N/A
Internet-Draft
Intended status: Informational
Expires: September 9, 2011

D. Brashear
OpenAFS Project

March 8, 2011

Authentication Name Mapping extension for AFS-3 Protection Service draft-brashear-afs3-pts-extended-names-09

Abstract

This document describes the extension of the format, use and communication of authentication names in the AFS-3 protocol to allow for additional authentication mechanisms to be represented and mapped to AFS IDs, independent of the AFS usernames currently used for management of PRDB entries. The new interface provides mechanisms for adding, removing, and listing mappings, and to allow the fileserver to map an authentication name to a PTS identity.

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 9, 2011.

Table of Contents

1. Introduction 3
2. Conventions Used in this Document 3
3. Background information on operation of AFS 3
4. Error Codes 4
5. Existing Protocol Constants 4
6. Existing Per-Identity Permission Flags 5
7. RPC Interface 5
7.1. New Data Types 5
7.2. Authentication Name Types 6
7.3. GetCapabilities RPC 6
7.4. Authentication name mapping RPCs 7
7.4.1. AuthNameToID 7
7.4.2. AuthNameToIDFallback 7
7.4.3. ListAuthNames 8
7.4.4. WhoAmI 8
7.4.5. AddAuthName 9
7.4.6. RemoveAuthName 9
8. Recommended Usage 9
9. Compatibility 10
10. Authentication Name Types 10
10.1. Kerberos V4 10
10.2. GSSAPI Export Names 10
10.3. Authentication Name Type Rewriting 10
10.3.1. Kerberos 4 Name Rewriting 10
10.3.2. Kerberos 5 Name Rewriting 10
11. Security Considerations 11
12. IANA Considerations 11
13. AFS-3 Registry Considerations 11
14. AFS3 Status 12
15. Acknowledgments 12
16. References 12
16.1. Normative References 12
16.2. Informational References 12
Author’s Address 12

1. Introduction

AFS-3 provides an authentication ID mapping service to map authentication system names in Kerberos Version 4 format to AFS-3 numeric identifiers. An augmented Kerberos 4 server was historically part of the AFS-3 protocol and service suite.

Security issues with Kerberos 4 as well as additional development in the space of authentication systems has created the need to map authentication names from other deployed systems to AFS identifiers. Some deployments provide several mechanisms to obtain AFS authentication. While mappings between Kerberos 4 and Kerberos 5 [RFC4120] authentication names allow use of most Kerberos 5 deployments with AFS, supporting more than a single realm requires matching usernames in all realms. Additionally, support for other systems is not provided at all.

An administrator or a user may know which authentication identities belong to the same individual, service, or role. However, not all deployments need to support interactive remapping; one use case could involve re-exporting another naming service via the AFS-3 PTS protocol, in which case only the name to identifier and identifier to name services would be provided.

The deployment implications for a mapping service are described below in Section 8.

2. Conventions Used in this Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Background information on operation of AFS

AFS is a distributed file system organized administratively into cells. In current practice, each AFS cell consists of one or more Volume Location Database (VLDB) servers, one or more Protection Service (PTS) servers, and one or more pairs of file servers and volume servers, plus possibly additional services not relevant to this document. Data stored in AFS is divided into collections of files called volumes. An AFS protocol client, when accessing a file within a specific AFS cell, first contacts a VLDB server for that cell to determine the file server for the AFS volume in which that file is located, and then contacts that file server directly to access the file. A client may also need to contact a PTS server for

that cell to register before accessing files in that cell. All communication is via remote procedure calls ('RPCs'), both between component servers and between clients and servers.

Because AFS provides for authenticated access to data, it is necessary to map authentication entities, generally corresponding to users of the AFS file service ('users') or services requiring access to data via the AFS file service ('services'), collectively clients of the AFS file service ('clients'), to AFS authentication identifiers ('AFS IDs'). The PTS service is used to provide this service to the file server when required, typically when a client first contacts a file server with a new authentication context, including unauthenticated. AFS originally included a Kerberos authentication (KA) server implementing the Kerberos 4 protocol as well as the AFS-3 KA protocol for obtaining authentication information. Because of this, PTS authentication names take the format typically used to represent a Kerberos 4 authentication identity as a string, namely, principal name, followed by a dot and instance if there is an instance, followed by an at-sign and the cell name if the cell is not the local cell.

4. Error Codes

AFS uses a library (com_err) and a tool from the MIT Project Athena suite (compile_et) to generate unique error codes protocol-wide. Codes are generated from a base value computed using a 4 or fewer character string known as an error table, identifying the subset of functionality which generated the error. All error codes are represented as signed 32 bit integers. The base value for PTS errors is 267264, generated from the error table name 'PT'. All RPCs when implemented are expected to return PTS errors. It is also possible for an implementation to support only part of this proposal. As such, any unimplemented RPCs should return error -455 (RXGEN_OPCODE), the error indicating an RPC is unsupported. No new errors are defined as part of this draft.

The existing error code PRPERM shall be used for permission errors due to insufficient privilege when attempting any of the described RPCs.

5. Existing Protocol Constants

When it is necessary to represent an identity which has provided either no authentication information, or information which does not include a corresponding AFS ID, an anonymous ID is defined and used.

```
#define      ANONYMOUSID     32766
```

The AFS ID 32766 shall be returned when an authentication name is unknown.

6. Existing Per-Identity Permission Flags

The PTS service includes a basic security model based around defined access flags. The flags are represented as 5 fields, each of which can have 3 values. One of these fields, the "s" field, is used to control who can get information about the entity. Valid values are 'S', which maps to PRP_STATUS_ANY, meaning anyone can get the information; 's', which maps to PRP_STATUS_MEM, meaning only members, the entity owner, and administrators can get the information; and '-' (or unset), meaning only the entity owner and administrators can get the information.

7. RPC Interface

Six new RPCs are defined for the AFS PTS service. Five of these are used to manipulate data related to authentication names, while the sixth is a general-purpose RPC to be used for capabilities indication. Additionally, several new data types are defined to allow representation of authentication names and identifiers in these new RPCs.

7.1. New Data Types

In order to represent a set of AFS IDs, the nidlist type is defined. AFS IDs are expected to be enhanced to allow a larger set of IDs concurrently with this advancement of this proposal.

```
typedef afs_int64 nidlist<>;
```

A list of AFS IDs, represented as 64 bit integers.

Authentication names are expected to be mechanism-specific. Thus an authentication-type agnostic data structure must be provided to represent these names. An implementing server MUST be able to use a bitwise comparison operation on the data portion of a PrAuthName. Support for all available name types is not expected.

```
#define AUTHDATAMAX 2048
#define AUTHPRINTABLEMAX 2048
struct PrAuthName {
  afs_int32 kind;
  opaque data<AUTHDATAMAX>;
  opaque display<AUTHPRINTABLEMAX>; };
```

It is expected that some mechanisms will provide name data which is not human-readable, or is any single format is sufficient to represent all possible mechanism authentication names. Therefore the PrAuthName data structure includes an integer tag denoting type, and an opaque data object representing the mechanism-specific authentication name data. The display portion is to be used for display to end-users, and MUST NOT be used for comparison purposes. If a display portion is not provided, the data portion MUST NOT be used directly for user display purposes. A client with knowledge of the particular name type used MAY use the data portion to derive a suitable display name, if none is provided.

```
typedef struct PrAuthName authnamelist<>;
```

Each AFS ID is expected to have more than one name mapped to it, possibly with more than one name of each type assigned to each. It is therefore necessary to support a list of PrAuthNames.

7.2. Authentication Name Types

The PrAuthName data type includes a 32 bit integer field to represent the kind of authentication being mapped. It is proposed that in addition to mappings for legacy Kerberos 4 based AFS names, that the first version of this additionally include mappings for Kerberos 5-based and GSSAPI-based authentication names.

```
#define PRAUTHTYPE_KRB4 1
#define PRAUTHTYPE_GSS 2
```

7.3. GetCapabilities RPC

As with other AFS services, the PTS service could be enhanced with the ability to represent new abilities to file servers and clients. We propose a new general-purpose RPC of the type implemented by other AFS services, namely, a bit stream. It is proposed that the first 32 bits be allocated to capabilities flags, while the remainder be reserved for future standardisation.

A complying server MUST implement GetCapabilities.

```
const PTSCAPABILITIESMAX = 196;
typedef afs_uint32 PrCapabilities<PTSCAPABILITIESMAX>;

const PTS_CAPABILITY_AUTHNAMEMAPPING       = 1;

GetCapabilities(
PrCapabilities *prcapabilities
) multi = 65536;
```

An implementing server should return any capabilities data it wishes to advertise. A server may choose to not advertise the same capabilities to all callers; For instance, the set of capabilities advertised to an authenticated caller may be different than the set advertised to an anonymous caller. In addition to the bitstream, return value is 0 for success, or a PTS error in the event of error.

7.4. Authentication name mapping RPCs

In addition to the general-purpose (GetCapabilities) RPC needed to represent this extended functionality, a complying server will include RPCs to represent the mapping of extended names to AFS IDs.

7.4.1. AuthNameToID

A complying server MUST implement the AuthNameToID RPC. This RPC is used to convert one or more authentication names, obtained by a file server or client in a mechanism-specific manner, to AFS IDs.

```
AuthNameToID(IN authnamelist *alist,
             OUT nidlist *ilist) = 65537;
```

For each authname provided in alist, an AFS ID will be provided in ilist at the corresponding position. Where an authentication name cannot be looked up, the AFS ID in list will be ANONYMOUSID.

On success of the call, 0 shall be returned. Success includes calls where none of the identities provided in authnamelist can be mapped to AFS IDs.

A client being used to manipulate mappings SHOULD use this RPC to discover the existing mapping, if any, for a given authentication name. A caller using the mapping service for service operation SHOULD instead call AuthNameToIDFallback.

7.4.2. AuthNameToIDFallback

A complying server MUST implement the AuthNameToIDFallback RPC. This RPC is used to convert one or more authentication names, obtained by

a file server or client in a mechanism-specific manner, to AFS IDs to be used in making authorization decisions related to the named authentication identities.

```
AuthNameToIDFallback(IN authnamelist *alist,
            OUT nidlist *ilist) = 65538;
```

For each authname provided in alist, an AFS ID will be provided in ilist at the corresponding position. Implicit fallback mappings will be used to map identities where no explicit mapping is provided. Where neither an explicit nor implicit mapping is available, the AFS ID in ilist will be ANONYMOUSID.

An explicit mapping is one that can be configured, represented, and returned by the protocol defined in this document. An implicit mapping is site- or implementation-specific. Examples of implicit mapping would be translation of a client authenticating as Kerberos 5 principal user@REALM to a PTS name of user, and of a client authenticating as Kerberos 5 principal

On success of the call, 0 shall be returned. Success includes calls where none of the identities provided in authnamelist can be mapped to AFS IDs.

A caller using the mapping service for service operation SHOULD use this RPC. A client being used to manipulate mappings SHOULD use AuthNameToID instead.

7.4.3. ListAuthNames

A complying server SHOULD implement the ListAuthNames RPC. This RPC is used to list all authentication names attached to the provided AFS ID.

```
ListAuthNames(IN afs_int64 id, OUT authnamelist *alist)
            = 65539;
```

On success, 0 shall be returned, along with the list of authentication names in alist corresponding to the AFS ID provided in id.

7.4.4. WhoAmI

A complying server SHOULD implement the WhoAmI RPC. This RPC is used to return the current authentication name and AFS ID for a caller to the caller.

```
WhoAmI(OUT afs_int64 id, OUT struct PrAuthName *alist) = 65540;
```

On success, 0 shall be returned, along with the identity and authentication name corresponding to the caller of the RPC.

7.4.5. AddAuthName

A complying server MAY implement the AddAuthName RPC. This RPC is used to add an authentication name to an existing AFS ID.

```
AddAuthName(IN afs_int64 id, IN PrAuthName *aname) = 65541;
```

On success, 0 shall be returned. If the authentication name provided is already mapped to another AFS ID, PREXIST shall be returned. If the AFS ID specified does not exist, PRNOENT shall be returned.

7.4.6. RemoveAuthName

A complying server MAY implement the RemoveAuthName RPC. This RPC is used to remove an authentication name from an existing AFS ID.

```
RemoveAuthName(IN PrAuthName *aname) = 65542;
```

On success, 0 shall be returned. If the authentication name provided is not mapped, PRNOENT shall be returned.

8. Recommended Usage

Two classes of service providers are possible. In one, where data is provided in a read-only fashion from an alternate authorization service, only the AuthNameToID and ListAuthNames RPCs are required. In the other, all RPCs are needed, such that mappings can be constructed and maintained.

The ListAuthNames operation should be permitted for administrators as well as for the owner of the entry to be listed. If the 'S' (PRP_STATUS_ANY) bit is set on the entry, then other users may also perform this operation.

The AddAuthName and RemoveAuthName operations should always be permitted for administrators, and should normally also be permitted for the owner of the entry to be updated. For RemoveAuthName, the entry being updated is always the one to which the specified name currently maps, and should be an identity which is not the one being removed. Regardless, any given authentication name can map only to one ID; attempting to add (via AddAuthName) a second mapping for the same authentication name MUST fail.

The AuthNameToID operation is analogous to the existing NameToID

operation, and like that, should be able to be used by anyone.

9. Compatibility

An implementation of the AFS protection service protocol implementing these extensions shall indicate this by including in the response to a GetCapabilities PTS RPC the capability flag PTS_CAPABILITY_AUTHNAMEMAPPING.

10. Authentication Name Types

While the authentication name is an opaque type with respect to the AFS-3 protocol, it is recommended that the following definitions be used for the proposed types described herein.

10.1. Kerberos V4

The format of the Kerberos V4 name type is either name@REALM or name.inst@REALM, depending on whether a non-null instance is present. In both cases the trailing NUL character is *not* part of the authentication name data.

10.2. GSSAPI Export Names

The format of the GSSAPI name type is that described in section 3.2 of [RFC2743]. The data opaque object will contain the GSSAPI canonical name as generated by GSS_Export_name after GSS_Canonicalize_name. The display opaque object will contain the GSSAPI display name type and name string as generated by GSS_Display_name. The display name MUST only be used for printing; the canonical name MUST NOT be used for printing.

10.3. Authentication Name Type Rewriting

10.3.1. Kerberos 4 Name Rewriting

When Kerberos 4 is used, the Kerberos name type must be used. In particular, Kerberos 4 principal names MUST NOT be represented as Kerberos 5 names or vice versa.

10.3.2. Kerberos 5 Name Rewriting

When Kerberos 5 is used, with or without the GSSAPI, Kerberos 5 principal names MUST be represented using the GSSAPI export name type with the Kerberos mechanism OID. Servers accepting Kerberos 5 without GSSAPI MUST convert to a GSSAPI Kerberos 5 export name using

the Kerberos mechanism OID before calling AuthNameToID. This is intended to allow all users of Kerberos 5 as an authentication mechanism, regardless of bindings, to use the same authentication name. Implementations which do not support GSSAPI can support this by preparing a name as specified in section 2.1.1 of [RFC1964], subject to the constraints in section 2.1.3. This string, when prefixed with the sequence of octets

04 01 00 0B 06 09 2A 86 48 86 F7 12 01 02 02 HX XX XX XL

where

HX XX XX XL

represent the network byte order hex encoding of the length of the string, can be treated as the canonicalized export name, while the string itself can be treated as the display name.

11. Security Considerations

Allowing AuthNameToID to be used by any caller exposes information about whether an authentication name is mapped at all, or, indeed, exists. In some environments, this information may be considered privileged.

Allowing a user to add arbitrary mappings to their identity via AddAuthName may allow unintended permissions to be granted if the user makes a mistake when mapping identities to the AFS identity in question.

Because a server is not required to know about all available name types, display names are provided by the caller to AddAuthName. It is possible to provide an erroneous display name whether for malicious or benign reasons. In either case, making a decision based on the display name may result in problems.

12. IANA Considerations

This document doesn’t require any IANA registrations.

13. AFS-3 Registry Considerations

This document requires the registration of the new RPCs as provided in the section Section 7 above, as well as new registries for PTS capabilities and for name types, also as above.

14. AFS3 Status

This document has been adopted by the AFS3 Standards Group as experimental.

15. Acknowledgments

The author thanks Magnus Ahltorp, Love Hoernquist Aestrand, and Jeffrey T. Hutzelman for their work on the scope of this enhancement, the attendees of the 2009 AFS hackathon in Edinburgh, notably Marcus Watts and Simon Wilkinson, for their assistance in defining the name object for GSSAPI, and Jeffrey T. Hutzelman and Simon Wilkinson discussion of an acceptable means for representing Kerberos 5 names consistently regardless of mechanism.

16. References

16.1. Normative References

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

16.2. Informational References

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

[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.

Author’s Address

Derrick J. Brashear
OpenAFS Project
2829 Larkins Way
Pittsburgh, PA 15203
USA

Email: shadow@openafs.org