechanism OID. As such, a particular encryption type needs to be chosen. By convention, there is a single object identifier arc for the EAP family of GSS-API mechanisms. A specific mechanism is chosen by adding the numeric Kerberos encryption type number to the root of this arc. However, in order to register the SASL name, the specific usage with a given encryption type needs to be registered. This document defines the EAP-AES128 GSS-API mechanism.

5.2. Processing received tokens

Whenever a context token is received, the receiver performs the following checks. First the receiver confirms the object identifier is that of the mechanism being used. The receiver confirms that the token type corresponds to the role of the peer: acceptors will only process initiator tokens and initiators will only process acceptor tokens.

Implementations of this mechanism maintain a state machine for the context establishment process. Both the initiator and acceptor start out in the initial state; see Section 5.4 for a description of this state. Associated with each state are a set of subtoken types that are processed in that state and rules for processing these subtoken types. The reciever examines the subtokens in order, processing any that are appropriate for the current state. Unknown subtokens or subtokens that are not expected in the current state are ignored if their critical bit (see below) is clear.

A state may have a set of required subtoken types. If a subtoken

type is required by the current state but no subtoken of that type is present, then the context establishment MUST fail.

The most-significant bit (0x80000000) in a subtoken type is the critical bit. If a subtoken with this bit set in the type is received, the receiver MUST fail context establishment unless the subtoken is understood and processed for the current state.

The subtoken type MUST be unique within a given token.

5.3. Error Subtokens

The acceptor may always end the exchange by generating an error subtoken. The error subtoken has the following format:

Pos	Description
0..3	0x80 00 00 01
4..7	length of error token
8..11	major status from RFC 2744 as 32-bit network byte order
12..15	GSS EAP error code as 32-bit network byte order; see Section 7.6

Initiators MUST ignore octets beyond the GSS EAP error code for future extensibility. As indicated, the error token is always marked critical.

5.4. Initial State

Both the acceptor and initiator start the context establishment process in the initial state.

The initiator sends a token to the acceptor. It MAY be empty; no subtokens are required in this state. Alternatively the initiator MAY include a vendor ID subtoken or an acceptor name request subtoken.

The acceptor responds to this message. It MAY include an acceptor name response subtoken. It MUST include a first eap request; this is an EAP request/identity message (see Section 5.5.1 for the format of this subtoken).

The initiator and acceptor then transition to authenticate state.

5.4.1. Vendor Subtoken

The vendor ID token has type 0x0000000B and the following structure:

Pos	Description
0..3	0x0000000B
4..7	length of vendor token
8..8+length	Vendor ID string

The vendor ID string is an UTF-8 string describing the vendor of this implementation. This string is unstructured and for debugging purposes only.

5.4.2. Acceptor Name Request

The acceptor name request token is sent from the initiator to the acceptor indicating that the initiator wishes a particular acceptor name. This is similar to TLS Server Name Indication [RFC6066] which permits a client to indicate which one of a number of virtual services to contact. The structure is as follows:

Pos	Description
0..3	0x00000002
4..7	Length of subtoken
8..n	string form of acceptor name

It is likely that channel binding and thus authentication will fail if the acceptor does not choose a name that is a superset of this name. That is, if a hostname is sent, the acceptor needs to be willing to accept this hosntame.

5.4.3. Acceptor Name Response

The acceptor name response subtoken indicates what acceptor name is used. This is useful for example if the initiator supplied no target name to context initialization. This allows the initiator to learn the acceptor name. EAP channel bindings will provide confirmation that the acceptor is accurately naming itself.

this token is sent from the acceptor to initiator. In the Initial state, this token would typically be sent if the acceptor name request is absent, because if the initiator already sent an acceptor name then the initiator knows what acceptor it wishes to contact. This subtoken is also sent in extensions state Section 5.6 so the initiator can protect against a man-in-the-middle modifying the acceptor name request subtoken.

Pos	Description
0..3	0x00000003
4..7	Length of subtoken
8..n	string form of acceptor name

5.5. Authenticate State

In this state, the acceptor sends EAP requests to the initiator and the initiator generates EAP responses. The goal of the state is to perform a successful EAP authentication. Since the acceptor sends an identity request at the end of the initial state, the first half-round-trip in this state is a response to that request from the initiator.

The EAP conversation can end in a number of ways:

- If the EAP state machine generates an EAP success message, then the EAP authenticator believes the authentication is successful. The Acceptor MUST confirm that a key has been derived (Section 7.10 of [RFC3748]). The acceptor MUST confirm that this success indication is consistent with any protected result indication for combined authenticators and with AAA indication of success for pass-through authenticators. If any of these checks fail, the acceptor MUST send an error subtoken and fail the context establishment. If these checks succeed the acceptor sends the success message using the EAP Request subtoken type and transitions to Extensions state. If the initiator receives an EAP Success message, it confirms that a key has been derived and that the EAP success is consistent with any protected result indication. If so, it transitions to Extensions state. Otherwise, it returns an error to the caller of GSS_Init_Sec_context without producing an output token.

- If the acceptor receives an EAP failure, then the acceptor sends this in the Eap Request subtoken type. If the initiator receives

an EAP Failure, it returns GSS failure.

- o If there is some other error, the acceptor MAY return an error subtoken.

5.5.1. EAP Request Subtoken

The EAP Request subtoken is sent from the acceptor to the initiator. This subtoken is always critical and is REQUIRED in the authentication state.

Pos	Description
0..3	0x80000005
4..7	Length of EAP message
8..8+length	EAP message

5.5.2. EAP Response Subtoken

This subtoken is REQUIRED in authentication state messages from the initiator to the acceptor. It is always critical.

Pos	Description
0..3	0x80000004
4..7	Length of EAP message
8..8+length	EAP message

5.6. Extension State

After EAP success, the initiator sends a token to the acceptor including additional subtokens that negotiate optional features or provide GSS-API channel binding (see Section 6.1). The acceptor then responds with a token to the initiator. When the acceptor produces its final token it returns GSS_S_COMPLETE; when the initiator consumes this token it returns GSS_S_COMPLETE if no errors are detected.

The acceptor SHOULD send an acceptor name response (Section 5.4.3) so that the initiator can get a copy of the acceptor name protected by

the MIC subtoken.

Both the initiator and acceptor MUST include and verify a MIC subtoken to protect the extensions exchange.

5.6.1. Flags Subtoken

This token is sent to convey initiator flags to the acceptor. The flags are sent as a 32-bit integer in network byte order. The only flag defined so far is GSS_C_MUTUAL_FLAG, indicating that the initiator successfully performed mutual authentication of the acceptor. This flag is communicated to the acceptor because some protocols [RFC4462] require the acceptor to know whether the initiator has confirmed its identity. This flag has the value 0x2 to be consistent with RFC 2744.

Pos	Description
0..3	0x0000000C
4..7	length of flags token
8..11	flags

Initiators MUST send 4 octets of flags. Acceptors MUST ignore flag octets beyond the first 4 and MUST ignore flag bits other than GSS_C_MUTUAL_FLAG. Initiators MUST send undefined flag bits as zero.

5.6.2. GSS Channel Bindings Subtoken

This token is always critical when sent. It is sent from the initiator to the acceptor. The contents of this token are an RFC 3961 get_mic token of the application data from the GSS channel bindings structure passed into the context establishment call.

Pos	Description
0..3	0x80000006
4..7	length of token
8..8+length	get_mic of channel binding application data

Again, only the application data is sent in the channel binding. Any

initiator and acceptor addresses passed by an application into context establishment calls are ignored and not sent over the wire. The checksum type of the get_mic token SHOULD be the mandatory to implement checksum type of the Context Root Key (CRK.) The key to use is the CRK and the key usage is 60 (KEY_USAGE_GSSEAP_CHBIND_MIC). An acceptor MAY accept any MIC in the channel bindings subtoken if the channel bindings input to GSS_Accept_Sec_context is not provided. If the channel binding input to GSS_Accept_Sec_context is provided, the acceptor MUST return failure if the channel binding MIC in a received channel binding subtoken fails to verify.

The initiator MUST send this token if channel bindings including application data are passed into GSS_Init_Sec_context and MUST NOT send this token otherwise.

5.6.3. MIC Subtoken

This token MUST be the last subtoken in the tokens sent in Extensions state. This token is sent both by the initiator and acceptor.

Pos	Description
0..3	0x8000000D for initiator 0x8000000E for acceptor
4..7	Length of RFC 3961 MIC token
8..8+length	RFC 3961 result of get_mic

As with any call to get_mic, a token is produced as described in RFC 3961 using the CRK Section 6 as the key and the mandatory checksum type for the encryption type of the CRK as the checksum type. The key usage is 61 (KEY_USAGE_GSSEAP_ACCTOKEN_MIC) for the subtoken from the acceptor to the initiator and 62 (KEY_USAGE_GSSEAP_INITTOKEN_MIC) for the subtoken from the initiator to the acceptor. The input is as follows:

1. The DER-encoded object identifier of the mechanism in use; this value starts with 0x06 (the tag for object identifier). When encoded in an RFC 2743 context token, the object identifier is preceeded by the tag and length for [Application 0] SEQUENCE. This tag and the length of the overall token is not included; only the tag, length and value of the object identifier itself.

2. A 16-bit token type in network byte order of the RFC 4121 token identifier (0x0601 for initiator, 0x0602 for acceptor).

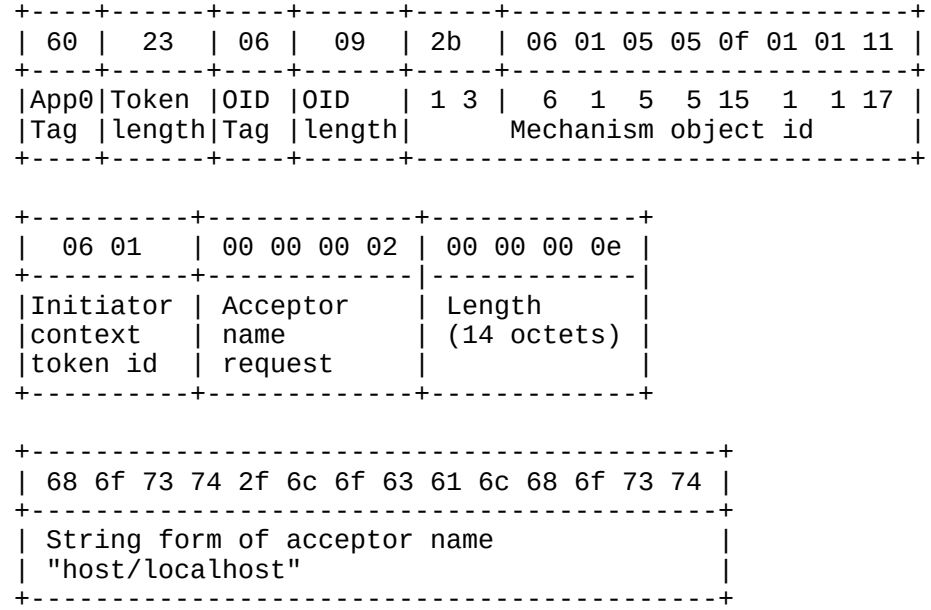
3. For each subtoken other than the MIC subtoken itself in the order the subtokens appear in the token:

   1. A four octet subtoken type in network byte order

   2. A four byte length in network byte order

   3. Length octets of value from that subtoken

5.7. Example Token



Example Initiator Token

5.8. Context Options

GSS-API provides a number of optional per-context services requested by flags on the call to GSS_Init_sec_context and indicated as outputs from both GSS_Init_sec_context and GSS_Accept_sec_context. This section describes how these services are handled. Which services the client selects in the call to GSS_Init_sec_context controls what EAP methods MAY be used by the client. Section 7.2 of RFC 3748 describes a set of security claims for EAP. As described below, the selected GSS options place requirements on security claims that MUST be met.

This GSS mechanism MUST only be used with EAP methods that provide dictionary attack resistance. Typically dictionary attack resistance is obtained by using an EAP tunnel method to tunnel an inner method in TLS.

The EAP method MUST support key derivation. Integrity, confidentiality, sequencing and replay detection MUST be indicated in the output of GSS_Init_Sec_Context and GSS_Accept_Sec_context regardless of which services are requested.

The PROT_READY service defined in Section 1.2.7 of [RFC2743] is never available with this mechanism. Implementations MUST NOT offer this flag or permit per-message security services to be used before context establishment.

The EAP method MUST support mutual authentication and channel binding. See Section 3.4 for details on what is required for successful mutual authentication. Regardless of whether mutual authentication is requested, the implementation MUST include channel bindings in the EAP authentication. If mutual authentication is requested and successful mutual authentication takes place as defined in Section 3.4, the initiator MUST send a flags subtoken Section 5.6.1 in Extensions state.

6. Acceptor Services

The context establishment process may be passed through to a EAP server via a backend authentication protocol. However after the EAP authentication succeeds, security services are provided directly by the acceptor.

This mechanism uses an RFC 3961 cryptographic key called the context root key (CRK). The CRK is derived from the GMSK (GSS-API MSK). The GMSK is the result of the random-to-key [RFC3961] operation of the encryption type of this mechanism consuming the appropriate number of bits from the EAP master session key. For example for aes128-cts-hmac-sha1-96, the random-to-key operation consumes 16 octets of key material; thus the first 16 bytes of the master session key are input to random-to-key to form the GMSK. If the MSK is too short, authentication MUST fail.

In the following, pseudo-random is the RFC 3961 pseudo-random operation for the encryption type of the GMSK and random-to-key is the RFC 3961 random-to-key operation for the enctype of the mechanism. The truncate function takes the initial l bits of its input. The goal in constructing a CRK is to call the pseudo-random function enough times to produce the right number of bits of output and discard any excess bits of output.

The CRK is derived from the GMSK using the following procedure

```
Tn = pseudo-random(GMSK, n || "rfc4121-gss-eap")
CRK = random-to-key(truncate(L, T0 || T1 || .. || Tn))
L = random-to-key input size
```

Where n is a 32-bit integer in network byte order starting at 0 and incremented to each call to the pseudo_random operation.

6.1. GSS-API Channel Binding

GSS-API channel binding [RFC5554] is a protected facility for exchanging a cryptographic name for an enclosing channel between the initiator and acceptor. The initiator sends channel binding data and the acceptor confirms that channel binding data has been checked.

The acceptor SHOULD accept any channel binding provided by the initiator if null channel bindings are passed into gss_accept_sec_context. Protocols such as HTTP Negotiate [RFC4559] depend on this behavior of some Kerberos implementations.

As discussed, the GSS channel bindings subtoken is sent in the extensions state.

6.2. Per-message security

The per-message tokens of section 4 of RFC 4121 are used. The CRK SHALL be treated as the initiator sub-session key, the acceptor sub-session key and the ticket session key.

6.3. Pseudo Random Function

The pseudo random function defined in [RFC4402] is used to provide GSS_Pseudo_Random functionality to applications.

7. Iana Considerations

This specification creates a number of IANA registries.

7.1. OID Registry

IANA is requested to create a registry of ABFAB object identifiers titled "Object Identifiers for Application Bridging for federated Access". The initial contents of the registry are specified below. The registration policy is IETF review or IESG approval. Early allocation is permitted. IANA is requested to update the reference for the root of this OID delegation to point to the newly created registry.

Prefix: iso.org.dod.internet.security.mechanisms.abfab (1.3.6.1.5.5.15)

Decimal	Name	Description	References
0	Reserved	Reserved	
1	mechanisms	A sub-arc containing ABFAB mechanisms	
2	nametypes	A sub-arc containing ABFAB GSS-API Name Types	

NOTE: the following mechanisms registry are the root of the OID for the mechanism in question. As discussed in Section 5.1 [draft-ietf-abbfab-gss-eap], a Kerberos encryption type number [RFC3961] is appended to the mechanism version OID below to form the OID of a specific mechanism.

Prefix: iso.org.dod.internet.security.mechanisms.abfab.mechanisms (1.3.6.1.5.5.15.1)

Decimal	Name	Description	References
0	Reserved	Reserved	
1	gss-eap-v1	The GSS-EAP mechanism	[this spec

Prefix: iso.org.dod.internet.security.mechanisms.abfab.nametypes (1.3.6.1.5.5.15.2)

Decimal	Name	Description	References
0	Reserved	Reserved	
1	GSS_EAP_NT_EAP_NAME		sect 3.1

7.2. RFC 4121 Token Identifiers

In the top level registry titled "Kerberos V GSS-API Mechanism Parameters," a sub-registry called "Kerberos GSS-API Token Type Identifiers" is created; the overall reference for this subregistry is section 4.1 of RFC 4121. The allocation procedure is expert review [RFC5226]. The expert's primary job is to make sure that token type identifiers are requested by an appropriate requester for the RFC 4121 mechanism in which they will be used and that multiple values are not allocated for the same purpose. For RFC 4121 and this mechanism, the expert is currently expected to make allocations for token identifiers from documents in the IETF stream; effectively for these mechanisms the expert currently confirms the allocation meets the requirements of the IETF review process.

The ID field is a hexadecimal token identifier specified in network byte order.

The initial registrations are as follows:

ID	Description	Reference
01 00	KRB_AP_REQ	RFC 4121 sect 4.1
02 00	KRB_AP_REP	RFC 4121 sect 4.1
03 00	KRB_ERROR	RFC 4121 sect 4.1
04 04	MIC tokens	RFC 4121 sect 4.2.6.1
05 04	wrap tokens	RFC 4121 sect 4.2.6.2
06 01	GSS-EAP initiator context token	Section 5
06 02	GSS EAP acceptor context token	Section 5

7.3. GSS EAP Subtoken Types

This document creates a top level registry called "The Extensible Authentication Protocol Mechanism for the Generic Security Services Application Programming Interface (GSS-EAP) Parameters". In any short form of that name, including any URI for this registry, it is important that the string GSS come before the string EAP; this will help to distinguish registries if EAP methods for performing GSS-API authentication are ever defined.

In this registry is a subregistry of subtoken types; identifiers are 32-bit integers; the upper bit (0x80000000) is reserved as a critical flag and should not be indicated in the registration. Assignments of GSS EAP subtoken types are made by expert review. The expert is expected to require a public specification of the subtoken similar in detail to registrations given in this document. The security of GSS-EAP depends on making sure that subtoken information has adequate protection and that the overall mechanism continues to be secure. Examining the security and architectural consistency of the proposed registration is the primary responsibility of the expert.

Type	Description	Reference
0x00000001	Error	Section 5.3
0x0000000B	Vendor	Section 5.4.1
0x00000002	Acceptor name request	Section 5.4.2
0x00000003	Acceptor name response	Section 5.4.3
0x00000005	EAP request	Section 5.5.1
0x00000004	EAP response	Section 5.5.2
0x0000000C	Flags	Section 5.6.1
0x00000006	GSS-API channel bindings	Section 5.6.2
0x0000000D	Initiator MIC	Section 5.6.3
0x0000000E	Acceptor MIC	Section 5.6.3

7.4. RADIUS Attribute Assignments

The following RADIUS attribute type values [RFC3575] are assigned. The assignment rules in section 10.3 of [I-D.ietf-radext-radius-extensions] may be used if that specification is approved when IANA actions for this specification are processed.

Name	Attribute	Description
GSS-Acceptor-Service-Name	TBD1	user-or-service portion of name
GSS-Acceptor-Host-Name	TBD2	host portion of name
GSS-Acceptor-Service-specifics	TBD3	service-specifics portion of name
GSS-Acceptor-Realm-Name	TBD4	Realm portion of name

7.5. Registration of the EAP-AES128 SASL Mechanisms

Subject: Registration of SASL mechanisms
EAP-AES128 and EAP-AES128-PLUS

SASL mechanism names: EAP-AES128 and EAP-AES128-PLUS

Security considerations: See RFC 5801 and draft-ietf-abfab-gss-eap

Published specification (recommended): draft-ietf-abfab-gss-eap

Person & email address to contact for further information:
Abfab Working Group abfab@ietf.org

Intended usage: common

Owner/Change controller: iesg@ietf.org

Note: This mechanism describes the GSS-EAP mechanism used with the aes128-cts-hmac-sha1-96 enctype. The GSS-API OID for this mechanism is 1.3.6.1.5.5.15.1.1.17
As described in RFC 5801 a PLUS varient of this mechanism is also required.

7.6. GSS EAP Errors

A new subregistry is created in the GSS EAP parameters registry titled "Error Codes". The error codes in this registry are unsigned 32-bit numbers. Values less than or equal to 127 are assigned by standards action. Values 128 through 255 are assigned with the specification required assignment policy. Values greater than 255 are reserved; updates to registration policy may make these values available for assignment and implementations MUST be prepared to

receive them.

This table provides the initial contents of the registry.

Value	Description
0	Reserved
1	Buffer is incorrect size
2	Incorrect mechanism OID
3	Token is corrupted
4	Token is truncated
5	Packet received by direction that sent it
6	Incorrect token type identifier
7	Unhandled critical subtoken received
8	Missing required subtoken
9	Duplicate subtoken type
10	Received unexpected subtoken for current state xxx
11	EAP did not produce a key
12	EAP key too short
13	Authentication rejected
14	AAA returned an unexpected message type
15	AAA response did not include EAP request
16	Generic AAA failure

7.7. GSS EAP Context Flags

A new sub-registry is created in the GSS EAP parameters registry. This registry holds registrations of flag bits sent in the flags subtoken Section 5.6.1. There are 32 flag bits available for registration represented as hexadecimal numbers from the most-

significant bit 0x80000000 to the least significant bit 0x1. The registration policy for this registry is IETF review or in exceptional cases IESG approval. The following table indicates initial registrations; all other values are available for assignment.

Flag	Name	Reference
0x2	GSS_C_MUTUAL_FLAG	Section 5.6.1

8. Security Considerations

RFC 3748 discusses security issues surrounding EAP. RFC 5247 discusses the security and requirements surrounding key management that leverages the AAA infrastructure. These documents are critical to the security analysis of this mechanism.

RFC 2743 discusses generic security considerations for the GSS-API. RFC 4121 discusses security issues surrounding the specific per-message services used in this mechanism.

As discussed in Section 4, this mechanism may introduce multiple layers of security negotiation into application protocols. Multiple layer negotiations are vulnerable to a bid-down attack when a mechanism negotiated at the outer layer is preferred to some but not all mechanisms negotiated at the inner layer; see section 7.3 of [RFC4462] for an example. One possible approach to mitigate this attack is to construct security policy such that the preference for all mechanisms negotiated in the inner layer falls between preferences for two outer layer mechanisms or falls at one end of the overall ranked preferences including both the inner and outer layer. Another approach is to only use this mechanism when it has specifically been selected for a given service. The second approach is likely to be common in practice because one common deployment will involve an EAP supplicant interacting with a user to select a given identity. Only when an identity is successfully chosen by the user will this mechanism be attempted.

EAP channel binding is used to give the GSS-API initiator confidence in the identity of the GSS-API acceptor. Thus, the security of this mechanism depends on the use and verification of EAP channel binding. Today EAP channel binding is in very limited deployment. If EAP channel binding is not used, then the system may be vulnerable to phishing attacks where a user is diverted from one service to another. If the EAP method in question supports mutual authentication then users can only be diverted between servers that are part of the same AAA infrastructure. For deployments where membership in the AAA infrastructure is limited, this may serve as a significant limitation on the value of phishing as an attack. For other deployments, use of EAP channel binding is critical to avoid phishing. These attacks are possible with EAP today although not typically with common GSS-API mechanisms. For this reason, implementations are required to implement and use EAP channel binding; see Section 3 for details.

The security considerations of EAP channel binding [I-D.ietf-emu-chbind] describe the security properties of channel binding. Two attacks are worth calling out here. First, when a

tunneled EAP method is used, it is critical that the channel binding be performed with an EAP server trusted by the peer. With existing EAP methods this typically requires validating the certificate of the server tunnel endpoint back to a trust anchor and confirming the name of the entity who is a subject of that certificate. EAP methods may suffer from bid-down attacks where an attacker can cause a peer to think that a particular EAP server does not support channel binding. This does not directly cause a problem because mutual authentication is only offered at the GSS-API level when channel binding to the server's identity is successful. However when an EAP method is not vulnerable to these bid-down attacks, additional protection is available. This mechanism will benefit significantly from new strong EAP methods such as [I-D.ietf-emu-eap-tunnel-method].

Every proxy in the AAA chain from the authenticator to the EAP server needs to be trusted to help verify channel bindings and to protect the integrity of key material. GSS-API applications may be built to assume a trust model where the acceptor is directly responsible for authentication. However, GSS-API is definitely used with trusted-third-party mechanisms such as Kerberos.

RADIUS does provide a weak form of hop-by-hop confidentiality of key material based on using MD5 as a stream cipher. Diameter can use TLS or IPsec but has no mandatory-to-implement confidentiality mechanism. Operationally, protecting key material as it is transported between the IDP and RP is critical to per-message security and verification of GSS-API channel binding [RFC5056]. Mechanisms such as RADIUS over TLS [I-D.ietf-radext-radsec] provide significantly better protection of key material than the base RADIUS specification.

9. Acknowledgements

Luke Howard, Jim Schaad, Alejandro Perez Mendez, Alexey Melnikov and Sujing Zhou provided valuable reviews of this document.

Rhys Smith provided the text for the OID registry section. Sam Hartman's work on this draft has been funded by JANET.

10. References

10.1. Normative References

[GSS-IANA] IANA, "GSS-API Service Name Registry", <http://www.iana.org/assignments/gssapi-service-names/gssapi-service-names.xhtml>.

[I-D.ietf-abfab-eapapplicability] Winter, S. and J. Salowey, "Update to the EAP Applicability Statement for ABFAB", draft-ietf-abfab-eapapplicability-00 (work in progress), July 2012.

[I-D.ietf-emu-chbind] Hartman, S., Clancy, T., and K. Hoeper, "Channel Binding Support for EAP Methods", draft-ietf-emu-chbind-16 (work in progress), May 2012.

[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.

[RFC3575] Aboba, B., "IANA Considerations for RADIUS (Remote Authentication Dial In User Service)", RFC 3575, July 2003.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[RFC3961] Raeburn, K., "Encryption and Checksum Specifications for Kerberos 5", RFC 3961, February 2005.

[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, July 2005.

[RFC4282] Aboba, B., Beadles, M., Arkko, J., and P. Eronen, "The Network Access Identifier", RFC 4282, December 2005.

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

[RFC4402] Williams, N., "A Pseudo-Random Function (PRF) for the Kerberos V Generic Security Service Application Program Interface (GSS-API) Mechanism", RFC 4402, February 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.

[RFC5554] Williams, N., "Clarifications and Extensions to the Generic Security Service Application Program Interface (GSS-API) for the Use of Channel Bindings", RFC 5554, May 2009.

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

10.2. Informative References

[I-D.ietf-abfab-arch]
Howlett, J., Hartman, S., Tschofenig, H., Lear, E., and J. Schaad, "Application Bridging for Federated Access Beyond Web (ABFAB) Architecture", draft-ietf-abfab-arch-03 (work in progress), July 2012.

[I-D.ietf-emu-eap-tunnel-method]
Zhou, H., Cam-Winget, N., Salowey, J., and S. Hanna, "Tunnel EAP Method (TEAP) Version 1", draft-ietf-emu-eap-tunnel-method-03 (work in progress), June 2012.

[I-D.ietf-krb-wg-gss-cb-hash-agility]
Emery, S., "Kerberos Version 5 GSS-API Channel Binding Hash Agility", draft-ietf-krb-wg-gss-cb-hash-agility-10 (work in progress), January 2012.

[I-D.ietf-radext-radius-extensions]
DeKok, A. and A. Lior, "Remote Authentication Dial In User Service (RADIUS) Protocol Extensions", draft-ietf-radext-radius-extensions-06 (work in progress), June 2012.

[I-D.ietf-radext-radsec]

Wierenga, K., McCauley, M., Winter, S., and S. Venaas, "Transport Layer Security (TLS) encryption for RADIUS", draft-ietf-radext-radsec-12 (work in progress), February 2012.

[RFC1964] Linn, J., "The Kerberos Version 5 GSS-API Mechanism", RFC 1964, June 1996.

[RFC3579] Aboba, B. and P. Calhoun, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", RFC 3579, September 2003.

[RFC4072] Eronen, P., Hiller, T., and G. Zorn, "Diameter Extensible Authentication Protocol (EAP) Application", RFC 4072, August 2005.

[RFC4178] Zhu, L., Leach, P., Jaganathan, K., and W. Ingersoll, "The Simple and Protected Generic Security Service Application Program Interface (GSS-API) Negotiation Mechanism", RFC 4178, October 2005.

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

[RFC4462] Hutzelman, J., Salowey, J., Galbraith, J., and V. Welch, "Generic Security Service Application Program Interface (GSS-API) Authentication and Key Exchange for the Secure Shell (SSH) Protocol", RFC 4462, May 2006.

[RFC4559] Jaganathan, K., Zhu, L., and J. Brezak, "SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows", RFC 4559, June 2006.

[RFC5178] Williams, N. and A. Melnikov, "Generic Security Service Application Program Interface (GSS-API) Internationalization and Domain-Based Service Names and Name Type", RFC 5178, May 2008.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5247] Aboba, B., Simon, D., and P. Eronen, "Extensible Authentication Protocol (EAP) Key Management Framework", RFC 5247, August 2008.

[RFC6066] Eastlake, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, January 2011.

Appendix A. Pre-Publication RADIUS VSA

As described in Section 3.4, RADIUS attributes are used to carry the acceptor name when this family of mechanisms is used with RADIUS. Prior to publication of this specification, a vendor-specific RADIUS attribute was used. This non-normative appendix documents that attribute as it may be seen from older implementations.

Prior to IANA assignment, GSS-EAP used a RADIUS vendor-specific attribute for carrying the acceptor name. The VSA with enterprise ID 25622 is formatted as a VSA according to the recommendation in the RADIUS specification. The following sub-attributes are defined:

Name	Attribute	Description
GSS-Acceptor-Service-Name	128	user-or-service portion of name
GSS-Acceptor-Host-Name	129	host portion of name
GSS-Acceptor-Service-specifics	130	service-specifics portion of name
GSS-Acceptor-Realm-Name	131	Realm portion of name

Authors' Addresses

Sam Hartman (editor)
Painless Security

Email: hartmans-ietf@mit.edu

Josh Howlett
JANET

Email: josh.howlett@ja.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 18, 2013

S. Hartman
Painless Security
J. Howlett
JANET(UK)

November 14, 2012

Name Attributes for the GSS-API EAP mechanism
draft-ietf-abfab-gss-eap-naming-07

Abstract

The naming extensions to the Generic Security Services Application Programming interface provide a mechanism for applications to discover authorization and personalization information associated with GSS-API names. The Extensible Authentication Protocol GSS-API mechanism allows an Authentication/Authorization/Accounting peer to provide authorization attributes along side an authentication response. It also provides mechanisms to process Security Assertion Markup Language (SAML) messages provided in the AAA response. This document describes the necessary information to use the naming extensions API to access that information.

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 May 18, 2013.

Table of Contents

1. Introduction

The naming extensions [I-D.ietf-kitten-gssapi-naming-exts] to the Generic Security Services Application Programming interface (GSS-API) [RFC2743] provide a mechanism for applications to discover authorization and personalization information associated with GSS-API names. The Extensible Authentication Protocol GSS-API mechanism [I-D.ietf-abfab-gss-eap] allows an Authentication/Authorization/Accounting (AAA) peer to provide authorization attributes along side an authentication response. It also provides mechanisms to process Security Assertion Markup Language (SAML) messages provided in the AAA response. Other mechanisms such as SAML EC [I-D.ietf-kitten-sasl-saml-ec] also support SAML assertions and attributes carried in the GSS-API. This document describes the necessary information to use the naming extensions API to access SAML assertions in the federated context and AAA attributes.

The semantics of setting attributes defined in this specification are undefined and left to future work.

2. Requirements notation

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. Naming Extensions and SAML

SAML assertions can carry attributes describing properties of the subject of the assertion. For example, an assertion might carry an attribute describing the organizational affiliation or e-mail address of a subject. 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 Universal Resource Identifier (URI) describing the format of the name. The second part, whose form depends on the format URI, is the actual name. GSS-API name attributes may take a form starting with a URI describing the form of the name; the rest of the name is specified by that URI.

SAML attributes carried in GSS-API names are named with three parts. The first is a Universal Resource Name (URN) indicating that the name is a SAML attribute and describing the context (Section 4). This URN is followed by a space, the URI indicating the format of the SAML name, a space and the SAML attribute name. The URI indicating the format of the SAML attribute name is not optional and MUST be present.

SAML attribute names may not be globally unique. Many names that are named by URNs or URIs are likely to have semantics independent of the issuer. However other name formats, including unspecified name formats, make it easy for two issuers to choose the same name for attributes with different semantics. Attributes using the federated context Section 4 are issued by the same party performing the authentication. So, based on who is the subject of the name, the semantics of the attribute can be determined.

4. Federated Context

GSS-API naming extensions have the concept of an authenticated name attribute. The mechanism guarantees that the contents of an authenticated name attribute are an authenticated statement from the trusted source of the peer credential. The fact that an attribute is authenticated does not imply that the trusted source of the peer credential is authorized to assert the attribute.

In the federated context, the trusted source of the peer credential is typically some identity provider. In the GSS EAP mechanism, information is combined from AAA and SAML sources. The SAML IDP and home AAA server are assumed to be in the same trust domain. However, this trust domain is not typically the same as the trust domain of the service. With other SAML mechanisms using this specification, the SAML assertion also comes from the party performing authentication. Typically, the IDP is run by another organization in the same federation. The IDP is trusted to make some statements, particularly related to the context of a federation. For example, an academic federation's participants would typically trust an IDP's assertions about whether someone was a student or a professor. However that same IDP would not typically be trusted to make assertions about local entitlements such as group membership. Thus, a service MUST make a policy decision about whether the IDP is permitted to assert a particular attribute and about whether the asserted value is acceptable. This policy can be implemented as local configuration on the service, as rules in AAA proxies, or through other deployment-specific mechanisms.

In contrast, attributes in an enterprise context are often verified by a central authentication infrastructure that is trusted to assert most or all attributes. For example, in a Kerberos infrastructure, the KDC typically indicates group membership information for clients to a server using KDC-authenticated authorization data.

The context of an attribute is an important property of that attribute; trust context is an important part of this overall context. In order for applications to distinguish the context of attributes, attributes with different context need different names. This specification defines attribute names for SAML and AAA attributes in the federated context.

These names MUST NOT be used for attributes issued by a party other than one closely associated with the source of credentials unless the source of credentials is re-asserting the attributes. For example, a source of credentials can consult whatever sources of attributes it chooses, but acceptors can assume attributes in the federated context are from the source of credentials. This requirement is typically

enforced in mechanism specifications. For example [I-D.ietf-abfab-aaa-saml] provides enough information thatwe know the attributes it carries today are in the federated context. Similarly, we know that the requirements of this paragraph are met by SAML mechanisms where the assertion is the means of authentication.

5. Name Attributes for GSS-EAP

This section describes how RADIUS attributes received in an access-accept message by the GSS-EAP [I-D.ietf-abfab-gss-eap] mechanism are named. The use of attributes defined in this section for other RADIUS messages or prior to the access-accept message is undefined at this time. Future specifations can explore these areas giving adequate weight to backward compatibility. In particular, this specification defines the meaning of these attributes for the src_name output of GSS_Accept_sec_context after that function returns GSS_S_COMPLETE. Attributes MAY be absent or values MAY change in other circumstances; future specifications MAY define this behavior.

The first portion of the name is urn:ietf:params:gss:radius-attribute (a URN indicating that this is a GSS-EAP RADIUS AVP). This is followed by a space and a numeric RADIUS name as described by section 2.6 of [I-D.ietf-radext-radius-extensions]. For example the name of the User-Name attribute is "urn:ietf:params:gss:radius-attribute 1". The name of extended type 1 within type 241 would be "urn:ietf:params:gss:radius-attribute 241.1".

Consider a case where the RADIUS access-accept response includes the RADIUS username attribute. An application wishing to retrieve the value of this attribute would first wait until GSS-_Accept_sec_Context returned GSS_S_COMPLETE. Then the application would take the src_name output from GSS_Accept_sec_context and call GSS_Get_name_attribute passing this name and an attribute of "urn:ietf:params:gss:radius-attribute 1" as inputs. After confirming that the authenticated boolean output is true, the application can find the username in the values output.

The value of RADIUS attributes is the raw octets of the packet. Integers are in network byte order. The display value SHOULD be a human readable string; an implementation can only produce this string if it knows the type of a given RADIUS attribute. If multiple attributes are present with a given name in the RADIUS message, then a multi-valued GSS-API attribute SHOULD be returned. As an exception, implementations SHOULD concatenate RADIUS attributes such as EAP-Message or large attributes defined in [I-D.ietf-radext-radius-extensions] that use multiple attributes to carry more than 253 octets of information.

6. Names of SAML Attributes in the Federated Context

6.1. Assertions

An assertion generated by the credential source is named by "urn:ietf:params:gss:federated-saml-assertion". The value of this attribute is the assertion carried in the AAA protocol or used for authentication in a SAML mechanism. This attribute is absent from a given acceptor name if no such assertion is present or if the assertion fails local policy checks.

When GSS_Get_name_attribute is called, This attribute will be returned with the authenticated output set to true only if the mechanism can successfully authenticate the SAML statement. For the GSS-EAP mechanism this is true if the AAA exchange has successfully authenticated. However, uses of the GSS-API MUST confirm that the attribute is marked authenticated as other mechanisms MAY permit an initiator to provide an unauthenticated SAML statement.

Mechanisms MAY perform additional local policy checks and MAY remove the attribute corresponding to assertions that fail these checks.

6.2. SAML Attributes

Each attribute carried in the assertion SHOULD also be a GSS name attribute. The name of this attribute has three parts, all separated by an ASCII space character. The first part is urn:ietf:params:gss:federated-saml-attribute. The second part is the URI for the <saml:Attribute> element's NameFormat XML attribute. The final part is the <saml:Attribute> element's Name XML attribute. The SAML attribute name may itself contain spaces. As required by the URI specification, spaces within a URI are encoded as "%20". Spaces within a URI, including either the first or second part of the name, encoded as "%20" do not separate parts of the GSS-API attribute name; they are simply part of the URI.

As an example, if the eduPersonEntitlement attribute is present in an assertion, then an attribute with the name "urn:ietf:params:gss:federated-saml-attribute urn:oasis:names:tc:SAML:2.0:attrname-format:uri urn:oid:1.3.6.1.4.1.5923.1.1.1.7" could be returned from GSS_Inquire_Name. If an application calls GSS_Get_name_attribute with this attribute in the attr parameter then the values output would include one or more URIs of entitlements that were associated with the authenticated user.

If the content of each <saml:AttributeValue> element is a simple text node (or nodes), then the raw and "display" values of the GSS name

attribute MUST be the text content of the element(s). The raw value MUST be encoded as UTF-8.

If the value is not simple or is empty, then the raw value(s) of the GSS name attribute MUST be a namespace well-formed serialization [XMLNS]of the <saml:AttributeValue> element(s) encoded as UTF-8. The "display" values are implementation-defined.

These attributes SHOULD be marked authenticated if they are contained in SAML assertions that have been successfully validated back to the trusted source of the peer credential. In the GSS-EAP mechanism, a SAML assertion carried in an integrity-protected and authenticated AAA protocol SHALL be successfully validated; attributes from that assertion SHALL be returned from GSS_Get_name_attribute with the authenticated output set to true. An implementation MAY apply local policy checks to each attribute in this assertion and discard the attribute if it is unacceptable according to these checks.

6.3. SAML Name Identifiers

The <saml:NameID> carried in the subject of the assertion SHOULD also be a GSS name attribute. The name of this attribute has two parts, separated by an ASCII space character. The first part is urn:ietf:params:gss:federated-saml-nameid. The second part is the URI for the <saml:NameID> element’s Format XML attribute.

The raw value of the GSS name attribute MUST be the well-formed serialization of the <saml:NameID> element encoded as UTF-8. The "display" value is implementation-defined. For formats defined by section 8.3 of [OASIS.saml-core-2.0-os], missing values of the NameQualifier or SPNameQualifier XML attributes MUST be populated in accordance with the definition of the format prior to serialization. In other words, the defaulting rules specified for the "persistent" and "transient" formats MUST be applied prior to serialization.

This attribute SHOULD be marked authenticated if the name identifier is contained in a SAML assertion that has been successfully validated back to the trusted source of the peer credential. In the GSS-EAP mechanism, a SAML assertion carried in an integrity-protected and authenticated AAA protocol SHALL be sufficiently validated. An implementation MAY apply local policy checks to this assertion and discard it if it is unacceptable according to these checks.

7. Security Considerations

This document describes how to access RADIUS attributes, SAML attributes and SAML assertions from some GSS-API mechanisms. These attributes are typically used for one of two purposes. The least sensitive is personalization: a central service MAY provide information about an authenticated user so they need not enter it with each acceptor they access. A more sensitive use is authorization.

The mechanism is responsible for authentication and integrity protection of the attributes. However, the acceptor application is responsible for making a decision about whether the credential source is trusted to assert the attribute and validating the asserted value.

Mechanisms are permitted to perform local policy checks on SAML assertions, attributes and name identifiers exposed through name attributes defined in this document. If there is another way to get access to the SAML assertion, for example the mechanism described in [I-D.ietf-abfab-aaa-saml], then an application MAY get different results depending on how the SAML is accessed. This is intended behavior; applications who choose to bypass local policy checks SHOULD perform their own evaluation before relying on information.

8. IANA Considerations

A new top-level registry is created titled "Generic Security Service Application Program Interface Parameters".

In this top-level registry, a sub-registry titled "GSS-API URN Parameters" is created. Registration in this registry is by the IETF review or expert review procedures [RFC5226].

This paragraph gives guidance to designated experts. Registrations in this registry are generally only expected as part of protocols published as RFCs on the IETF stream; other URIs are expected to be better choices for non-IETf work. Expert review is permitted mainly to permit early registration related to specifications under development when the community believes they have reach sufficient maturity. The expert SHOULD evaluate the maturity and stability of such an IETF-stream specification. Experts SHOULD review anything not from the IETF stream for consistency and consensus with current practice. Today such requests would not typically be approved.

If the "paramname" parameter is registered in this registry then its URN will be "urn:ietf:params:gss:paramname". The initial registrations are as follows:

Parameter	Reference
radius-attribute	Section 5
federated-saml-assertion	Section 6.1
federated-saml-attribute	Section 6.2
federated-saml-nameid	Section 6.3

8.1. Registration of the GSS URN Namespace

IANA is requested to register the "gss" URN sub-namespace in the IETF URN sub-namespace for protocol parameters defined in [RFC3553].

Registry Name: gss

Specification: draft-ietf-abfab-gss-eap-naming

Repository: GSS-API URN Parameters (Section 8)

Index Value: Sub-parameters MUST be specified in UTF-8 using standard

URI encoding where necessary.

9. Acknowledgements

Scott Cantor contributed significant text and multiple reviews of this document.

The authors would like to thank Stephen Farrell, Luke Howard, and Jim Schaad

Sam hartman’s work on this specification has been funded by Janet.

10. References

10.1. Normative References

[I-D.ietf-abfab-gss-eap]
Hartman, S. and J. Howlett, "A GSS-API Mechanism for the Extensible Authentication Protocol", draft-ietf-abfab-gss-eap-09 (work in progress), August 2012.

[I-D.ietf-kitten-gssapi-naming-exts]
Williams, N., Johansson, L., Hartman, S., and S. Josefsson, "GSS-API Naming Extensions", draft-ietf-kitten-gssapi-naming-exts-15 (work in progress), May 2012.

[I-D.ietf-radext-radius-extensions]
DeKok, A. and A. Lior, "Remote Authentication Dial In User Service (RADIUS) Protocol Extensions", draft-ietf-radext-radius-extensions-06 (work in progress), June 2012.

[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.

[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.

[RFC3553] Mealling, M., Masinter, L., Hardie, T., and G. Klyne, "An IETF URN Sub-namespace for Registered Protocol Parameters", BCP 73, RFC 3553, June 2003.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[XMLNS] W3C, "XML Namespaces Conformance", 2009, <http://www.w3.org/TR/2009/REC-xml-names-20091208/#Conformance>.

10.2. Informative References

[I-D.ietf-abfab-aaa-saml]
Howlett, J. and S. Hartman, "A RADIUS Attribute, Binding and Profiles for SAML", draft-ietf-abfab-aaa-saml-04 (work in progress), October 2012.

[I-D.ietf-kitten-sasl-saml-ec]
Cantor, S. and S. Josefsson, "SAML Enhanced Client SASL and GSS-API Mechanisms", draft-ietf-kitten-sasl-saml-ec-04 (work in progress), October 2012.

Authors' Addresses

Sam Hartman
Painless Security

Email: hartmans-ietf@mit.edu

Josh Howlett
JANET(UK)

Email: josh.howlett@ja.net

This Internet-Draft, draft-jones-diameter-abfab-00.txt, has expired, and has been deleted from the Internet-Drafts directory. An Internet-Draft expires 185 days from the date that it is posted unless it is replaced by an updated version, or the Secretariat has been notified that the document is under official review by the IESG or has been passed to the RFC Editor for review and/or publication as an RFC. This Internet-Draft was not published as an RFC.

Internet-Drafts are not archival documents, and copies of Internet-Drafts that have been deleted from the directory are not available. The Secretariat does not have any information regarding the future plans of the authors or working group, if applicable, with respect to this deleted Internet-Draft. For more information, or to request a copy of the document, please contact the authors directly.

Draft Authors:
Mark Jones<mark@azu.ca>
Hannes Tschofenig<Hannes.Tschofenig@gmx.net>

ABFAB
Internet-Draft
Intended status: Informational
Expires: September 10, 2011

J. Howlett
JANET(UK)
S. Hartman
Painless Security
H. Tschofenig
Nokia Siemens Networks
E. Lear
Cisco Systems GmbH
March 9, 2011

Application Bridging for Federated Access Beyond Web (ABFAB) Architecture
draft-lear-abfab-arch-02.txt

Abstract

Over the last decade a substantial amount of work has occurred in the space of federated access management. Most of this effort has focused on two use-cases: network and web-based access. However, the solutions to these use-cases that have been proposed and deployed tend to have few common building blocks in common.

This memo describes an architecture that makes use of extensions to the commonly used security mechanisms for both federated and non-federated access management, including RADIUS, Diameter, GSS, GS2, EAP and SAML. The architecture addresses the problem of federated access management to primarily non-web-based services, in a manner that will scale to large numbers of federations.

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 September 10, 2011.

Copyright Notice

Table of Contents

1. Introduction

The Internet uses numerous security mechanisms to manage access to various resources. These mechanisms have been generalized and scaled over the last decade through mechanisms such as SASL/GS2 [RFC5801], Security Assertion Markup Language (SAML) [OASIS.saml-core-2.0-os], RADIUS, and Diameter.

A Relying Party (RP) is the entity that manages access to some resource. The actor that is requesting access to that resource is often described as the Subject. Many security mechanisms are manifested as an exchange of information between these actors. The RP is therefore able to decide whether the Subject is authorised, or not.

Some security mechanisms allow the RP to delegate aspects of the access management decision to an actor called the Identity Provider (IdP). This delegation requires technical signalling, trust and a common understanding of semantics between the RP and IdP. These aspects are generally managed within a relationship known as a 'federation'. This style of access management is accordingly described as 'federated access management'.

Federated access management has evolved over the last decade through such standards as SAML, OpenID [1], and OAUTH [RFC5849]. The benefits of federated access management include:

- Single or Simplified sign-on. An Internet service can delegate access management, and the associated responsibilities such as identity management and credentialing, to an organisation that already has a long-term relationship with the Subject. This is often attractive for Relying Parties who frequently do not want these responsibilities. The Subject may also therefore require fewer credentials, which is often desirable.

- Privacy. Often a Relying Party does not need to know the identity of a Subject to reach an access management decision. It is frequently only necessary for the Relying Party to establish, for example, that the Subject is affiliated with a particular organisation or has a certain role or entitlement. Sometimes the RP does require an identifier for the Subject (for example, so that it can recognise the Subject subsequently); in this case, it is common practise for the IdP to only release a pseudonym that is specific to that particular Relying Party. Federated access management therefore provides various strategies for protecting the Subject's privacy.

- Provisioning. Sometimes a Relying Party needs, or would like, to know more about a subject that an affiliation or pseudonym. For example, a Relying Party may want the Subject's email address or name. Some federated access management technologies provide the ability for the IdP to provision this information, either on request by by the RP or unsolicited.

1.1. Terminology

This document uses identity management and privacy terminology from [I-D.hansen-privacy-terminology]. In particular, this document uses the terms pseudonymity, unlinkability, anonymity.

We make one note about the IdP: in this architecture, the IdP consists of the following components: an EAP server, a radius server, and optionally a SAML Assertion service. The IdP is also responsible for authentication, even though it may rely upon other components within a domain for such an operation. The reader is advised that for succinctness, in most cases the term IdP is used, except where additional clarity seems appropriate.

1.2. An Overview of Federation

In the previous section we introduced the following actors:

- the Subject,
- the Identity Provider, and
- the Relying Party.

These entities and their relationships are illustrated graphically in Figure 1.

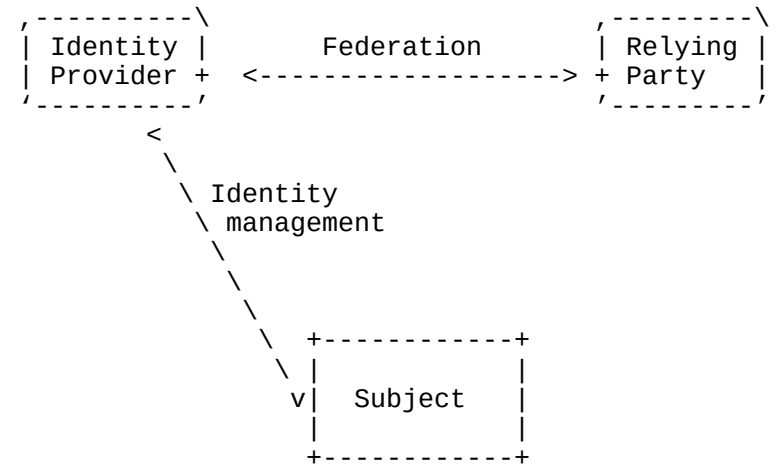


Figure 1: General federation framework model

Figure 1 shows the federation relationship between the IdP and RP. Typically this relationship encompasses the following:

- Technical infrastructure. This infrastructure is generally used to support signalling and trust establishment between the IdP and RP.

- Policy framework. This framework is generally used to establish expectations between an IdP and RP that may facilitate stronger trust and common semantics.

The nature of federation dictates that there is some form of relationship between the identity provider and the relying party. This is particularly important when the relying party wants to use information obtained from the identity provider for access management decisions and when the identity provider does not want to release information to every relying party (or only under certain conditions).

In a federation, policy is agreed upon by some form of administrative management. The policy and infrastructure are then instantiated through an operational framework. This may include the measurement of compliance in some fashion. To support the more generic deployment case, we assume that the identity provider and the RP belong to different administrative domains.

While it is possible to have a bilateral agreement between every IdP and every RP; on an Internet scale this setup requires the introduction of the multi-lateral federation concept, as the

management of such pair-wise relationships would otherwise prove burdensome.

While many of the non-technical aspects of federation, such as business practices and operational arrangements, are outside the scope of the IETF they still impact the architecture setup on how to ensure the dynamic establishment of trust.

Some deployments are sometimes required to deploy complex technical infrastructure including message routing intermediaries to offer the required technical functionality while in other deployments those are missing.

Figure 1 also shows the relationship between the IdP and the Subject. Often a real world entity is associated with the Subject; for example, a person or some software.

The IdP will typically have a long-term relationship with the Subject. This relationship would typically involve the IdP positively identifying and credentialling the Subject (for example, at time of enrollment in the context of employment within an organisation). The relationship will often be instantiated within an agreement between the IdP and the Subject (for example, within an employment contract or terms of use that stipulates the appropriate use of credentials and so forth).

While federation is often discussed within the context of relatively formal relationships, such as between an Enterprise and an Employee or a Government and a Citizen, federation does not in any way require this; nor, indeed, does it require any particular level of formality. It is, for example, entirely compatible with a relationship between the IdP and Subject that is only as weak as completing a web form and confirming the verification email.

However, the nature and quality of the relationship between the Subject and the IdP is an important contributor to the level of trust that an RP may attribute to an assertion describing a Subject made by an IdP. This is sometimes described as the Level of Assurance.

Similarly it is also important to note that, in the general case, there is no requirement of a relationship betweem the RP and the Subject. This is a property of federation that yields many of its benefits. However, federation does not preclude the possibility relationship between the RP and the Subject, should needs dictate.

Finally, it is important to reiterate that in some scenarios there might indeed be a human behind the device denoted as Subject and in other cases there is no human involved in the actual protocol

execution.

1.3. Challenges to Contemporary Federation

JH: Much more content needed!

As the number of such federated services has proliferated, however, the role of the individual has become ambiguous in certain circumstances. For example, a school might provide online access to grades to a parent who is also a teacher. She must clearly distinguish her role upon access. After all, she is probably not allowed to edit her own child's grades.

Similarly, as the number of federations proliferates, it becomes increasingly difficult to discover which identity provider a user is associated with. This is true for both the web and non-web case, but particularly acute for the latter ans many non-web authentication systems are not semantically rich enough on their own to allow for such ambiguities. For instance, in the case of an email provider, the use of SMTP and IMAP protocols does not on its own provide for a way to select a federation. However, the building blocks do exist to add this functionality.

1.4. An Overview of ABFAB-based Federation

The previous section described the general model of federation, and its the application of federated access management. This section provides a brief overview of ABFAB in the context of this model.

The steps taken generally in an ABFAB federated authentication/ authorization exchange are as follows:

1. Principal provides NAI to Application: Somehow the client is configured with at least the realm portion of an NAI, which represents the IdP to be discovered.

2. Authentication mechanism selection: this is the step necessary to indicate that the GSS-EAP SASL/GS2 mechanism will be used for authentication/authorization.

3. Client Application provides NAI to RP: At the conclusion of mechanism selection the NAI must be provided to the RP for discovery.

4. Discovery of federated IdP: This is discussed in detail below. Either the RP is configured with authorized IdPs, or it makes use of a federation proxy.

5. Request from Relying Party to IdP: Once the RP knows who the IdP is, it or its agent will forward RADIUS request that encapsulates a GSS/EAP access request to an IdP. This may or may not contain a SAML request as a series of attributes.. At this stage, the RP will likely have no idea who the principal is. The RP claims its identity to the IdP in AAA attributes, and it makes whatever SAML Attribute Requests through a AAA attribute. XXX- Check order of SAML attribute request.

6. IdP informs the principal of which EAP method to use: The available and appropriate methods are discussed below in this memo.

7. A bunch of EAP messages happen between the endpoints: Messages are exchanged between the principal and the IdP until a result is determined. The number and content of those messages will depend on the EAP method. If the IdP is unable to authenticate the principal, the process concludes here. As part of this process, the principal will, under protection of EAP, assert the identity of the RP to which it intends to authenticate.

8. Successful Authentication: At the very least the IdP (its EAP server) and EAP peer / subject have authenticated one another. As a result of this step, the subject and the IdP hold two cryptographic keys- a Master Session Key (MSK), and an Extended MSK (EMSK). If the asserted identity of the RP by the principal matches the identity the RP itself asserted, there is some confidence that the RP is now authenticated to the IdP.

9. Local IdP Policy Check: At this stage, the IdP checks local policy to determine whether the RP and subject are authorized for a given transaction/service, and if so, what if any, attributes will be released to the RP. Additional policy checks will likely have been made earlier just through the process of discovery.

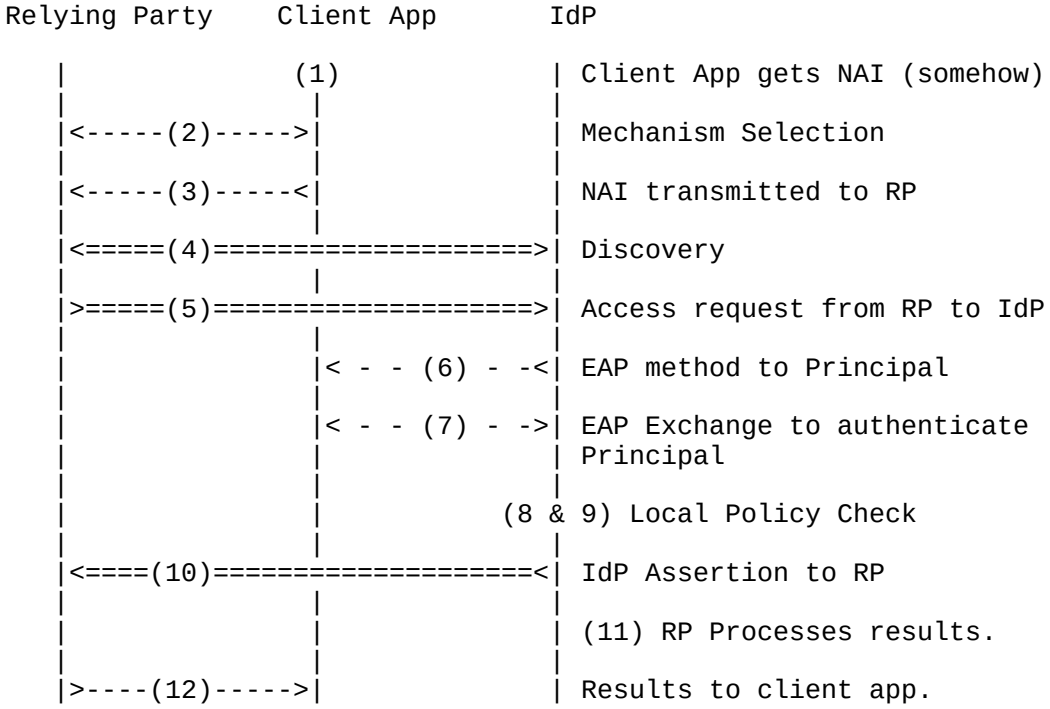
10. Response from the IdP to the Relying Party: Once the IdP has made a determination of whether and how to authenticate or authorize the principal to the RP, it returns either a negative AAA result to the RP, or it returns a positive result to the RP, along with an optional set of AAA attributes associated with the principal that could include one or more SAML assertions. In addition, an EAP MSK is returned to the subject.

11. RP Processes Results. When the RP receives the result from the IdP, it should have enough information to either grant or refuse a resource access request. It may have information that leads it to make additional attribute queries. It may have

information that associates the principal with specific authorization identies. It will apply these results in an application-specific way.

12. RP returns results to principal: Once the RP has a response it must inform the client application of the result. If all has gone well, all are authenticated, and the application proceeds with appropriate authorization levels.

An example communication flow is given below:



----- = Between Client App and RP
===== = Between RP and IdP
- - - = Between Client App and IdP

1.5. Design Goals

Our key design goals are as follows:

- Each party of a transaction will be authenticated, and the principal will be authorized for access to a specific resource .
- Means of authentication is decoupled so as to allow for multiple authentication methods.
- Hence, the architecture requires no sharing of long term private keys.
- The system will scale to large numbers of identity providers, relying parties, and users.
- The system will be designed primarily for non-Web-based authentication.
- The system will build upon existing standards, components, and operational practices.

Designing new three party authentication and authorization protocols is hard and frought with risk of cryptographic flaws. Achieving widespead deployment is even more difficult. A lot of attention on federated access has been devoted to the Web. This document instead focuses on a non-Web-based environment and focuses on those protocols where HTTP is not used. Despite the increased excitement for layering every protocol on top of HTTP there are still a number of protocols available that do not use HTTP-based transports. Many of these protocols are lacking a native authentication and authorization framework of the style shown in Figure 1.

1.6. Use of AAA

Interestingly, for network access authentication the usage of the AAA framework with RADIUS [RFC2865] and Diameter [RFC3588] was quite successful from a deployment point of view. To map the terminology used in Figure 1 to the AAA framework the IdP corresponds to the AAA server, the RP corresponds to the AAA client, and the technical building blocks of a federation are AAA proxies, relays and redirect agents (particularly if they are operated by third parties, such as AAA brokers and clearing houses). The front-end, i.e. the end host to AAA client communication, is in case of network access authentication offered by link layer protocols that forward authentication protocol exchanges back-and-forth. An example of a large scale RADIUS-based federation is EDUROAM [2].

Is it possible to design a system that builds on top of successful protocols to offer non-Web-based protocols with a solid starting point for authentication and authorization in a distributed system?

2. Architecture

Section 1 already introduced the federated access architecture, with the illustration of the different actors that need to interact, but it did not expand on the specifics of providing support for non-Web based applications. This section details this aspect and motivates design decisions. The main theme of the work described in this document is focused on re-using existing building blocks that have been deployed already and to re-arrange them in a novel way.

Although this architecture assumes updates to both the relying party as well as to the end host for application integration, those changes are kept at a minimum. A mechanism that can demonstrate deployment benefits (based on ease of update of existing software, low implementation effort, etc.)is preferred and there may be a need to specify multiple mechanisms to support the range of different deployment scenarios.

There are a number of ways for encapsulating EAP into an application protocol. For ease of integration with a wide range of non-Web based application protocols the usage of the GSS-API was chosen. Encapsulating EAP into the GSS-API also allows EAP to be used in SASL. A description of the technical specification can be found in [I-D.ietf-abfab-gss-eap]. Other alternatives exist as well and may be considered later, such as "TLS using EAP Authentication" [I-D.nir-tls-eap].

There are several architectural layers in the system; this section discusses the individual layers.

2.1. Federation Substrate

The federation substrate is responsible for the connunication between the relying party and the identity provider. This layer is responsible for the inter-domain communication and for the technical mechanisms necessary to establish inter-domain trust.

A key design goal is the re-use of an existing infrastructure, we build upon the AAA framework as utilized by RADIUS [RFC2138] and Diameter [RFC3588]. Since this document does not aim to re-describe the AAA framework the interested reader is referred to [RFC2904]. Building on the AAA infrastructure, and RADIUS and Diameter as protocols, modifications to that infrastructure is to be avoided. Also, modifications to AAA servers should be kept at a minimum.

The astute reader will notice that RADIUS and Diameter have substantially similar characteristics. Why not pick one? A key difference is that today RADIUS is largely transported upon UDP, and

its use is largely, though not exclusively, intra-domain. Diameter itself was designed to scale to broader uses. We leave as a deployment decision, which protocol will be appropriate.

Through the integrity protection mechanisms in the AAA framework, the relying party can establish technical trust that messages are being sent by the appropriate relying party. Any given interaction will be associated with one federation at the policy level. The legal or business relationship defines what statements the identity provider is trusted to make and how these statements are interpreted by the relying party. The AAA framework also permits the relying party or elements between the relying party and identity provider to make statements about the relying party.

The AAA framework provides transport for attributes. Statements made about the subject by the identity provider, statements made about the relying party and other information is transported as attributes.

2.1.1. Discovery, Rules Determination, and Technical Trust

One demand that the AAA substrate must make of the upper layers is that they must properly identify the end points of the communication. That is- it must be possible for the AAA client at the RP to determine where to send each RADIUS or Diameter message. Without this requirement, it would be the RP's responsibility to determine the identity of the principal on its own, without the assistance of an IdP. This architecture makes use of the Network Access Identifier (NAI), where the IdP is indicated in the realm component [RFC4282]. The NAI is represented and consumed by the GSS-API layer as GSS_C_NT_USER_NAME as specified in [RFC2743]. The GSS-API EAp mechanism includes the NAI in the EAP Response/Identity message.

The RP needs to discover which federation will be used to contact the IDP. As part of this process, the RP determines the set of business rules and technical policies governing the relationship; this is called rules determination. The RP also needs to establish technical trust in the communications with the IDP.

Rules determination covers a broad range of decisions about the exchange. One of these is whether the given RP is permitted to talk to the IDP using a given federation at all, so rules determination encompasses the basic authorization decision. Other factors are included, such as what policies govern release of information about the principal to the RP and what policies govern the RP's use of this information. While rules determination is ultimately a business function, it has significant impact on the technical exchanges. The protocols need to communicate the result of authorization. When multiple sets of rules are possible, the protocol must disambiguate

which set of rules are in play. Some rules have technical enforcement mechanisms; for example in some federations intermediates validate information that is being communicated within the federation.

Several deployment approaches are possible. These can most easily be classified based on the mechanism for technical trust that is used. The choice of technical trust mechanism constrains how rules determination is implemented. Regardless of what deployment strategy is chosen, the technical trust mechanism MUST constrain the names of both parties to the exchange. The trust mechanism MUST make sure that the entity acting as IDP for a given NAI is permitted to be the IDP for that realm. The mechanism MUST make sure that any service name claimed by the RP is permitted to be claimed by that entity. Here are the categories of technical trust determination:

AAA Proxy: The simplest model is that an RP supports a request directly to an AAA proxy. The hop-by-hop integrity protection of the AAA fabric provides technical trust. An RP can submit a request directly to a federation. Alternatively, a federation disambiguation fabric can be used. Such a fabric takes information about what federations the RP is part of and what federations the IDP is part of and routes a message to the appropriate federation. The routing of messages across the fabric plus attributes added to requests and responses provides rules determination. For example, when a disambiguation fabric routes a message to a given federation, that federation’s rules are chosen. Naming is enforced as messages travel across the fabric. The entities near the RP confirm its identity and validate names it claims. The fabric routes the message towards the appropriate IDP, validating the IDP’s name in the process. The routing can be statically configured. Alternatively a routing protocol could be developed to exchange reachability information about given IDPs and to apply policy across the AAA fabric. Such a routing protocol could flood naming constraints to the appropriate points in the fabric.

Trust Broker: Instead of routing messages through AAA proxies, some trust broker could establish keys between entities near the RP and entities near the IDP. The advantage of this approach is efficiency of message handling. Fewer entities are needed to be involved for each message. Security may be improved by sending individual messages over fewer hops. Rules determination involves decisions made by trust brokers about what keys to grant. Also, associated with each credential is context about rules and about other aspects of technical trust including names that may be claimed. A routing protocol similar to the one for AAA proxies is likely to be useful to trust brokers in flooding rules and naming

constraints.

Global Credential: A global credential such as a public key and certificate in a public key infrastructure can be used to establish technical trust. A directory or distributed database such as the Domain Name System is needed for an RP to discover what endpoint to contact for a given NAI. Certificates provide a place to store information about rules determination and naming constraints. Provided that no intermediates are required and that the RP and IDP are sufficient to enforce and determine rules, rules determination is reasonably simple. However applying certain rules is likely to be quite complex. For example if multiple sets of rules are possible between an IDP and RP, confirming the correct set is used may be difficult. This is particularly true if intermediates are involved in making the decision. Also, to the extent that directory information needs to be trusted, rules determination may be more complex.

Real-world deployments are likely to be mixtures of these basic approaches. For example, it will be quite common for an RP to route traffic to a AAA proxy within an organization. That proxy MAY use any of the three methods to get closer to the IDP. It is also likely that rather than being directly reachable, an IDP may have a proxy within its organization. Federations MAY provide a traditional AAA proxy interface even if they also provide another mechanism for increased efficiency or security.

2.2. Subject To Identity Provider

Traditional web federation does not describe how a subject communicates with an identity provider. As a result, this communication is not standardized. There are several disadvantages to this approach. It is difficult to have subjects that are machines rather than humans that use some sort of programatic credential. In addition, use of browsers for authentication restricts the deployment of more secure forms of authentication beyond plaintext username and password known by the server. In a number of cases the authentication interface may be presented before the subject has adequately validated they are talking to the intended server. By giving control of the authentication interface to a potential attacker, then the security of the system may be reduced and phishing opportunities introduced.

As a result, it is desirable to choose some standardized approach for communication between the subject’s end-host and the identity provider. There are a number of requirements this approach must meet.

Experience has taught us one key security and scalability requirement: it is important that the relying party not get in possession of the long-term secret of the entity being authenticated by the AAA server. Aside from a valuable secret being exposed, a synchronization problem can also often develop. Since there is no single authentication mechanism that will be used everywhere there is another associated requirement: The authentication framework must allow for the flexible integration of authentication mechanisms. For instance, some identity providers may require hardware tokens while others may use passwords. A service provider would want to support both sorts of federations, and others.

Fortunately, these requirements can be met by utilizing standardized and successfully deployed technology, namely by the Extensible Authentication Protocol (EAP) framework [RFC3748]. Figure 2 illustrates the integration graphically.

EAP is an end-to-end framework; it provides for two-way communication between a peer (i.e,service client or principal) through the authenticator (i.e., service provider) to the back-end (i.e., identity provider). Conveniently, this is precisely the communication path that is needed for federated identity. Although EAP support is already integrated in AAA systems (see [RFC3579] and [RFC4072]) several challenges remain: one is to carry EAP payloads from the end host to the relying party. Another is to verify statements the relying party has made to the subject, confirm these statements are consistent with statements made to the identity provider and confirm all the above are consistent with the federation and any federation-specific policy or configuration. Another challenge is choosing which identity provider to use for which service.

2.3. Application to Service

One of the remaining layers is responsible for integration of federated authentication into the application. There are a number of approaches that applications have adopted for security. So, there may need to be multiple strategies for integration of federated authentication into applications. However, we have started with a strategy that provides integration to a large number of application protocols.

Many applications such as SSH [RFC4462], NFS [RFC2203], DNS [RFC3645] and several non-IETF applications support the Generic Security Services Application Programming Interface [RFC2743]. Many applications such as IMAP, SMTP, XMPP and LDAP support e Simple Authentication and Security Layer (SASL) [RFC4422] framework. These two approaches work together nicely: by creating a GSS-API mechanism,

SASL integration is also addressed. In effect, using a GSS-API mechanism with SASL simply requires placing some headers on the front of the mechanism and constraining certain GSS-API options.

GSS-API is specified in terms of an abstract set of operations which can be mapped into a programming language to form an API. When people are first introduced to GSS-API, they focus on it as an API. However, from the prospective of authentication for non-web applications, GSS-API should be thought of as a protocol not an API. It consists of some abstract operations such as the initial context exchange, which includes two sub-operations (gss_init_sec_context and gss_accept_sec_context). An application defines which abstract operations it is going to use and where messages produced by these operations fit into the application architecture. A GSS-API mechanism will define what actual protocol messages result from that abstract message for a given abstract operation. So, since this work is focusing on a particular GSS-API mechanism, we generally focus on protocol elements rather than the API view of GSS-API.

The API view has significant value. Since the abstract operations are well defined, the set of information that a mechanism gets from the application is well defined. Also, the set of assumptions the application is permitted to make is generally well defined. As a result, an application protocol that supports GSS-API or SASL is very likely to be usable with a new approach to authentication including this one with no required modifications. In some cases, support for a new authentication mechanism has been added using plugin interfaces to applications without the application being modified at all. Even when modifications are required, they can often be limited to supporting a new naming and authorization model. For example, this work focuses on privacy; an application that assumes it will always obtain an identifier for the principal will need to be modified to support anonymity, unlinkability or pseudonymity.

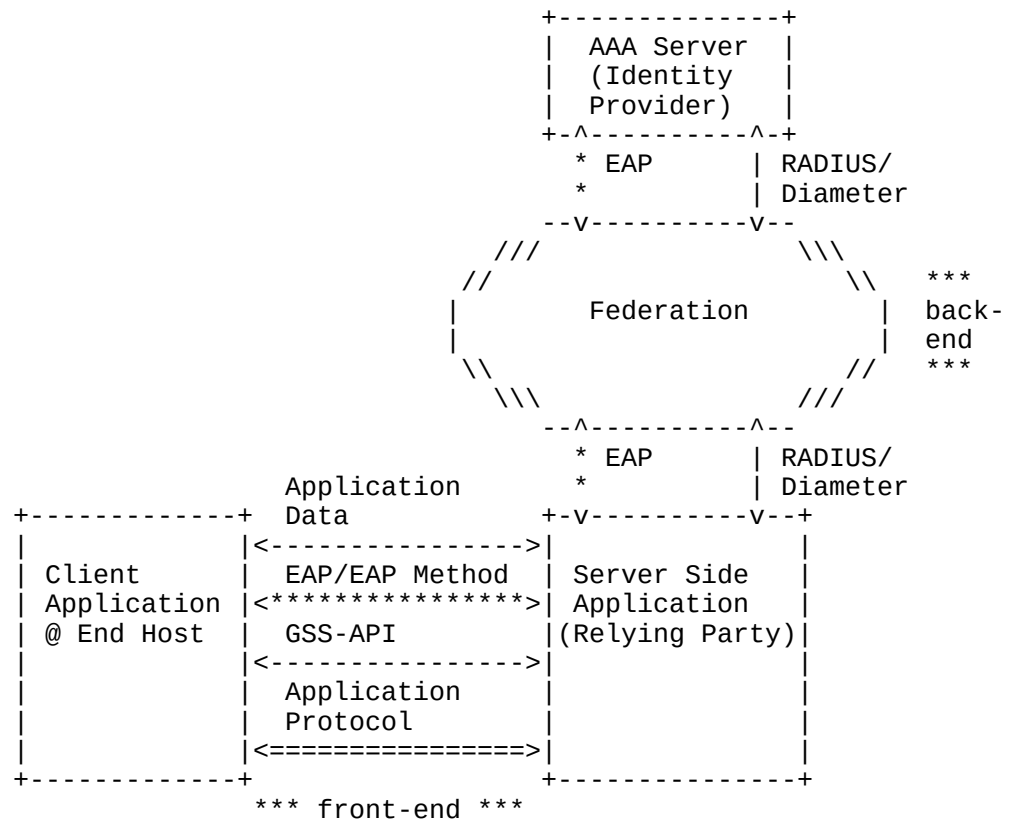
So, we use GSS-API and SASL because a number of the application protocols we wish to federate support these strategies for security integration. What does this mean from a protocol standpoint and how does this relate to other layers? This means we need to design a concrete GSS-API mechanism. We have chosen to use a GSS-API mechanism that encapsulates EAP authentication. So, GSS-API (and SASL) encapsulate EAP between the end-host and the service. The AAA framework encapsulates EAP between the relying party and the identity provider. The GSS-API mechanism includes rules about how principals and services are named as well as per-message security and other facilities required by the applications we wish to support.

2.4. Personalization Layer

The AAA framework provides a way to transport statements from the identity provider to the relying party. However, we also need to say more about the content of these statements. In simple cases, attributes particular to the AAA protocol can be defined. However in more complicated situations it is strongly desirable to re-use an existing protocol for asking questions and receiving information about subjects. SAML is used for this.

SAML usage may be as simple as the identity provider including a SAML Response message in the AAA response. Alternatively the relying party may generate a SAML request XXX to whom, how, and at what point? (see above XXX).

2.5. Tieing Layers Together



Legend:

<****>: End-to-end exchange
<---->: Hop-by-hop exchange
<====>: Protocol through which GSS-API/GS2 exchanges are tunnelled

Figure 2: Architecture for Federated Access of non-Web based Applications

3. Application Security Services

One of the key goals is to integrate federated authentication into existing application protocols and where possible, existing implementations of these protocols. Another goal is to perform this integration while meeting the best security practices of the technologies used to perform the integration. This section describes security services and properties required by the EAP GSS-API mechanism in order to meet these goals. This information could be viewed as specific to that mechanism. However, other future application integration strategies are very likely to need similar services. So, it is likely that these services will be expanded across application integration strategies if new application integration strategies are adopted.

3.1. Server (Mutual) Authentication

GSS-API provides an optional security service called mutual authentication. This service means that in addition to the initiator providing (potentially anonymous or pseudonymous) identity to the acceptor, the acceptor confirms its identity to the initiator. Especially for the ABFAB context, this service is confusingly named. We still say that mutual authentication is provided when the identity of an acceptor is strongly authenticated to an anonymous initiator.

RFC 2743 does not explicitly talk about what mutual authentication means. Within the GSS-API community successful mutual authentication has come to mean:

- If a target name is supplied by the initiator, then the initiator trusts that the supplied target name describes the acceptor. This implies both that appropriate cryptographic exchanges took place for the initiator to make such a trust decision, and that after evaluating the results of these exchanges, the initiator’s policy trusts that the target name is accurate.

- The initiator trusts that its idea of the acceptor name correctly names the entity it is communicating with.

- Both the initiator and acceptor have the same key material for per-message keys and both parties have confirmed they actually have the key material. In EAP terms, there is a protected indication of success.

Mutual authentication is an important defense against certain aspects of phishing. Intuitively, users would like to assume that if some party asks for their credentials as part of authentication, successfully gaining access to the resource means that they are

talking to the expected party. Without mutual authentication, the acceptor could "grant access" regardless of what credentials are supplied. Mutual authentication better matches this user intuition.

The GSS-EAP mechanism MUST implement mutual authentication. That is, an initiator needs to be able to request mutual authentication. When mutual authentication is requested, only EAP methods capabale of providing the necessary service can be used, and appropriate steps need to be taken to provide mutual authentication. A broader set of EAP methods could be supported when a particular application does not request mutual authentication. It is an open question whether the mechanism will permit this.

3.2. GSS-API Channel Binding

[RFC5056] defines a concept of channel binding to prevent man-in-the-middle attacks. It is common to provide SASL and GSS-API with another layer to provide transport security; Transport Layer Security (TLS) is the most common such layer. TLS provides its own server authentication. However there are a variety of situations where this authentication is not checked for policy or usability reasons. Even when it is checked, if the trust infrastructure behind the TLS authentication is different from the trust infrastructure behind the GSS-API mutual authentication. If the endpoints of the GSS-API authentication are different than the endpoints of the lower layer, this is a strong indication of a problem such as a man-in-the-middle attack. Channel binding provides a facility to determine whether these endpoints are the same.

The GSS-EAP mechanism needs to support channel binding. When an application provides channel binding data, the mechanism needs to confirm this is the same on both sides consistent with the GSS-API specification. XXXThere is an open question here as to the details; today RFC 5554 governs. We could use that and the current draft assumes we will. However in Beijing we became aware of some changes to these details that would make life much better for GSS authentication of HTTP. We should resolve this with kitten and replace this note with a reference to the spec we’re actually following.

Typically when considering channel binding, people think of channel binding in combination with mutual authentication. This is sufficiently common that without additional qualification channel binding should be assumed to imply mutual authentication. Without mutual authentication, only one party knows that the endpoints are correct. That’s sometimes useful. Consider for example a user who wishes to access a protected resource from a shared whiteboard in a conference room. The whiteboard is the initiator; it does not need

to actually authenticate that it is talking to the correct resource because the user will be able to recognize whether the displayed content is correct. If channel binding were used without mutual authentication, it would in effect be a request to only disclose the resource in the context of a particular channel. Such an authentication would be similar in concept to a holder-of-key SAML assertion. However, also note that while it is not happening in the protocol, mutual authentication is happening in the overall system: the user is able to visually authenticate the content. This is consistent with all uses of channel binding without protocol level mutual authentication found so far.

RFC 5056 channel binding (also called GSS-API channel binding when GSS-API is involved) is not the same thing as EAP channel binding. EAP channel binding is also used in the ABFAB context in order to implement acceptor naming and mutual authentication. Details are discussed in the mechanisms specification [I-D.ietf-abfab-gss-eap].

3.3. Host-Based Service Names

IETF security mechanisms typically take the name of a service entered by a user and make some trust decision about whether the remote party in an interaction is the intended party. GSS-API has a relatively flexible naming architecture. However most of the IETF applications that use GSS-API, including SSH, NFS, IMAP, LDAP and XMPP, have chosen to use host-based service names when they use GSS-API. In this model, the initiator names an acceptor based on a service such as "imap" or "host" (for login services such as SSH) and a host name.

Using host-based service names leads to a challenging trust delegation problem. Who is allowed to decide whether a particular hostname maps to an entity. The public-key infrastructure (PKI) used by the web has chosen to have a number of trust anchors (root certificate authorities) each of wich can map any name to a public key. A number of GSS-API mechanisms suchs as Kerberos [RFC1964] split the problem into two parts. A new concept called a realm is introduced. Then the mechanism decides what realm is responsible for a given name. That realm is responsible for deciding if the acceptor entity is allowed to claim the name. ABFAB needs to adopt this approach.

Host-based service names do not work ideally when different instances of a service are running on different ports. Also, these do not work ideally when SRV record or other insecure referrals are used.

The GSS-EAP mechanism needs to support host-based service names in order to work with existing IETF protocols.

3.4. Per-Message Tokens

GSS-API provides per-message security services that can provide confidentiality and integrity. Some IETF protocols such as NFS and SSH take advantage of these services. As a result GSS-EAP needs to support these services. As with mutual authentication, per-message services will limit the set of EAP methods that are available. Any method that produces a Master Session Key (MSK) should be able to support per-message security services.

GSS-API provides a pseudo-random function. While the pseudo-random function does not involve sending data over the wire, it provides an algorithm that both the initiator and acceptor can run in order to arrive at the same key value. This is useful for designs where a successful authentication is used to key some other function. This is similar in concept to the TLS extractor. No current IETF protocols require this. However GSS-EAP supports this service because it is valuable for the future and easy to do given per-message services. Non-IETF protocols are expected to take advantage of this in the near future.

4. Future Work: Attribute Providers

This architecture provides for a federated authentication and authorization framework between IdPs, RPs, principals, and subjects. It does not at this time provide for a means to retrieve attributes from 3rd parties. However, it envisions such a possibility. We note that in any extension to the model, an attribute provider must be authorized to release specific attributes to a specific RP for a specific principal. In addition, we note that it is an open question beyond this architecture as to how the RP should know to trust a particular attribute provider.

There are a number of possible technical means to provide attribute provider capabilities. One possible approach is for the IdP to provide a signed attribute request to RP that it in turn will provide to the attribute authority. Another approach would be for the IdP to provide a URI to the RP that contains a token of some form. The form of communications between the IdP and attribute provider as well as other considerations are left for the future. One thing we can say now is that the IdP would merely be asserting who the attribute authority is, and not the contents of what the attribute authority would return. (Otherwise, the IdP might as well make the query to the attribute authority and then resign it.)

5. Privacy Considerations

Sharing identity information raises privacy violations and as described throughout this document an existing architecture is re-used for a different usage environment. As such, a discussion about the privacy properties has to take these pre-conditions into consideration. We use the approach suggested in [I-D.morris-privacy-considerations] to shed light into what data is collected and used by which entity, what the relationship between these entities and the end user is, what data about the user is likely needed to be collected, and what the identification level of the data is.

5.1. What entities collect and use data?

Figure 2 shows the architecture with the involved entities. Message exchanges are exchanged between the client application, via the relying part to the AAA server. There will likely be intermediaries between the relying party and the AAA server, labeled generically as "federation".

In order for the relying party to route messages to the AAA server it is necessary for the client application to provide enough information to enable the identification of the AAA server's domain. While often the username is attached to the domain (in the form of a Network Access Identity (NAI) this is not necessary for the actual protocol operation. The EAP server component within the AAA server needs to authenticate the user and therefore needs to execute the respective authentication protocol. Once the authentication exchange is complete authorization information needs to be conveyed to the relying party to grant the user the necessary application rights. Information about resource consumption may be delivered as part of the accounting exchange during the lifetime of the granted application session.

The authentication exchange may reveal an identifier that can be linked to a user. Additionally, a sequence of authentication protocol exchanges may be linked together. Depending on the chosen authentication protocol information at varying degrees may be revealed to all parties along the communication path. This authorization information exchange may disclose identity information about the user. While accounting information is created by the relying party it is likely to needed by intermediaries in the federation for financial settlement purposes and will be stored for billing, fraud detection, statistical purposes, and for service improvement by the AAA server operator.

5.2. Relationship between User's and other Entities

The AAA server is a first-party site the user typically has a relationship with. This relationship will be created at the time when the security credentials are exchange and provisioned. The relying party and potential intermediares in the federation are typically operate under the contract of the first-party site. The user typically does not know about the intermediaries in the federation nor does he have any relationship with them. The protocol interaction triggered by the client application happens with the relying party at the time of application access. The relying party does not have a direct contractual relationship with the user but depending on the application the interaction may expose the brand of the application running by the relying party to the end user via some user interface.

5.3. What Data about the User is likely Needed to be Collected?

The data that is likely going to be collected as part of a protocol exchange with application access at the relying party is accounting information and authorization information. This information is likely to be kept beyond the terminated application usage for trouble shooting, statistical purposes, etc. There is also like to be additional data collected to to improve application service usage by the relying party that is not conveyed to the AAA server as part of the accounting stream.

5.4. What is the Identification Level of the Data?

With regard to identification there are several protocol layers that need to be considered separately. First, there is the EAP method exchange, which as an authentication and key exchange protocol has properties related to identification and protocol linkage. Second, there is identification at the EAP layer for routing of messages. Then, in the exchange between the client application and the relying party the identification depends on the underlying application layer protocol the EAP/GSS-API exchange is tunneled over. Finally, there is the backend exchange via the AAA infrastructure, which involves a range of RADIUS and Diameter extensions and yet to be defined extensions, such as encoding authorization information inside SAML assertions.

Since this document does not attempt to define any of these exchanges but rather re-uses existing mechanisms the level of identification heavily depends on the selected mechanisms. The following two examples aim to illustrate the amount of existing work with respect to decrease exposure of personal data.

1. When designing EAP methods a number of different requirements may need to get considered; some of them are conflicting. RFC 4017 [RFC4017], for example, tried to list requirements for EAP methods utilized for network access over Wireless LANs. It also recommends the end-user identity hiding requirement, which is privacy-relevant. Some EAP methods, such as EAP-IKEv2 [RFC5106], also fulfill this requirement.

2. EAP, as the layer encapsulating EAP method specific information, needs identity information to allow routing requests towards the user's home AAA server. This information is carried within the Network Access Identifier (NAI) and the username part of the NAI (as supported by RFC 4282 [RFC4282]) can be obfuscated.

5.5. Privacy Challenges

While a lot of standarization work was done to avoid leakage of identity information to intermediaries (such as eavesdroppers on the communication path between the client application and the relying party) in the area of authentication and key exchange protocols. However, from current deployments the weak aspects with respect to security are:

1. Today business contracts are used to create federations between identity providers and relying parties. These contracts are not only financial agreements but they also define the rules about what information is exchanged between the AAA server and the relying party and the potential involvement of AAA proxies and brokers as intermediaries. While these contracts are openly available for university federations they are not public in case of commercial deployments. Quite naturally, there is a lack of transparency for external parties.

2. In today's deployments the ability for users to determine the amount of information exchanged with other parties over time, as well as the possibility to control the amount of information exposed via an explict consent is limited. This is partially due the nature of application capabilities at the time of network access authentication. However, with the envisioned extension of the usage, as described in this document, it is desirable to offer these capabilities.

6. Deployment Considerations

6.1. EAP Channel Binding

Discuss the implications of needing EAP channel binding.

6.2. AAA Proxy Behavior

Discuss deployment implications of our proxy requirements.

7. Security Considerations

This entire document is about security. A future version of the document will highlight some important security concepts.

8. IANA Considerations

This document does not require actions by IANA.

9. Acknowledgments

We would like to thank Mayutan Arumaithurai and Klaas Wierenga for their feedback. Additionally, we would like to thank Eve Maler, Nicolas Williams, Bob Morgan, Scott Cantor, Jim Fenton, and Luke Howard for their feedback on the federation terminology question.

Furthermore, we would like to thank Klaas Wierenga for his review of the pre-00 draft version.

10. References

10.1. Normative References

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

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC3588] Calhoun, P., Loughney, J., Guttman, E., Zorn, G., and J. Arkko, "Diameter Base Protocol", RFC 3588, September 2003.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[RFC3579] Aboba, B. and P. Calhoun, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", RFC 3579, September 2003.

[RFC4072] Eronen, P., Hiller, T., and G. Zorn, "Diameter Extensible Authentication Protocol (EAP) Application", RFC 4072, August 2005.

[RFC4282] Aboba, B., Beadles, M., Arkko, J., and P. Eronen, "The Network Access Identifier", RFC 4282, December 2005.

[I-D.hansen-privacy-terminology] Pfitzmann, A., Hansen, M., and H. Tschofenig, "Terminology for Talking about Privacy by Data Minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management", draft-hansen-privacy-terminology-01 (work in progress), August 2010.

[I-D.ietf-abfab-gss-eap] Hartman, S. and J. Howlett, "A GSS-API Mechanism for the Extensible Authentication Protocol", draft-ietf-abfab-gss-eap-01 (work in progress), February 2011.

10.2. Informative References

[I-D.nir-tls-eap] Nir, Y., Sheffer, Y., Tschofenig, H., and P. Gutmann, "TLS using EAP Authentication", draft-nir-tls-eap-10 (work in

progress), February 2011.

[I-D.morris-privacy-considerations]
Aboba, B., Morris, J., Peterson, J., and H. Tschofenig, "Privacy Considerations for Internet Protocols", draft-morris-privacy-considerations-02 (work in progress), November 2010.

[RFC4017] Stanley, D., Walker, J., and B. Aboba, "Extensible Authentication Protocol (EAP) Method Requirements for Wireless LANs", RFC 4017, March 2005.

[RFC5106] Tschofenig, H., Kroeselberg, D., Pashalidis, A., Ohba, Y., and F. Bersani, "The Extensible Authentication Protocol-Internet Key Exchange Protocol version 2 (EAP-IKEv2) Method", RFC 5106, February 2008.

[RFC1964] Linn, J., "The Kerberos Version 5 GSS-API Mechanism", RFC 1964, June 1996.

[RFC2203] Eisler, M., Chiu, A., and L. Ling, "RPCSEC_GSS Protocol Specification", RFC 2203, September 1997.

[RFC3645] Kwan, S., Garg, P., Gilroy, J., Esibov, L., Westhead, J., and R. Hall, "Generic Security Service Algorithm for Secret Key Transaction Authentication for DNS (GSS-TSIG)", RFC 3645, October 2003.

[RFC2138] Rigney, C., Rigney, C., Rubens, A., Simpson, W., and S. Willens, "Remote Authentication Dial In User Service (RADIUS)", RFC 2138, April 1997.

[RFC4462] Hutzelman, J., Salowey, J., Galbraith, J., and V. Welch, "Generic Security Service Application Program Interface (GSS-API) Authentication and Key Exchange for the Secure Shell (SSH) Protocol", RFC 4462, May 2006.

[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.

[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.

[RFC5849] Hammer-Lahav, E., "The OAuth 1.0 Protocol", RFC 5849, April 2010.

[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.

[RFC2904] Vollbrecht, J., Calhoun, P., Farrell, S., Gommans, L., Gross, G., de Bruijn, B., de Laat, C., Holdrege, M., and D. Spence, "AAA Authorization Framework", RFC 2904, August 2000.

URIs

[1] <http://www.openid.net>

[2] <http://www.eduroam.org>

Authors' Addresses

Josh Howlett
JANET(UK)
Lumen House, Library Avenue, Harwell
Oxford OX11 0SG
UK

Phone: +44 1235 822363
Email: Josh.Howlett@ja.net

Sam Hartman
Painless Security

Phone:
Email: hartmans-ietf@mit.edu

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

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

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