Network Working Group
Internet-Draft
Expires: March 4, 2007

S. Hartman
MIT

August 31, 2006

Desired Enhancements to GSSAPI Version 3 Naming
draft-ietf-kitten-gss-naming-05.txt

Status of this Memo

Copyright Notice

Abstract

The Generic Security Services API (GSS-API) provides a naming architecture that supports name-based authorization. GSS-API authenticates two named parties to each other. Names can be stored on access control lists to make authorization decisions. Advances in security mechanisms and the way implementers wish to use GSS-API require this model to be extended for the next version of GSS-API. As people move within an organization or change their names, the name authenticated by GSS-API may change. Using some sort of constant identifier would make ACLs more stable. Some mechanisms such as public-key mechanisms do not have a single name to be used across all environments. Other mechanisms such as Kerberos may include group membership or role information as part of authentication. This document motivates extensions to GSS-API naming and describes the extensions under discussion.

Table of Contents

1. Introduction

The Generic Security Services API [2] authenticates two named parties to each other. GSS names can be imported in a variety of formats through the gss_import_name call. Several mechanism-independent name formats are provided including GSS_C_NT_HOSTBASED_SERVICE for services running on an Internet host and GSS_C_NT_USER_NAME for the names of users. Other mechanism-specific name types are also provided. By the time a name is used in acquiring a mechanism-specific credential or establishing a security context, it has been transformed into one of these mechanism-specific name types. In addition, the GSS-API provides a function called gss_export_name that will transform a GSS-API name into a binary blob suitable for comparisons. This binary blob can be stored on ACLs and then authorization decisions can be made simply by comparing the name exported from a newly accepted context to the name on the ACL.

Storing names on ACLs can be problematic because names tend to change over time . If the name contains organizational information such as a domain part or an indication of what department someone works for, this changes as the person moves around the organization. Even if no organizational information is included in the name, the name will change as people change their names. Updating ACLs to reflect name changes is difficult. Another significant problem is that names can be reused to apply to an entity other than the entity to which they originally applied. For example if a Unix user ID is placed on an ACL, the account deleted and then a new user assigned the old ID, then that new user may gain privileges intended for the old user.

Inherent in the GSS naming model is the idea that mechanism names need to be able to be represented in a single canonical form. Anyone importing that name needs to be able to retrieve the canonical form of that name.

Several security mechanisms have been proposed for which this naming architecture is too restrictive. In some cases it is not always possible to canonicalize any name that is imported. In other cases there is no single canonical name.

Also, as GSS-API is used in more complex environments, there is a desire to use attribute certificates [6], Kerberos authorization data [3], or other non-name-based authorization models. GSS-API needs to be enhanced in order to support these uses in a mechanism-independent manner.

This document discusses the particular naming problems with two important classes of GSS-API mechanisms. It also discusses the set of proposed solutions and open issues with these solutions. This

draft limits discussion to these solutions and provides a description of the problem against which the solutions can be judged. These solutions are targeted for incorporation into GSS-API Version 3.

2. Kerberos Naming

The Kerberos mechanism demonstrates both the naming stability problem and the authorization extension problem.

The Kerberos Referrals draft [4] proposes a new type of Kerberos name called an enterprise name. The intent is that the enterprise name is an alias that the user knows for themselves and can use to login. The Kerberos KDC translates this name into a normal Kerberos principal and gives the user tickets for this principal. This normal principal is used for authorization. The intent is that the enterprise name tracks the user as they move throughout the organization, even if they move to parts of the organization that have different naming policies. The name they type at login remains constant, but the Kerberos principal used to authenticate them to services changes.

Unauthenticated services cannot generally perform a mapping from enterprise name to principal name. Even authenticated services may not be authorized to map names other than the name of the authenticated service. Also, Kerberos does not (and does not plan to) provide a mechanism for mapping enterprise names to principals besides authentication as the enterprise name. Thus, any such mapping would be vendor-specific. With this feature in Kerberos, it is not possible to implement gss_canonicalize_name for enterprise name types. Of course other names types such as traditional principal names could be used for GSS-API applications. Naturally this loses the benefits of enterprise names.

Another issue arises with enterprise names. In some cases, it would be desirable to put the enterprise name on the ACL instead of a principal name for greater ACL stability. At first glance this could be accomplished by including the enterprise name in the name exported by gss_export_name. Unfortunately, if this were done, the exported name would change whenever the mapping changed, invalidating any ACL entries based off the old exported name and defeating the purpose of including the enterprise name in the exported name. In some cases it would be desirable to have the exported name be based on the enterprise name and in others based on the principal name, but this is not permitted by the current GSS-API.

Another development also complicates GSS-API naming for Kerberos. Several vendors have been looking at mechanisms to include group membership information in Kerberos authorization data. It is desirable to put these group names on ACLs. Again, GSS-API currently has no mechanism to use this information.

3. X.509 Names

X.509 names are more complicated than Kerberos names. In the Kerberos case there is a single principal carried in all Kerberos messages. X.509 certificates have multiple options. It seems the subject name might be the appropriate name to use as the name to be exported in a GSS-API mechanism. However RFC 3280 [5] allows the subject name to be an empty sequence in end-entity certificates. Therefore the subjectAltName extension might be the only portion of the certificate that identifies the subject. As in the case of Kerberos group memberships, there may be many subjectAltName extensions available in a certificate. Different applications will care about different name forms. One possible candidate for an exported name would be all the names from the subject field and the subjectAltName extension from a certificate. However as new names are added then existing ACL entries would be invalidated; this is undesirable. Thus there is no single value that can be defined as the exported GSS-API name that will be useful in all environments.

A profile of a particular X.509 GSS-API mechanism could require a specific name be used. However this would limit that mechanism to require a particular type of certificate. There is interest in being able to use arbitrary X.509 certificates with GSS-API for some applications.

Experience so far has not lead to sufficient interoperability with GSS-API X.509 mechanisms. Even if the subject name is used, there is ambiguity in how to handle sorting of name components. Martin Rex said that he was aware of several SPKM [1] implementations but no two were fully interoperable on names.

Also, as discussed in the introduction, it is desirable to support X.509 attribute certificates.

4. Composite Names

One proposal to solve these problems is to extend the concept of a GSS-API name to include a set of name attributes. Each attribute would be an octet-string labeled by an OID. Examples of attributes would include Kerberos enterprise names, group memberships in an authorization infrastructure, Kerberos authorization data attributes and subjectAltName attributes in a certificate. Several new operations would be needed:

1. Add an attribute to name.

2. Query attributes of name.

3. Query values of an attribute.

4. Delete an attribute from a name.

5. Export a complete composite name and all its attributes for transport between processes.

Note that an exported composite name would not generally be suitable for binary comparison. Avoiding confusion between this operation and the existing gss_export_name operation will require careful work. However many attributes of composite names will be appropriate for binary comparisons. Such attributes can be used on ACLs just as exported names are used on ACLs today. For example if a particular SubjectAltName extension contains the appropriate identity for an application, then the name attribute for this SubjectAltName can be placed on the ACL. This is only true if the name attribute is stored in some canonical form.

Additional utility operations will probably be needed depending on the implementation of name attributes.

4.1. Usage of Name Attributes

Since attributes are part of GSS-API names, the acceptor can retrieve the attributes of the initiator's and acceptor's name from the context. These attributes can then be used for authorization.

Most name attributes will probably not come from explicit operations to add attributes to a name. Instead, name attributes will probably come from mechanism-specific credentials. Components of these mechanism-specific credentials may come from platform or environment-specific names. mechanism-specific naming and group membership can be mapped into name attributes by the mechanism implementation. The specific form of this mapping will generally require protocol

specification for each mechanism.

4.2. Open issues

This section describes parts of the proposal to add attributes to names that will need to be explored before the proposal can become a protocol specification.

Are mechanisms expected to be able to carry arbitrary name attributes as part of a context establishment? At first it seems like this would be desirable. However the purpose of GSS-API is to establish an authenticated context between two peers. In particular, a context authenticates two named entities to each other. The names of these entities and attributes associated with these names will be used for authorization decisions. If an initiator or acceptor is allowed to assert name attributes and the authenticity of these assertions is not validated by the mechanisms, then security problems will result. On the other hand, requiring that name attributes be mechanism-specific and only be carried by mechanisms that understand the name attributes and can validate them compromises GSS-API's place as a generic API. Application authors would be forced to understand mechanism-specific attributes to make authorization decisions. In addition if mechanisms are not required to transport arbitrary attributes, then application authors will need to deal with different implementations of the same mechanism that support different sets of name attributes. One possible solution is to carry a source along with each name attribute; this source could indicate whether the attribute comes from a mechanism data structure or from the other party in the authentication.

Another related question is how will name attributes be mapped into their mechanism-specific forms. For example it would be desirable to map many Kerberos authorization data elements into name attributes. In the case of the Microsoft PAC, it would be desirable for some applications to get the entire PAC. However in many cases, the specific lists of security IDs contained in the PAC would be more directly useful to an application. So there may not be a good one-to-one mapping between the mechanism-specific elements and the representation desirable at the GSS-API layer.

Specific name matching rules need to be developed. How do names with attributes compare? What is the effect of a name attribute on a target name in gss_accept_sec_context?

4.3. Handling gss_export_name

For many mechanisms, there will be an obvious choice to use for the name exported by gss_export_name. For example in the case of

Kerberos, the principal name can continue to be used as the exported name. This will allow applications depending on existing GSS-API name-based authorization to continue to work. However it is probably desirable to allow GSS-API mechanisms for which gss_export_name cannot meaningfully be defined. The behavior of gss_export_name in such cases should probably be to return some error. Such mechanisms may not work with existing applications and cannot conform to the current version of the GSS-API.

5. Credential Extensions

An alternative to the name attributes proposal is to extend GSS-API credentials with extensions labeled by OIDs. Interfaces would be needed to manipulate these credential extensions and to retrieve the credential extensions for credentials used to establish a context. Even if name attributes are used, credential extensions may be useful for other unrelated purposes.

It is possible to solve problems discussed in this document using some credential extension mechanism. Doing so will have many of the same open issues as discussed in the composite names proposal. The main advantage of a credential extensions proposal is that it avoids specifying how name attributes interact with name comparison or target names.

The primary advantage of the name attributes proposal over credential extensions is that name attributes seem to fit better into the GSS-API authorization model. Names are already available at all points when authorization decisions are made. In addition, for many mechanisms the sort of information carried as name attributes will also be carried as part of the name in the mechanism

6. Mechanisms for Export Name

Another proposal is to define some GSS-API mechanisms whose only purpose is to have an exportable name form that is useful. For example, you might be able to export a name as a local machine user ID with such a mechanism.

This solution works well especially for name information that can be looked up in a directory. It was unclear whether this solution would allow mechanism-specific name information to be extracted from a context. If so, then this solution would meet many of the goals of this document.

One advantage of this solution is that it requires few if any changes to GSS-API semantics. It is not as flexible as other solutions. Also, it is not clear how to handle mechanisms that do not have a well defined name to export with this solution.

7. Selection of Source Identity

Today, applications such as e-mail clients and web browsers require connections to multiple targets. For each target there may be one or more source identities that is appropriate for the connection. Currently each application must choose the source name to use when acquiring credentials or initiating a security context. However the rules that applications use can be generalized to a large extent. GSS-API could simplify application design and implementation by taking a larger role in selection of source identity to use when connecting to a particular target.

Currently GSS-API credentials represent a single mechanism name. that is, by the time credentials are acquired, they must act as if a particular single identity is chosen for each mechanism in the credential. All these identities must correspond to a single mechanism independent name.

Two possibilities have been proposed for involving GSS-API in the selection of source identities. First, the restriction that a mechanism name must be chosen when credentials are acquired could be relaxed. Some name form would need to be used, but this name form could represent a set of possibilities. The particular identity would be chosen when context establishment happened. This could involve information received from the target in identity selection.

Another possibility is to provide a mechanism to acquire credentials and to provide information about the target when credentials are acquired. This would be much less of a change to GSS-API but would not allow information received from the target to choose identity selection.

With both approaches, information to communicate the needs of the application to the GSS-API mechanism will be required. For example, hinting about whether information can be cached and about the scope of cache entries is required.

Another possibility can be implemented in GSS-API V2 today. Do not bind the credentials to a mechanism name until either the credentials are queried or they are used to set up a context. This is undesirable because if an application uses the credential inquiry interface then it will get different behavior than cases where this interface is not used. For this reason, the working group favors an extension to GSS-API V3.

8. Compatibility with GSS-API V2

In order to avoid breaking existing applications or mechanisms the following backward compatibility requirements need to be met:

1. Existing APIs must continue to behave as they do in GSS-API V2.

2. GSS-API V2 mechanisms must produce the same exported name forms; composite names cannot change the existing exported name forms.

3. Extensions add new optional behavior.

If GSS-API V3 mechanisms are more permissive than GSS-API V2 mechanisms then care must be taken so that GSS-API V2 applications do not select these mechanisms.

9. Security Considerations

GSS-API sets up a security context between two named parties. The GSS-API names are security assertions that are authenticated by the context establishment process. As such the GSS naming architecture is critical to the security of GSS-API.

Currently GSS-API uses a simplistic naming model for authorization. Names can be compared against a set of names on an access control list. This architecture is relatively simple and its security properties are well understood. However it does not provide the flexibility and feature set for future deployments of GSS-API.

This proposal will significantly increase the complexity of the GSS naming architecture. As this proposal is fleshed out, we need to consider ways of managing security exposures created by this increased complexity.

One area where the complexity may lead to security problems is composite names with attributes from different sources. This may be desirable so that name attributes that carry their own authentication. However the design of any solutions needs to make sure that applications can assign appropriate trust to name components.

10. Acknowledgements

John Brezak, Paul Leach and Nicolas Williams all participated in discussions that lead to a desire to enhance GSS naming. Martin Rex provided descriptions of the current naming architecture and pointed out many ways in which proposed enhancements would create interoperability problems or increase complexity. Martin also provided excellent information on what aspects of GSS naming have tended to be implemented badly or have not met the needs of some customers.

Nicolas Williams helped describe the possible approaches for enhancing naming.

11. Informative References

[1] Adams, C., "The Simple Public-Key GSS-API Mechanism (SPKM)", rfc 2025, October 1996.

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

[3] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", draft-ietf-krb-wg-kerberos-clarifications-06.txt (work in progress), June 2004.

[4] Jaganathan , K., Zhu, L., Swift, M., and J. Brezak, "Generating KDC Referrals to locate Kerberos realms", draft-ietf-krb-wg-kerberos-referrals-03.txt (work in progress), 2004.

[5] Housley, R., Polk, W., Ford, W., and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", rfc 3280, April 2002.

[6] Farrell, S. and R. Housley, "An Internet Attribute Certificate Profile for Authorization.", rfc 3281, April 2002.

Author's Address

Sam Hartman
MIT

Email: hartmans@mit.edu

Intellectual Property Statement

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.