1. Abstract

This document describes an alternate authentication method for ISAKMP/Oakley which makes use of GSS-API to authenticate the Diffie-Hellman exchange. The mechanism described here extends the authentication modes defined in [Harkins97] without introducing any modifications to the ISAKMP/Oakley key exchange protocol. This constraint however, necessarily restricts the number of GSS-API exchanges and may limit the broader applicability of this method. Additional work is needed to provide a fully generalized solution. Such a solution will require ISAKMP/Oakley protocol modifications.

For a list of changes since the previous version of the IPSEC DOI, please see Section 5.
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

[Harkins97] 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 mode. 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 the initiator and responder GSS-API tokens generated by the local GSS-API’s using GSS_Init_sec_context and GSS_Accept_sec_context respectively.

GSSIIi and GSSIr are the concatenation of optional GSS-API identity strings and either GSSI or GSSr for the initiator and responder respectively.

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 [Pip97].

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.

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 ([Harkins97]) defines a key negotiation protocol that blends the Oakley key determination protocol ([Orman97]) with ISAKMP ([Maughan97]) to provide authenticated cryptographic key exchange for use with IP security protocols (e.g. AH/ESP). The ISAKMP/Oakley 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 ([Linn97]) 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., [Wray97]). 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 SNEGO ([Pinkas97])). 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 ([ISAKMP], 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 ISAKMP/Oakley 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 4.3). Note that these name strings are bound to the exchange but otherwise unauthenticated. The GSS-API endpoint names are also assumed to be opaque.

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 4.1). 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:

<table>
<thead>
<tr>
<th>Flag</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>mutual_req_flag</td>
<td>MUST</td>
</tr>
<tr>
<td>integ_req_flag</td>
<td>MUST</td>
</tr>
<tr>
<td>conf_req_flag</td>
<td>SHOULD</td>
</tr>
</tbody>
</table>

If the GSS-API authentication cannot be completed in 1.5 round-trips, the method described in this document will not work. To fully generalize this extension, a new XCHG type will need to be created that allows for any number of GSS-API exchanges but is otherwise similar to the existing Main Mode exchange. A single Main Mode-like XCHG type is probably sufficient since there would be little use for an Aggressive Mode construct given the open ended nature of GSS-API.

The primary motivation for this work was to integrate Kerberos authentication into ISAKMP/Oakley in an environment where the host operating system did not expose the underlying Kerberos authentication services except as a GSS-API package. Since the details of the host operating system’s Kerberos package were known, the limitations described above were addressed in a reasonable manner by simply failing the ISAKMP/Oakley negotiation when the GSS-API’s
failed to converge in the requisite number of round-trips. When implemented this way, this event SHOULD be auditable and should clearly differentiate this type of authentication failure from one caused by truly bad credentials.

4.1 SKEYID Generation for GSS-API

[Harkins97] 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:

\[
\text{HASH}_I = \text{GSS}
\text{Wrap}(\text{prf(SKEYID, g^xi | g^xr | CKY-I | CKY-R | SAi_b | IDii_b | GSSIi))}
\]

\[
\text{HASH}_R = \text{GSS}
\text{Wrap}(\text{prf(SKEYID, g^xr | g^xi | CKY-R | CKY-I | SAi_b | IDir_b | GSSIr))}
\]

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.

4.2 ISAKMP/Oakley Phase 1 Authenticated With GSS-API

Using GSS-API, the ancillary information exchanged during the second roundtrip 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.
Aggressive mode using GSS-API is defined as

\[\text{Initiator} \quad \text{Responder}\]

\[
\begin{align*}
\text{HDR, SA, KE, Ni, Idii, GSSi} & \rightarrow \\
\text{HDR, SA, KE, Nr, GSSr} & \leftarrow \\
\text{HDR*, IDii, HASH_I} & \rightarrow \\
\text{HDR*, IDir, HASH_R} & \leftarrow \\
\text{HDR, HASH_I} & \rightarrow
\end{align*}
\]

4.3 GSS-API Identifiers and Attributes

Implementations using the GSS-API Authentication Mode will need to agree on the following values. These numbers are simply the beginning of the "private use" range for each particular list.

- Authentication Method
  
  Authentication with GSS-API 65001

Attribute Classes

<table>
<thead>
<tr>
<th>class</th>
<th>value</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSS Identity Name</td>
<td>32001</td>
<td>B/V</td>
</tr>
</tbody>
</table>

- GSS Identity Name

  When using the GSS-API authentication mode, 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 GSS-API.

5. Change Log

5.1 Changes from V0

- GSSIi and GSSIr are required; removed optional brackets
- added text for GSS_Wrap/GSS_Unwrap over HASH_I and HASH_R
6. Security Considerations

This entire draft pertains to a negotiated key management protocol, combining Oakley ([Orman97]) with ISAKMP ([Maughan97]), 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.

Acknowledgments

Thanks to Dan Harkins for reviewing the early drafts and for allowing me to liberate the notation from [Harkins97]. Special thanks to Bill Sommerfeld, Ran Canetti, Pau-Chen Cheng, and Hugo Krawczyk for pointing out problems in previous versions of this document.

References


Author’s Address:

Derrell Piper <ddp@network-alchemy.com>
Network Alchemy
1521.5 Pacific Ave
Santa Cruz, California, 95060
United States of America
+1 408 460-3822