INTERNET DRAFT R. Housley
Internet Engineering Task Force (IETF) Vigil Security
Intended Status: Standards Track T. Polk
NIST

Expires: 14 May 2010 14 November 2010

Database of Long-Lived Symmetric Cryptographic Keys
<draft-ietf-karp-crypto-key-table-00.txt>

Status of this Memo

Copyright Notice

Abstract

This document specifies the information contained in a database of long-lived cryptographic keys used by many different security

protocols. The database design supports both manual and automated key management. In many instances, the security protocols do not directly use the long-lived key, but rather a key derivation function is used to derive a short-lived key from a long-lived key.

1. Introduction

This document specifies the information that needs to be included in a database of long-lived cryptographic keys. This conceptual database is designed to support both manual key management and automated key management. The intent is to allow many different implementation approaches to the specified cryptographic key database.

Security protocols such as TCP-AO [RFC5925] are expected to use an application program interface (API) to select a long-lived key from the database. In many instances, the long-lived keys are not used directly in security protocols, but rather a key derivation function is used to derive short-lived key from the long-lived keys in the database. In other instances, security protocols will directly use the long-lived key from the database. The database design supports both use cases.

2. Conceptual Database Structure

The database is characterized as a table, where each row represents a single long-lived symmetric cryptographic key. Each key should only have one row; however, in the (hopefully) very rare cases where the same key is used for more than one purpose, multiple rows will contain the same key value. The columns in the table represent the key value and attributes of the key.

To accommodate manual key management, then formatting of the fields has been purposefully chosen to allow updates with a plain text editor.

The table has the following columns:

LocalKeyID

LocalKeyID is a 16-bit integer in hexadecimal. The LocalKeyID can be used by a peer to identify this entry in the database. For pairwise keys, the most significant bit in LocalKeyID is set to zero, and the integer value must be unique among all the pairwise keys in the database. For group keys, the most significant bit in LocalKeyID is set to one, but collisions among group key identifiers must be accommodated.

PeerKeyID

For pairwise keys, the PeerKeyID field is a 16 bit integer in hexadecimal provided by the peer. If the peer has not yet provided this value, the PeerKeyID is set to "unknown". For group keying, the PeerKeyID field is set to "group", which easily accommodates group keys generated by a third party.

Peers

The Peers field identifies the peer system or set of systems that have this key configured in their own database of long-lived keys. For pairwise keys, the database on the peer system LocalKeyID field will contain the value specified in the PeerKeyID field in the local database. For group keying, the Peers field names the group, not the individual systems that comprise the group.

Interfaces

The Interfaces field identifies the set of interfaces for which it is appropriate to use this key. When the long-lived value in the Key field is intended for use on any interface, the Interfaces field is set to "all".

Protocol

The Protocol field identifies a single security protocol where this key may be used to provide cryptographic protection.

KDF

The KDF field indicates which key derivation function is used to generate short-lived keys from the long-lived value in the Key field. When the long-lived value in the Key field is intended for direct use, the KDF field is set to "none".

KDFInputs

The KDFInputs field is used when supplementary public or private data is supplied to the KDF. For protocols that do not require additional information for the KDF, the KDFInputs field is set to "none". The Protocol field will determine the format of this field if it is not "none".

AlgID

The AlgID field indicates which cryptographic algorithm to be used with the security protocol for the specified peer. The algorithm may be an encryption algorithm and mode (such as AES-128-CBC), an authentication algorithm (such as HMAC-SHA1-96 or AES-128-CMAC), or any other symmetric cryptographic algorithm needed by a security protocol. If the KDF field contains "none", then the long-lived key is used directly with this algorithm, otherwise the derived short-lived key is used with this algorithm. When the long-lived key is used to

generate a set of short-lived keys for use with the security protocol, the AlgID field identifies a ciphersuite rather than a single cryptographic algorithm.

Key

The Key is a hexadecimal string representing a long-lived symmetric cryptographic key. The size of the Key depends on the KDF and the AlgID. For example, a KDF=none and AlgID=AES128 requires a 128-bit key, which is represented by 32 hexadecimal digits.

Direction

The Direction field indicates whether this key may be used for inbound traffic, outbound traffic, or both. The supported values are "in", "out", and "both", respectively. The Protocol field will determine which of these values are valid.

NotBefore

The NotBefore field specifies the earliest date and time in Universal Coordinated Time (UTC) at which this key should be considered for use. The format is YYYYMMDDHHSSZ, where four digits specify the year, two digits specify the month, two digits specify the day, two digits specify the hour, and two digits specify the minute. The "Z" is included as a clear indication that the time is in UTC.

NotAfter

The NotAfter field specifies the latest date and time at which this key should be considered for use. The format is the same as the NotBefore field.

Note that some security protocols use a KeyID value of zero for special purposes, so care is needed if this KeyID value is included in the table.

3. Key Selection and Rollover

When a system desires to protect a unicast protocol data unit for a remote system H using security protocol P via interface I, the local system selects a long-lived key at time T from the database, any key that satisfies the following conditions may be used:

(1) the Peer field includes H;

(2) the PeerKeyID field is not "unknown";

(3) the Protocol field matches P;

(4) the Interfaces field includes I;

(5) the Direction field is either "out" or "both"; and

(6) NotBefore <= T <= NotAfter.

The value in the PeerKeyID field is used to identify the selected key to the remote system H.

Group key selection is different than pairwise key selection. When a system desires to protect a multicast protocol data unit for a group of systems G using security protocol P via interface I, the local system selects a long-lived key at time T from the database, any key that satisfies the following conditions may be used:

(1) the Peer field includes the multicast group G;

(2) the PeerKeyID field is "group";

(3) the Protocol field matches P;

(4) the Interfaces field includes I;

(5) the Direction field is either "out" or "both"; and

(6) NotBefore <= T <= NotAfter.

The value in the LocalKeyID field is used to identify the selected key since all of the systems in the group G use the same identifier.

During algorithm transition, multiple entries may exist associated with different cryptographic algorithms or ciphersuites. Systems should support selection of keys based on algorithm preference.

In addition, multiple entries with overlapping use periods are expected to be employed to provide orderly key rollover. In these cases, the expectation is that systems will transition to the newest key available. To meet this requirement, this specification recommends supplementing the key selection algorithm with the following differentiation: select the long-lived key specifying the most recent time in the NotBefore field.

When a system participates in a security protocol, a sending peer system H has selected a long-lived key and the LocalKeyID is included in the protocol control information. When retrieving the long-lived key (for direct use or for key derivation), the local system should confirm the following conditions are satisfied before use:

(1) the Peer field includes H;

(2) the Protocol field matches P;

(3) the Interface field includes I;

(4) the Direction field is either "in" or "both"; and

(5) NotBefore <= T <= NotAfter.

Note that the key usage is loosely bound by the times specified in the NotBefore and NotAfter fields. New security associations should not be established except within the period of use specified by these fields, while allowing some grace time for clock skew. However, if a security association has already been established based on a particular long-lived key, exceeding the lifetime does not have any direct impact. Implementations of protocols that involve long-lived security association should be designed to periodically interrogate the database and rollover to new keys without tearing down the security association.

For group keying, the local system should confirm the following conditions are satisfied before use:

(1) the Peer field includes the multicast group G;

(2) the PeerKeyID field is "group";

(3) the Protocol field matches P;

(4) the Interface field includes I;

(5) the Direction field is either "in" or "both"; and

(6) NotBefore <= T <= NotAfter.

As long as a key remains in the database, the key may be used for received traffic. Any key that is unacceptable for received traffic needs to be removed from the database.

4. Operational Considerations

If usage periods for long-lived keys do not overlap and system clocks are inconsistent, it is possible to construct scenarios where systems cannot agree upon a long-lived key. When installing a series of keys to be used one after the other (sometimes called a key chain), operators should configure the NotAfter field of the preceding key to be several days after the NotBefore field of the subsequent key to

ensure that clock skew is not a concern.

For group keys, the most significant bit in LocalKeyID must be set to one. Collisions among group key identifiers can be avoided by subdividing the remaining 15 bits of the LocalKeyID field into an identifier of the group key generator and an identifier assigned by that generator.

5. Security Considerations

Management of encryption and authentication keys has been a significant operational problem, both in terms of key synchronization and key selection. For example, current guidance [RFC3562] warns against sharing TCP MD5 keying material between systems, and recommends changing keys according to a schedule. The same general operational issues are relevant for the management of other cryptographic keys.

It is recognized in [RFC4107] that automated key management is not viable in some situations. The conceptual database specified in this document is intended to accommodate both manual key management and automated key management. A future specification to automatically populate rows in the database is envisioned.

Designers should recognize the warning provided in [RFC4107]:

> Automated key management and manual key management provide very different features. In particular, the protocol associated with an automated key management technique will confirm the liveness of the peer, protect against replay, authenticate the source of the short-term session key, associate protocol state information with the short-term session key, and ensure that a fresh short-term session key is generated. Further, an automated key management protocol can improve interoperability by including negotiation mechanisms for cryptographic algorithms. These valuable features are impossible or extremely cumbersome to accomplish with manual key management.

6. IANA Considerations

No IANA actions are required.

{{{ RFC Editor: Please remove this section prior to publication. }}}

7. Acknowledgments

This document reflects many discussions with many different people over many years. In particular, the authors thank Jari Arkko, Ran

Atkinson, Ron Bonica, Ross Callon, Lars Eggert, Pasi Eronen, Adrian Farrel, Sam Hartman, Gregory Lebovitz, Sandy Murphy, Eric Rescorla, Mike Shand, Dave Ward, and Brian Weis for their insights.

8. Informational References

[RFC3562] Leech, M., "Key Management Considerations for the TCP MD5 Signature Option", RFC 3562, July 2003.

[RFC4107] Bellovin, S. and R. Housley, "Guidelines for Cryptographic Key Management", RFC 4107, BCP 107, June 2005.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, June 2010.

Authors' Addresses

Russell Housley
Vigil Security, LLC
918 Spring Knoll Drive
Herndon, VA 20170
USA
EMail: housley@vigilsec.com

Tim Polk
National Institute of Standards and Technology
100 Bureau Drive, Mail Stop 8930
Gaithersburg, MD 20899-8930
USA
EMail: tim.polk@nist.gov