Network Working Group
INTERNET-DRAFT
draft-ietf-ipsec-isakmp-gss-auth-07.txt

Derrell Piper
Nokia
Brian Swander
Microsoft
July 14, 2001

A GSS-API Authentication Method for IKE
<draft-ietf-ipsec-isakmp-gss-auth-07.txt>

Status of this Memo

Table of Contents

1. Abstract....2
2. Terms and Definitions....2
3. Discussion....4
3.1 SKEYID Generation for GSS-API....6
3.2 IKE Phase 1 Authentication for GSS-API....7
3.3 GSS-API Identifiers: Method, Attribute, and Payload....8
3.4 The GSS-API Authentication Method Vendor ID Signature....10
4. Change Log....10
5. Security Considerations....11
Acknowledgments....12

References.....12
Authors' Address.....12

1. Abstract

This document describes an alternate authentication method for IKE which makes use of GSS-API to authenticate the Diffie-Hellman exchange. The mechanism described here extends the authentication methods defined in RFC-2409 without introducing any modifications to the IKE key exchange protocol.

It also documents the Microsoft Windows 2000 implementation of this protocol, which uses Kerberos via the Microsoft SSPI interface to authenticate Windows 2000 machines within a Window 2000 domain, within trusted Windows 2000 domains, and when Windows 2000 is operating in MIT-Kerberos compatibility mode, such that Windows 2000 systems are members of the MIT-KDC realm or the MIT-KDC realm has Kerberos trust with the Windows 2000 domain.

For a list of changes since the previous version of this document, please see Section 4.

2. Terms and Definitions

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in [RFC 2119].

2.1 Notation

RFC-2409 uses the following notation throughout that draft. That notation is included here along with a few additions.

HDR is an ISAKMP header whose exchange type is the method. When written as HDR* it indicates payload encryption.

SA is an SA negotiation payload with one or more proposals. An initiator MAY provide multiple proposals for negotiation; a responder MUST reply with only one.

<P>_b indicates the body of payload <P>-- the ISAKMP generic payload is not included.

SAi_b is the entire body of the SA payload (minus the ISAKMP generic header)-- i.e. the DOI, situation, all proposals and all transforms offered by the Initiator.

CKY-I and CKY-R are the Initiator's cookie and the Responder's cookie, respectively, from the ISAKMP header.

g^xi and g^xr are the Diffie-Hellman public values of the

initiator and responder respectively.

g^xy is the Diffie-Hellman shared secret.

GIi and GIr are identity name strings for the GSS-API initiator and responder GSS-API endpoints. These name strings are private to GSS-API.

GSSi and GSSr are initiator and responder GSS-API tokens generated by the local GSS-API's using GSS_Init_sec_context and GSS_Accept_sec_context respectively.

GSSi(n) and GSSr(n) are optional tokens which may be included for additional GSS-API token exchanges in IKE Main Mode when either side encounters GSS_S_CONTINUE_NEEDED from its underlying GSS-API mechanism.

KE is the key exchange payload which contains the public information exchanged in a Diffie-Hellman exchange. There is no particular encoding used for the data of a KE payload.

Nx is the nonce payload; x can be: i or r for the ISAKMP initiator and responder respectively.

IDx is the identity payload for "x". x can be: "ii" or "ir" for the ISAKMP initiator and responder respectively during phase one negotiation; or "ui" or "ur" for the user initiator and responder respectively during phase two. The ID payload format for the Internet DOI is defined in RFC-2407.

HASH (and any derivative such as HASH(2) or HASH_I) is the hash payload. The contents of the hash are specific to the authentication method.

prf(key, msg) is the keyed pseudo-random function-- often a keyed hash function-- used to generate a deterministic output that appears pseudo-random. prf's are used both for key derivations and for authentication (i.e. as a keyed MAC).

SKEYID is a string derived from secret material known only to the active players in the exchange.

SKEYID_e is the keying material used by the ISAKMP SA to protect it's messages.

SKEYID_a is the keying material used by the ISAKMP SA to authenticate it's messages.

SKEYID_d is the keying material used to derive keys for non-ISAKMP security associations.

<x>y indicates that "x" is encrypted with the key "y".

--> signifies "initiator to responder" communication (requests).

<-- signifies "responder to initiator" communication (replies).

| signifies concatenation of information-- e.g. X | Y is the concatenation of X with Y.

[x] indicates that x is optional.

< x | y > indicates that one of "x" or "y" will be chosen.

(n) indicates that this is the n-th instance of this item.

2.2 Payload Encryption

Payload encryption (when noted by a '*' after the ISAKMP header) MUST begin immediately after the ISAKMP header. When communication is protected, all payloads following the ISAKMP header MUST be encrypted. Encryption keys are generated from SKEYID_e in a manner that is defined for each algorithm.

3. Discussion

The ISAKMP/Oakley resolution document (RFC-2409) defines a key negotiation protocol that blends the Oakley key determination protocol (RFC-2412) with ISAKMP (RFC-2408) to provide authenticated cryptographic key exchange for use with IP security protocols (e.g. AH/ESP). The IKE negotiation includes an authentication method negotiation which is used to select a scheme to be used for authenticating a Diffie-Hellman key exchange. There are currently five defined authentication methods: pre-shared key, DSS signature, RSA signature, and two forms of RSA encryption. This document defines a new method that uses the Generic Security Services API ([Linn98]) to provide the necessary authentication.

The GSS-API abstraction is that a host operating system provides an API to applications that request security services (e.g. integrity protection or confidentiality) through a formal interface (e.g., [Wray98]). GSS-API provides opaque tokens to applications which are responsible for sending the tokens to peer GSS-API implementations, presumably on remote hosts. A by-product of any GSS-API exchange is a one way or mutual authentication using whatever authentication scheme the application chose to bind to when GSS-API was initialized

(or whatever was negotiated by SPNEGO (RFC-2478)). Typical authentication packages include Kerberos and SSL.

The ISAKMP/Oakley resolution defines a Main Mode and an Aggressive Mode for establishing Security Associations (SA's) between IPSEC hosts. These modes have a fixed set of round-trips: 4.5 or 5 for Main Mode and 1 or 1.5 for Aggressive (depending on whether the Commit bit (RFC-2408, Section 3.1) is used by the responder).

When using GSS-API, there's a separate protocol being run by the GSS-API packages on the initiator and on the responder. (Initiator and responder are ISAKMP terms, both are GSS-API clients.) The basic model is that the IKE initiator calls GSS_Init_sec_context (with mutual_req_flag) to construct a GSS-API token and sends this along with the KE and nonce in the second Main Mode exchange. The responder calls GSS_Accept_sec_context on this token and sends the output of GSS_Accept_sec_context (another token) back along with his KE and his nonce. On receipt of the responder's token, the initiator calls GSS_Init_sec_context a second time to complete the mutual authentication. Finally, each side exchanges a HASH payload which has been wrapped using GSS_Wrap. Successfully calling GSS_Unwrap to unwrap the HASH payloads along with verifying the hashes proves possession of the GSS-API shared secret and authenticates the Diffie-Hellman exchange.

GSS-API requires that a client identify the target GSS-API endpoint by name. If the initiator does not already know the GSS-API endpoint name of the ISAKMP target, a new Phase 1 attribute can be used to exchange endpoint names during the first Main Mode round trip (Section 3.2). Note that these name string are bound to the exchange but otherwise unauthenticated. The GSS-API endpoint names are also assumed to be opaque.

For Windows 2000 compatibility, these opaque blobs are encoded as unicode strings. For instance, for machine 'briansw' in domain 'IPSEC.MICROSOFT.COM', the identity is 'briansw@IPSEC.MICROSOFT.COM' (in unicode). This identity is just a particular example, and it should not be assumed that the GSS-API identity is necessarily the machine name + domain name.

Since the GSS-API tokens are exchanged during Phase 1 along with the KE payloads, they are not protected by the (yet to be formed) ISAKMP SA. To prevent a cut/paste attack on the GSS-API tokens, it's therefore necessary to include the tokens in the HASH_I and HASH_R computation (Section 3.1). This binds the tokens to a particular ISAKMP exchange. If used, the GSS Identity Name strings MUST also be included in these hash calculations.

In addition, the output from the prf for each hash is wrapped using GSS_Wrap. Upon receipt of either hash payload, each side MUST successfully call GSS_Unwrap. This proves possession of the GSS-API shared secret by each peer and prevents an active man-in-the-middle attack from simply forwarding on the GSS-API tokens. The choice of whether to use integrity protection only or integrity with confidentiality is somewhat mechanism specific. However, since the strength of the algorithm chosen necessarily determines the outcome of the authentication for ISAKMP, the strongest possible protection SHOULD be chosen. The following flags should be specified to GSS_Init_sec_context on the initiating side:

Flag	Requirement
mutual_req_flag	MUST
integ_req_flag	MUST
conf_req_flag	SHOULD

The number of messages in this protocol is dictated by whether or not either endpoint chooses to return GSS_S_CONTINUE_NEEDED. Depending on this, a message could be one of two possible outcomes. This choice in denoted by < opt1 | opt2>. For instance, in Main Mode, the Responder's third message may be either another GSS token or his final HASH payload. This is denoted as, < GSSr(n) | HASH_R >.

3.1 SKEYID Generation for GSS-API

RFC-2409 defines several authentication methods for Main Mode or Aggressive Mode -- digital signatures, authentication using public key encryption, and pre-shared keys. This document introduces another and defines the value of SKEYID for GSS-API authentication as follows.

For GSS-API: SKEYID = prf(Ni_b | Nr_b, g^xy)

To authenticate either exchange the initiator of the protocol generates HASH_I and the responder generates HASH_R where:

```
HASH_I = GSS_Wrap(prf(SKEYID, g^xi | g^xr | CKY-I | CKY-R | SAi_b |
             IDii_b [ | GIi ] | GSSi [ | GSSi(n) ... ]))

HASH_R = GSS_Wrap(prf(SKEYID, g^xr | g^xi | CKY-R | CKY-I | SAi_b |
             IDir_b [ | GIr ] | GSSr [ | GSSr(n) ... ]))
```

For authentication using GSS-API, the GSS-API package on either side provides authentication of the GSS-API identities, and HASH_I and HASH_R are used to bind the GSS-API identities and tokens to the Main Mode exchange. The GSS_Wrap (and subsequent GSS_Unwrap) proves

possession of the GSS-API shared secret for each peer. The initiator MUST specify the mutual_req_flag to request mutual authentication between the two GSS-API packages. A provision is defined for the GSS-API peers to exchange GSS-API identities during Main Mode, at the expense of identity protection for the GSS-API endpoint identities.

The content of the HASH_I and HASH_R ISAKMP payloads are the output tokens from GSS_Wrap. The input to GSS_Wrap is the output of the negotiated IKE hash function (prf) over the specified data. In other words, you take the data, hash it with the negotiated hash function, and then call GSS_Wrap on the hash digest. The output of GSS_Wrap is placed in the HASH_I and HASH_R payloads.

When the optional GSSi(n) and GSSr(n) tokens are sent in a Main Mode exchange (see Section 3.2). All of the GSS-API tokens exchanged MUST be included in the subsequent HASH_I/HASH_R calculations defined above.

3.2 IKE Phase 1 Authentication for GSS-API

Using GSS-API, the ancillary information exchanged during the second round-trip are GSS-API tokens; the exchange is authenticated in GSS-API and the GSS-API tokens are bound to the exchange using HASH_I and HASH_R.

If the GSS-API requires that the initiator and responder have prior knowledge of the GSS-API endpoint names for each peer, this information may be exchanged during the first round trip (by including the GSS Identity Name attribute in the SA) at the expense of identity protection for the GSS-API endpoints. When the GSS-API requires the exchange of identity names, Aggressive Mode cannot be used. For Windows 2000 compatibility, these entities MUST be exchanged.

Additionally, the local GSS-API may choose to make use of additional GSS-API token exchanges, using the optional GSSi2 and GSSr2 tokens, based on local criteria. For example, a GSS-API implementation using Kerberos may choose to make use of an extra round-trip for clock synchronization reasons. These extra round-trips can only be done in Main Mode. When extra messages are used, the HASH_I computation is deferred until each side is "done".

Main Mode using GSS-API is defined as

```
   Initiator                          Responder
  -----------                        -----------
   HDR, SA                    -->
                              <--     HDR, SA
```

```
        HDR, KE, Ni, GSSi           -->
                                    <--    HDR, KE, Nr, GSSr
        HDR*, IDii,
          < GSSI(n) | HASH_I>       -->
                                    <--    HDR*, IDir,
                                             < GSSR(n) | HASH_R >
     [ HDR*, < GSSI(n) | HASH_I >   -->
                                    <--    HDR*, <GSSR(n) | HASH_R > ]
```

The Main Mode exchange terminates when each side has generated and sent their corresponding HASH token and has successfully processed the other side's HASH token. The HASH token is generated when the underlying GSS-API mechanism returns GSS_S_COMPLETE (as opposed to GSS_S_CONTINUE_NEEDED). The receipt of a HASH token necessarily indicates that the peer is prepared to terminate the GSS-API exchange.

Aggressive Mode using GSS-API is defined as

```
        Initiator                       Responder
       -----------                     -----------
        HDR, SA, KE, Ni,
          IDii, GSSi            -->
                                <--     HDR, SA, KE, Nr,
                                          IDir, GSSr, HASH_R
        HDR, HASH_I             -->
```

Aggressive Mode works only for a single token exchange. If either the initiator's second call or any of the responder's calls encounter GSS_S_CONTINUE_NEEDED, Aggressive Mode cannot be used and each side should fall back to Main Mode. When this occurs, the side encountering the unexpected GSS_S_CONTINUE_NEEDED MUST send an ISAKMP Notify (UNSUPPORTED-EXCHANGE-TYPE) and terminate the Aggressive Mode exchange.

3.3 GSS-API Identifiers: Authentication Method, Attribute, and Payload

Implementations using the GSS-API Authentication Method will need to agree on the values for the following items, after exchanging recognizable ISAKMP Vendor ID payloads (Section 3.4).

3.3.1 Authentication Method (IKE)

GSS-API using Kerberos	65001
Generic GSS-API	65002
GSS-API with SPNEGO	65003
GSS-API using SPKM	65004

Generic GSS-API

Specifies generic GSS-API authentication. The underlying GSS-API implementation is not constrained to use any particular mechanism. The two parties must agree on the underlying mechanism using some out-of-band method.

GSS-API with SPNEGO

Specifies GSS-API authentication using The Simple and Protected GSS-API Negotiation Mechanism [RFC2478]. SPNEGO ensures that the two parties agree upon a mutually acceptable mechanism.

GSS-API using Kerberos

Specifies GSS-API authentication using The Kerberos Version 5 GSS-API Mechanism [RFC1964].

GSS-API using SPKM

Specifies GSS-API authentication using The Simple Public-Key GSS-API Mechanism (SPKM) [RFC2025].

3.3.2 Attribute Classes

class	value	type
GSS Identity Name	16384	B/V

GSS Identity Name Attribute (IKE)

When using the GSS-API authentication method, the GSS Identity Name attribute may be used to pass the GSS-API endpoint names for the initiator and responder. The format for these name strings are private to the underlying GSS-API mechanism.

3.3.3 GSS-API Token Payload (ISAKMP)

When using the GSS-API authentication method, the GSS Token Payload is used to pass the content of the GSSi[2] and GSSr[2] tokens. The Next Payload value for the GSS-API Token Payload is 129.

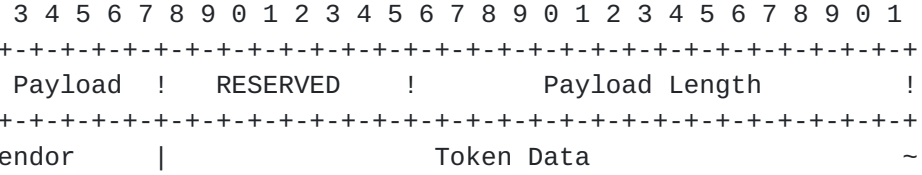

Figure 1: GSS-API Token Payload (ISAKMP)

- Next Payload (1 octet) - Identifier for the payload type of the next payload in the message. If the current payload is the last in the message, this field will be zero (0).
- RESERVED (1 octet) - Unused, must be zero (0).
- Payload Length (2 octets) - Length in octets of the current payload, including the generic payload header.
- Vendor Encoding (1 octet) - Vendor-specific encoding or versioning prefix, may be non-zero.
- Token Data (variable length) - GSS-API token data (private to the local GSS-API).

3.4 The GSS-API Authentication Method ISAKMP Vendor ID Signature

This memo describes a protocol that lives on top of RFC-2408 and as a companion to RFC-2409. These standards-track protocols reserve some of their "magic number" space for private use by mutually consenting parties. It is from this number space that this memo obtains some of the "magic numbers" it needs (payload types, authentication method, attributes). As part of the "mutually consenting parties" part of the requirement implementors of this protocol are encouraged to use a Vendor ID payload to announce willingness to engage in this protocol. The contents of the Vendor ID payload will be the following hexadecimal string: 0xb46d8914f3aaa3f2fedeb7c7db2943ca, which is the result of an MD5 hash of "A GSS-API Authentication Method for IKE". An RFC-2409 implementation which implements this protocol and receives a Vendor ID payload with this string in the body of the payload can assume that the sender of the Vendor ID payload has likewise implemented this protocol and is therefore a "mutually consenting party".

For Windows 2000 compatibility, the contents of the Vendor ID payload is instead the result of an MD5 hash over the ASCII string "GSSAPI". Microsoft's IKE implementation does not send this payload, but MUST receive either this Vendor ID or the generic Microsoft Vendor ID (an MD5 hash over the ASCII string "MS NT5 ISAKMPOAKLEY").

If this protocol is advanced to standards-track status IANA will assign new "magic numbers" out of the appropriate number spaces (the "magic numbers" will no longer be from the private use ranges) and the requirement to use a Vendor ID payload will cease.

4. Change Log

4.1 Changes from -06/07

- No changes. Draft resubmitted to make ID current again.

4.2 Changes from -05

- Specify unicode encoding for opaque endpoint ID's in Section 3 for Windows 2000 compatibility.
- Make endpoint ID exchange in Section 3.2 mandatory for Windows 2000 compatibility.
- Added extra reserved byte proceeding token payload format for Windows 2000 compatibility.
- Added Vendor ID usage guidelines for Windows 2000 compatibility.

4.3 Changes from -04

- Cleanup Section 3.2 description of GSS_S_CONTINUE_NEEDED handling with Aggressive Mode.

4.4 Changes from -03

- Restore private use numbers to V2 values (Microsoft Windows 2000).

4.5 Changes from -02

- Generalize exchange for "n" round-trips.
- Remove GSSIi and GSSIr nomenclature; use GIi and GIr explicitly.
- Move magic numbers into mutual consent range; add Section 3.4.
- Add second paragraph to Security Considerations.
- Update document references.
- Update preamble language (RFC-2026).

4.6 Changes from -01

- Add optional GSSi2 and GSSr2 token definitions to Section 3.1.
- Add optional GSSi2 and GSSr2 tokens to Main Mode diagram.
- Add GSS Token Payload Figure to Section 3.3.
- Update document references to reflect IPSEC RFC status (!).
- Update most references to ISAKMP/Oakley to IKE.

4.7 Changes from -00

- GSSIi and GSSIr are required; remove optional brackets.
- Add text for GSS_Wrap/GSS_Unwrap over HASH_I and HASH_R.

5. Security Considerations

This entire draft pertains to a negotiated key management protocol, combining Oakley (RFC-2412) with ISAKMP (RFC-2408), which negotiates and derives keying material for security associations in a secure and authenticated manner. Specific discussion of the various security protocols and transforms identified in this document can be found in the associated base documents, in the cipher references, and throughout this document.

This draft defines an authentication method that is based on GSS-API. The strength of the authentication is therefore completely dependent on the underlying GSS-API mechanism definition. This document defines a protocol which provides mutual authentication between the GSS-API peers and binds the IKE exchange to the GSS-API shared secrets. It does not provide any additional authentication beyond that provided by the GSS-API mechanism.

Acknowledgments

Thanks to Dan Harkins for reviewing the early drafts and for allowing me to liberate the notation from RFC-2409. Special thanks to Bill Sommerfeld, Ran Canetti, Pau-Chen Cheng, and Hugo Krawczyk for pointing out serious problems in the first version of this document.

References

[Linn98] Linn, J., "Generic Security Service Application Program Interface, Version 2, Update 1," draft-ietf-cat-rfc2078bis-08.txt (supersedes RFC-2078). Work in progress.

[Wray98] Wray, J., "Generic Security Service API Version 2 : C-bindings," draft-ietf-cat-gssv2-cbind-09.txt (supersedes RFC-1509). Work in progress.

Author's Address:

Derrell Piper <ddp@cips.nokia.com>
Nokia Corporation
1538 Pacific Ave
Santa Cruz, CA 95060
United States of America
+1 831 460-3800 x3822

Brian Swander <briansw@microsoft.com>
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

United States of America
+1 425 703-8182