Internet Engineering Task Force (IETF)
Request for Comments: 5858
Category: Standards Track
ISSN: 2070-1721
E. Ertekin
C. Christou
Booz Allen Hamilton
C. Bormann
Universitaet Bremen TZI
May 2010

IPsec Extensions to Support Robust Header Compression over IPsec

Abstract

Integrating Robust Header Compression (ROHC) with IPsec (ROHCoIPsec) offers the combined benefits of IP security services and efficient bandwidth utilization. However, in order to integrate ROHC with IPsec, extensions to the Security Policy Database (SPD) and Security Association Database (SAD) are required. This document describes the IPsec extensions required to support ROHCoIPsec.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc5858.

Table of Contents

1. Introduction

Using IPsec ([IPSEC]) protection offers various security services for IP traffic. However, these benefits come at the cost of additional packet headers, which increase packet overhead. By compressing the inner headers of these packets, the integration of Robust Header Compression (ROHC, [ROHC]) with IPsec (ROHCoIPsec, [ROHCOIPSEC]) can reduce the packet overhead associated with IPsec-protected flows.

IPsec-protected traffic is carried over Security Associations (SAs), whose parameters are negotiated on a case-by-case basis. The Security Policy Database (SPD) specifies the services that are to be offered to IP datagrams, and the parameters associated with SAs that have been established are stored in the Security Association Database (SAD). For ROHCoIPsec, various extensions to the SPD and SAD that incorporate ROHC-relevant parameters are required.

In addition, three extensions to IPsec processing are required. First, a mechanism for identifying ROHC packets must be defined. Second, a mechanism to ensure the integrity of the decompressed packet is needed. Finally, the order of the inbound and outbound processing must be enumerated when nesting IP Compression (IPComp [IPCOMP]), ROHC, and IPsec processing.

2. Terminology

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 RFC 2119 [BRA97].

3. Extensions to IPsec Databases

The following subsections specify extensions to the SPD and the SAD that MUST be supported for ROHCoIPsec. The ROHCoIPsec fields in the SPD are used to populate the ROHCoIPsec parameters in the SAD during the initialization or rekey of a child SA.

It is noted that these extensions do not have any implications on existing SPD fields or SAD parameters. Therefore, a ROHCoIPsec implementation is backwards-compatible with an IPsec implementation that does not support header compression.

Appendix A provides an example ASN.1 representation of an SPD that is extended to support ROHC.

3.1. Security Policy Database (SPD)

In general, the SPD is responsible for specifying the security services that are offered to IP datagrams. Entries in the SPD specify how to derive the corresponding values for SAD entries. To support ROHC, the SPD is extended to include per-channel ROHC parameters. Together, the existing IPsec SPD parameters and the ROHC parameters will dictate the security and header compression services that are provided to packets.

The fields contained within each SPD entry are defined in RFC 4301 [IPSEC], Section 4.4.1.2. To support ROHC, several processing info fields are added to the SPD; these fields contain information regarding the ROHC profiles and channel parameters supported by the local ROHC instance.

If the processing action associated with the selector sets is PROTECT, then the processing info must be extended with the following ROHC channel parameters:

MAX_CID: This field indicates the highest context ID that will be decompressed by the local decompressor. MAX_CID MUST be at least 0 and at most 16383 (the value 0 implies having one context).

MRRU: The MRRU parameter indicates the size of the largest reconstructed unit (in octets) that the local decompressor is expected to reassemble from ROHC segments. This size includes the Cyclic Redundancy Check (CRC) and the ROHC Integrity Check Value (ICV). NOTE: Since in-order delivery of ROHC packets cannot be guaranteed, the MRRU parameter SHOULD be set to 0 (as stated in Section 5.2.5.1 of RFC 5795 [ROHC] and Section 6.1 of RFC 5225 [ROHCV2]), which indicates that no segment headers are allowed on the ROHCoIPsec channel.

PROFILES: This field is a list of ROHC profiles supported by the local decompressor. Possible values for this list are contained in the "RObust Header Compression (ROHC) Profile Identifiers" registry [ROHCPROF].

In addition to these ROHC channel parameters, a ROHC integrity algorithm and a ROHC ICV Length field MUST be included within the SPD:

ROHC INTEGRITY ALGORITHM: This field is a list of integrity algorithms supported by the ROHCoIPsec instance. This will be used by the ROHC process to ensure that packet headers are properly decompressed (see Section 4.2). Authentication algorithms that MUST be supported are specified in the

"Authentication Algorithms" table in Section 3.1.1 ("ESP Encryption and Authentication Algorithms") of RFC 4835 [CRYPTO-ALG] (or its successor).

ROHC ICV LENGTH: This field specifies the length of the ICV that is used in conjunction with the ROHC integrity algorithm.

Several other ROHC channel parameters are omitted from the SPD, because they are set implicitly. The omitted channel parameters are LARGE_CIDS and FEEDBACK_FOR. The LARGE_CIDS channel parameter MUST be set based on the value of MAX_CID (i.e., if MAX_CID is <= 15, LARGE_CIDS is assumed to be 0). Finally, the ROHC FEEDBACK_FOR channel parameter MUST be set to the ROHC channel associated with the SA in the reverse direction. If an SA in the reverse direction does not exist, the FEEDBACK_FOR channel parameter is not set, and ROHC MUST NOT operate in bi-directional Mode.

3.2. Security Association Database (SAD)

Each entry within the SAD defines the parameters associated with each established SA. Unless the "populate from packet" (PFP) flag is asserted for a particular field, SAD entries are determined by the corresponding SPD entries during the creation of the SA.

The data items contained within the SAD are defined in RFC 4301 [IPSEC], Section 4.4.2.1. To support ROHC, the SAD must include a "ROHC Data Item"; this data item contains parameters used by ROHC instance. The ROHC Data Item exists for both inbound and outbound SAs.

The ROHC Data Item includes the ROHC channel parameters for the SA. These channel parameters (i.e., MAX_CID, PROFILES, MRRU) are enumerated above in Section 3.1. For inbound SAs, the ROHC Data Item MUST specify the ROHC channel parameters that are used by the local decompressor instance; conversely, for outbound SAs, the ROHC Data Item MUST specify the ROHC channel parameters that are used by local compressor instance.

In addition to these ROHC channel parameters, the ROHC Data Item for both inbound and outbound SAs MUST include three additional parameters. Specifically, these parameters store the integrity algorithm, the algorithm's respective key, and the ICV length that is used by the ROHC process (see Section 3.2). The integrity algorithm and its associated key are used to calculate a ROHC ICV of the specified length; this ICV is used to verify the packet headers post-decompression.

Finally, for inbound SAs, the ROHC Data Item MUST include a FEEDBACK_FOR parameter. The parameter is a reference to a ROHC channel in the opposite direction (i.e., the outbound SA) between the same compression endpoints. A ROHC channel associated with an inbound SA and a ROHC channel associated with an outbound SA MAY be coupled to form a bi-directional ROHC channel as defined in Sections 6.1 and 6.2 in RFC 3759 [ROHC-TERM].

"ROHC Data Item" values MAY be initialized manually (i.e., administratively configured for manual SAs), or initialized via a key exchange protocol (e.g., IKEv2 [IKEV2]) that has been extended to support the signaling of ROHC parameters [IKE-ROHC].

4. Extensions to IPsec Processing

4.1. Identification of Header-Compressed Traffic

A "ROHC" protocol identifier is used to identify header-compressed traffic on a ROHC-enabled SA. If an outbound packet has a compressed header, the Next Header field of the security protocol header (e.g., Authentication Header (AH) [AH], Encapsulating Security Payload (ESP) [ESP]) MUST be set to the "ROHC" protocol identifier. If the packet header has not been compressed by ROHC, the Next Header field does not contain the "ROHC" protocol identifier. Conversely, for an inbound packet, the value of the security protocol Next Header field MUST be checked to determine if the packet includes a ROHC header, in order to determine if it requires ROHC decompression.

Use of the "ROHC" protocol identifier for purposes other than ROHCoIPsec is currently not defined. Future protocols that make use of the allocation (e.g., other applications of ROHC in multi-hop environments) require specification of the logical compression channel between the ROHC compressor and decompressor. In addition, these specifications will require the investigation of the security considerations associated with use of the "ROHC" protocol identifier outside the context of the Next Header field of security protocol headers.

4.2. Verifying the Integrity of Decompressed Packet Headers

As documented in Section 6.1.4 of [ROHCOIPSEC], ROHC is inherently a lossy compression algorithm: the consequences of significant packet reordering or loss between ROHC peers may include undetected decompression failures, where erroneous packets are forwarded into the protected domain.

To ensure that a decompressed header is identical to the original header, ROHCoIPsec MAY use an additional integrity algorithm (and respective key) to compute a second Integrity Check Value (ICV). This ROHC ICV MUST be computed over the uncompressed IP header, as well at the higher-layer headers and the packet payload. When computed, the ICV is appended to the ROHC-compressed packet. At the decompressor, the decompressed packet (including the uncompressed IP header, higher-layer headers, and packet payload; but not including the authentication data) will be used with the integrity algorithm (and its respective key) to compute a value that will be compared to the appended ICV. If these values are not identical, the decompressed packet MUST be dropped.

Figure 1 illustrates the composition of a ROHCoIPsec-processed IPv4 packet. In the example, TCP/IP compression is applied, and the packet is processed with tunnel mode ESP.

```
                 BEFORE COMPRESSION AND APPLICATION OF ESP
                 ----------------------------
           IPv4  |orig IP hdr  |     |      |
                 |(any options)| TCP | Data |
                 ----------------------------

                  AFTER ROHCOIPSEC COMPRESSION AND APPLICATION OF ESP
                ------------------------------------------------------
          IPv4  | new IP hdr  |     | Cmpr. |    | ROHC | ESP    | ESP|
                |(any options)| ESP | Hdr.  |Data| ICV  |Trailer| ICV|
                ------------------------------------------------------
```

Figure 1. Example of a ROHCoIPsec-Processed Packet

Note: At the decompressor, the ROHC ICV field is not included in the calculation of the ROHC ICV.

4.2.1. ICV Computation and Integrity Verification

In order to correctly verify the integrity of the decompressed packets, the processing steps for ROHCoIPsec MUST be implemented in a specific order, as given below.

For outbound packets that are processed by ROHC and are IPsec-protected:

o Compute an ICV for the uncompressed packet with the negotiated (ROHC) integrity algorithm and its respective key.

o Compress the packet headers (as specified by the ROHC process).

- o Append the ICV to the compressed packet.
- o Apply AH or ESP processing to the packet, as specified in the appropriate SAD entry.

For inbound packets that are to be decompressed by ROHC:

- o Apply AH or ESP processing, as specified in the appropriate SAD entry.
- o Remove the ICV from the packet.
- o Decompress the packet header(s).
- o Compute an ICV for the decompressed packet with the negotiated (ROHC) integrity algorithm and its respective key.
- o Compare the computed ICV to the original ICV calculated at the compressor: if these two values differ, the packet MUST be dropped; otherwise, resume IPsec processing.

4.3. ROHC Segmentation and IPsec Tunnel MTU

In certain scenarios, a ROHCoIPsec-processed packet may exceed the size of the IPsec tunnel MTU. RFC 4301 [IPSEC] currently stipulates the following for outbound traffic that exceeds the SA Path MTU (PMTU):

Case 1: Original (cleartext) packet is IPv4 and has the Don't Fragment (DF) bit set. The implementation should discard the packet and send a PMTU ICMP message.

Case 2: Original (cleartext) packet is IPv4 and has the DF bit clear. The implementation should fragment (before or after encryption per its configuration) and then forward the fragments. It should not send a PMTU ICMP message.

Case 3: Original (cleartext) packet is IPv6. The implementation should discard the packet and send a PMTU ICMP message.

For the ROHCoIPsec processing model, there is one minor change to the procedure stated above. This change applies to pre-encryption fragmentation for Case 2. Since current ROHC compression profiles do not support compression of IP packet fragments, pre-encryption fragmentation MUST NOT occur before ROHC processing.

If the compressed packet exceeds the SA PMTU, and the MRRU is non-zero, ROHC segmentation MAY be used to divide the packet, where each segment conforms to the tunnel MTU. ROHC segmentation MUST occur before AH or ESP processing. Because in-order delivery of ROHC segments is not guaranteed, the use of ROHC segmentation is not recommended.

If segmentation is applied, the process MUST account for the additional overhead imposed by the IPsec process (e.g., AH or ESP overhead, crypto synchronization data, the additional IP header, etc.) such that the final IPsec-processed segments are less than the tunnel MTU. After segmentation, each ROHC segment is consecutively processed by the appropriate security protocol (e.g., AH, ESP) instantiated on the ROHC-enabled SA. Since ROHC segments are processed consecutively, the associated AH/ESP sequence number MUST be incremented by one for each segment transmitted over the ROHC channel. As such, after all ROHC segments receive AH/ESP processing, these segments can be identified (at the remote IPsec implementation) by a range of contiguous AH/ESP sequence numbers.

For channels where the MRRU is non-zero, the ROHCoIPsec decompressor MUST re-assemble the ROHC segments that are received. To accomplish this, the decompressor MUST identify the ROHC segments (as documented in Section 5.2 of RFC 5795 [ROHC]), and attempt reconstruction using the ROHC segmentation protocol (Section 5.2.5 of RFC 5795 [ROHC]). To assist the reconstruction process, the AH/ESP sequence number SHOULD be used to identify segments that may have been subject to reordering. If reconstruction fails, the packet MUST be discarded.

As stated in Section 3.2.1, if the ROHC integrity algorithm is used to verify the decompression of packet headers, this ICV is appended to the compressed packet. If ROHC segmentation is performed, the segmentation algorithm is executed on the compressed packet and the appended ICV. Note that the ICV is not appended to each ROHC segment.

Under certain circumstances, IPsec implementations will not process (or receive) unprotected ICMP messages, or they will not have a Path MTU estimated value. In these cases, the IPsec implementation SHOULD NOT attempt to segment the ROHC-compressed packet, as it does not have full insight into the path MTU in the unprotected domain.

4.4. Nested IPComp and ROHCoIPsec Processing

IPComp ([IPCOMP]) is another mechanism that can be implemented to reduce the size of an IP datagram. If IPComp and ROHCoIPsec are implemented in a nested fashion, the following steps MUST be followed for outbound and inbound packets.

For outbound packets that are to be processed by IPComp and ROHC:

- o The ICV is computed for the uncompressed packet, and the appropriate ROHC compression profile is applied to the packet.
- o IPComp is applied, and the packet is sent to the IPsec process.
- o The security protocol is applied to the packet.

Conversely, for inbound packets that are to be both ROHC- and IPComp-decompressed:

- o A packet received on a ROHC-enabled SA is IPsec-processed.
- o The datagram is decompressed based on the appropriate IPComp algorithm.
- o The packet is sent to the ROHC module for header decompression and integrity verification.

5. Security Considerations

A ROHCoIPsec implementer should consider the strength of protection provided by the integrity check algorithm used to verify decompressed headers. Failure to implement a strong integrity check algorithm increases the probability for an invalidly decompressed packet to be forwarded by a ROHCoIPsec device into a protected domain.

The implementation of ROHCoIPsec may increase the susceptibility for traffic flow analysis, where an attacker can identify new traffic flows by monitoring the relative size of the encrypted packets (i.e., a group of "long" packets, followed by a long series of "short" packets may indicate a new flow for some ROHCoIPsec implementations). To mitigate this concern, ROHC padding mechanisms may be used to arbitrarily add padding to transmitted packets to randomize packet sizes. This technique, however, reduces the overall efficiency benefit offered by header compression.

6. IANA Considerations

IANA has allocated the value 142 to "ROHC" within the "Protocol Numbers" registry [PROTOCOL]. This value will be used to indicate that the next-level protocol header is a ROHC header.

7. Acknowledgments

The authors would like to thank Sean O'Keeffe, James Kohler, Linda Noone of the Department of Defense, and Rich Espy of OPnet for their contributions and support for developing this document.

The authors would also like to thank Yoav Nir and Robert A Stangarone Jr.: both served as committed document reviewers for this specification.

Finally, the authors would like to thank the following for their numerous reviews and comments to this document:

- Magnus Westerlund
- Stephen Kent
- Lars-Erik Jonsson
- Carl Knutsson
- Pasi Eronen
- Jonah Pezeshki
- Tero Kivinen
- Joseph Touch
- Rohan Jasani
- Elwyn Davies
- Bert Wijnen

Appendix A. ASN.1 Representation for ROHCoIPsec

This appendix is included as an additional way to describe the ROHCoIPsec parameters that are included in the IPsec SPD. It uses portions of the ASN.1 syntax provided in Appendix C of RFC 4301 [IPSEC]. In addition, several new structures are defined.

This syntax has been successfully compiled. However, it is merely illustrative and need not be employed in an implementation to achieve compliance.

The "Processing" data structure, defined in Appendix C of RFC 4301, is augmented to include a ROHC parameters element as follows:

```
      Processing ::= SEQUENCE {
          extSeqNum   BOOLEAN, -- TRUE 64 bit counter, FALSE 32 bit
          seqOverflow BOOLEAN, -- TRUE rekey, FALSE terminate & audit
          fragCheck   BOOLEAN, -- TRUE stateful fragment checking,
                               -- FALSE no stateful fragment checking
          lifetime    SALifetime,
          spi         ManualSPI,
          algorithms  ProcessingAlgs,
          tunnel      TunnelOptions OPTIONAL,
          rohc        [7] RohcParams OPTIONAL

      }
```

The following data structures describe these ROHC parameters:

```
    RohcParams ::= SEQUENCE {
        rohcEnabled         BOOLEAN, --  TRUE, hdr compr. is enabled
                                     -- FALSE, hdr compr. is disabled
        maxCID              INTEGER (0..16383),
        mrru                INTEGER,
        profiles            RohcProfiles,
        rohcIntegAlg        RohcIntegAlgs,
        rohcIntegICVLength  INTEGER
        }

    RohcProfiles ::= SET OF RohcProfile
```

```
   RohcProfile ::= INTEGER {
       rohcv1-rtp          (1),
       rohcv1-udp          (2),
       rohcv1-esp          (3),
       rohcv1-ip           (4),

       rohcv1-tcp          (6),
       rohcv1-rtp-udpLite  (7),
       rohcv1-udpLite      (8),

       rohcv2-rtp        (257),
       rohcv2-udp        (258),
       rohcv2-esp        (259),
       rohcv2-ip         (260),

       rohcv2-rtp-udpLite (263),
       rohcv2-udpLite    (264)

       -- values taken from [ROHCPROF]

       }

   RohcIntegAlgs ::= SEQUENCE {
       algorithm   RohcIntegAlgType,
       parameters  ANY -- DEFINED BY algorithm -- OPTIONAL }

   RohcIntegAlgType ::= INTEGER {
       none                    (0),
       auth-HMAC-MD5-96        (1),
       auth-HMAC-SHA1-96       (2),
       auth-DES-MAC            (3),
       auth-KPDK-MD5           (4),
       auth-AES-XCBC-96        (5),
       auth-HMAC-MD5-128       (6),
       auth-HMAC-SHA1-160      (7),
       auth-AES-CMAC-96        (8),
       auth-AES-128-GMAC       (9),
       auth-AES-192-GMAC      (10),
       auth-AES-256-GMAC      (11),
       auth-HMAC-SHA2-256-128 (12),
       auth-HMAC-SHA2-384-192 (13),
       auth-HMAC-SHA2-512-256 (14)
       --  tbd (15..65535)

       -- values taken from "Transform Type 3 - Integrity
       -- Algorithm Transform IDs" at [IKEV2-PARA]

       }
```

8. References

8.1. Normative References

[IPSEC] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[ROHC] Sandlund, K., Pelletier, G., and L-E. Jonsson, "The RObust Header Compression (ROHC) Framework", RFC 5795, March 2010.

[IPCOMP] Shacham, A., Monsour, B., Pereira, R., and M. Thomas, "IP Payload Compression Protocol (IPComp)", RFC 3173, September 2001.

[BRA97] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[ROHCV2] Pelletier, G. and K. Sandlund, "RObust Header Compression Version 2 (ROHCv2): Profiles for RTP, UDP, IP, ESP and UDP-Lite", RFC 5225, April 2008.

[IKEV2] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

[IKE-ROHC] Ertekin, E., Christou, C., Jasani, R., Kivinen, T., and C. Bormann, "IKEv2 Extensions to Support Robust Header Compression over IPsec", RFC 5857, May 2010.

[AH] Kent, S., "IP Authentication Header", RFC 4302, December 2005.

[ESP] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

8.2. Informative References

[ROHCOIPSEC] Ertekin, E., Jasani, R., Christou, C., and C. Bormann, "Integration of Header Compression over IPsec Security Associations", RFC 5856, May 2010.

[ROHCPROF] IANA, "RObust Header Compression (ROHC) Profile Identifiers", <http://www.iana.org>.

[CRYPTO-ALG] Manral, V., "Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 4835, April 2007.

[ROHC-TERM] Jonsson, L-E., "Robust Header Compression (ROHC): Terminology and Channel Mapping Examples", RFC 3759, April 2004.

[PROTOCOL] IANA, "Assigned Internet Protocol Numbers", <http://www.iana.org>.

[IKEV2-PARA] IANA, "Internet Key Exchange Version 2 (IKEv2) Parameters", <http://www.iana.org>.

Authors' Addresses

Emre Ertekin
Booz Allen Hamilton
5220 Pacific Concourse Drive, Suite 200
Los Angeles, CA 90045
US

EMail: ertekin_emre@bah.com

Chris Christou
Booz Allen Hamilton
13200 Woodland Park Dr.
Herndon, VA 20171
US

EMail: christou_chris@bah.com

Carsten Bormann
Universitaet Bremen TZI
Postfach 330440
Bremen D-28334
Germany

EMail: cabo@tzi.org