6lo
Internet-Draft
Intended status: Standards Track
Expires: August 21, 2015

D. Migault, Ed.
Ericsson
T. Guggemos, Ed.
LMU Munich

February 17, 2015

Diet-IPsec: ESP Payload Compression of IPv6 / UDP / TCP / UDP-Lite
draft-mglt-6lo-diet-esp-payload-compression-01.txt

Abstract

ESP is a IPsec protocol that takes as input a Clear Text Data and outputs an encrypted ESP packet according to IPsec rules and parameters stored in different IPsec databases.

Diet-ESP compresses the ESP fields. However, Diet-ESP does not consider compression of the Clear Text Data. Instead, if compression of the Clear Text Data is expected protocols like ROHCoverIPsec can be used.

ROHCoverIPsec remains complex to implement in IoT devices, as states, and negotiations are involved between the compressors and decompressors of the two IoT devices. Most of this complexity can be avoided by considering the parameters that have been negotiated by IPsec.

This document describes an extension of the Diet-ESP Context that enables the compression of the Clear Text Data, without implementing the complex ROHCoverIPsec framework. As opposed to ROHCoverIPsec the compression is not generic and as such all communication will not benefit from this compression. However, we believe this extension addresses most of IoT communications.

This Internet-Draft will expire on August 21, 2015.

Table of Contents

1. Requirements notation

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 [RFC2119].

2. Introduction

Diet-ESP [I-D.mglt-6lo-diet-esp] describes how to compress ESP fields. Fields are compressed according to a Diet-ESP Context. Diet-ESP has been described as a specific ROHC [RFC5795] framework that has no IR, IR-DYN nor any feed back ROHC message. It works in the Unidirectional mode of operation (U mode), and all necessary parameters are transmitted via the Diet-ESP Context that is negotiated between the two peers. As a result Diet-ESP defines a very specific and simplified ROHC framework which makes possible to implement Diet-ESP without implementing the whole ROHC.

In fact, Diet-ESP avoids ROHC complexity as a lot of parameters have already been negotiated with IKEv2 [RFC7296].

This document describes the Diet-ESP Payload Compression Extension. It does not consider the compression of the ESP fields. Instead, it goes one step further and describes how to compress the Clear Text Data or ESP Payload before it is encrypted by Diet-ESP. The Clear Text Data is generally constituted by an IP packet with IP -- if IPsec tunnel mode is used --, transport and application layers. Similarly to Diet-ESP, compression takes advantage of the IPsec parameters -- like IP addresses, transport layer parameters -- that have been negotiated in order to establish the Security Association -- via IKEv2 for example. In addition, similarly to Diet-ESP, the compression is described using the ROHC terminology, but uses a very specific and simplified ROHC framework of Diet-ESP. This makes possible compression of the Clear Text Data without implementing a whole ROHC framework for ROHCoverIPsec [RFC5856].

[I-D.mglt-6lo-diet-esp] clarifies the interactions of Diet-ESP with ROHC and 6LoWPAN. The Diet-ESP extension explained in this document replaces ROHCoverIPsec and 6LoWPANoverIPsec, protocols which offers similar functionality without using the IPsec databases. The Diet-ESP Payload Compression Extension uses the IPsec databases to avoid complex dialogues between compressors and decompressors.

The Diet-ESP Payload Compression Extension can be described as follows:

- 1. Definition or Diet-ESP parameters: COMPRESS_ESP_PAYLOAD, TRANSPORT_CHECKSUM_LSB and TRANSPORT_SEQUENCE_NUMBER_LSB. COMPRESS_ESP_PAYLOAD indicates the peers expect the Clear Text Data to be compressed, TRANSPORT_CHECKSUM_LSB and TRANSPORT_SEQUENCE_NUMBER_LSB are additional parameters to perform the compression.

- 2. Definition of a Diet-ESP Payload Compression algorithm.

The remaining of the document is as follows. Section 4 describes the new parameters for the Diet-ESP Context. Section 5 describes the protocol. Section 6, Section 7, Section 7 and Section 9 describe the compression of the IP layer and the transport layer (UDP, UDP-Lite.

3. Terminology

Diet-ESP Context: Like defined in Diet-ESP document.

SPD: Security Policy Database

SAD: Security Association Database

TS: Traffic Selector of a Security Association.

LSB: Least Significant Byte

MSB: Most Significant Byte

4. Diet-ESP Context Extension

This section describes the additional parameters of the Diet-ESP Context to implement the ESP Payload Compression extension.

Context Field Name	Overview
COMPRESS_ESP_PAYLOAD	Defines the use of the Traffic Selector for (de-)compression.
TRANSPORT_CHECKSUM_LSB	LSB of the UDP, UDP-Lite or TCP checksum
TRANSPORT_SEQUENCE_NUMBER_LSB	LSB of the TCP Sequence Number.

Table 1: Diet-ESP Context.

COMPRESS_ESP_PAYLOAD:
Defines if the ESP Payload MUST be compressed or not. Note that as detailed later, compression of the ESP Payload requires that IP addresses, or protocols are unique in the Security Association Databases. If not the compression does not compress does not output a compressed ESP Payload.

TRANSPORT_CHECKSUM_LSB:
If an inner header provides a checksum this can be compressed by the LSB mechanism. How the checksum is compressed is specified by the related profiles, e.g. UDP Section 7 , UDP-Lite Section 8 and TCP Section 9.

TRANSPORT_SEQUENCE_NUMBER_LSB:
If an inner header provides a Sequence Number, one MAY choose to use the SN stored in the SA for compression. Therefore the context provides the LSB of the Sequence Number which is used by all profiles, defining the Sequence Number as compressed with LSB, e.g. TCP Section 9.

5. Protocol Overview

The Diet-ESP Payload Compression is described by the pseudo code in Figure 1. The Clear Text Data is compressed only if COMPRESS_ESP_PAYLOAD is set. Otherwise, it is left unchanged. When COMPRESS_ESP_PAYLOAD is set, compression is performed on the IP and transport layer if and only if two conditions are met. First the layer must exist. This means for example that the IP layer is compressed only for the tunnel mode. Then, the layer can be compressed if and only if the values are uniquely derived from the IPsec databases. More specifically, if a SPD match occurs with at least two different values, then the compression do not occurs.

As a result, the IP layer can be compressed only if the IP address appears as a Traffic Selector. If the Traffic Selector is defined as a subnetwork, a SPD match occurs with more then one IP address, and then no compression occurs. Similarly, the transport layer is compressed if and only if it appears as a Traffic Selector. If a SPD match occurs with different transport protocol then the compression of the transport layer does not occurs.

The Diet-ESP Payload Compression is straight forward, but may at some point not fits all the needs. At some point using alternative compression as those proposed by ROHCoverIPsec may be preferred. In these cases, Diet-ESP Payload Compression MUST NOT be performed and COMPRESS_ESP_PAYLOAD MUST be unset.

```
<CODE BEGINS>
if COMPRESS_ESP_PAYLOAD is set :
         proceed to Diet-ESP Payload Compression
else:
        clear_text_data is left unchanged.

def diet_esp_payload_compression(clear_text_data, \
                                 TRANSPORT_CHECKSUM_LSB,\
                                 TRANSPORT_SEQUENCE_NUMBER_LSB):
    ## Compress IP header if the ipsec mode is tunnel and
    ## the inner IP addresses can uniquely be derived from IPsec DB.
    ## In other words, subnets are not considered.
    if clear_text_data has IP layer and \
       IP addresses is a unique Traffic Selector and \
       ipsec mode is tunnel:
        compress the IP layer
    if clear_text_data has transport layer and \
       transport layer is a Traffic Selector:
        compress transport layer
<CODE ENDS>
```

Figure 1: Diet-ESP Payload Compression Pseudo Code

Roughly speaking Diet-ESP is able to remove all header fields which have unique values inside the Security Association Database. Most probably they are stored in the Traffic Selector, which defines the traffic which has to be secured with IPsec. Table 2 shows some header fields which can be adopted from the Traffic Selector. The table provides the ROHC class of these values, as we use the ROHC terminology to describe the compression algorithms.

Field	Protocol	ROHC class
IP version	IP/IPv6	STATIC-KNOWN
Source Address	IP/IPv6	STATIC-DEF
Destination Address	IP/IPv6	STATIC-DEF
Next Header	IP/IPv6	STATIC
Source PORT	UPD/TCP	STATIC-DEF
Destination PORT	UPD/TCP	STATIC-DEF

Table 2: This values are carried in the Security Association.

6. IP Layer Compression

This section describes how the compression of the IP layer is performed. The compression of this layer mostly occurs when the peers have negotiated the IPsec tunnel mode.

The basic idea for IP layer compression is to remove the IP layer before Diet-ESP encrypts the Clear Text Data. Similarly, for incoming packet, Diet-ESP decrypts the ESP packet, and restores the IP layer by reading the IP address in the IPsec SAD. However, the IP address is not sufficient to restore the complete IP header as other fields must be considered. To appropriately describes the compression of the IP layer, this section uses the ROHC terminology and describes the associated profile.

The IP header is classified as shown in Table 3

Field	Class	Compression Method	Diet-ESP ROHC class	Data origin
Version	STATIC	removed	STATIC	TS
Traffic Class	CHANGING	removed	INFERRED	outer IP
Flow Label	STATIC-DEF	removed	STATIC-DEF	outer IP
Payload Length	INFERRED	removed	INFERRED	outer IP
Next Header	STATIC	removed	STATIC	TS
Hop Limit	RACH	removed	INFERRED	outer IP
Source Address	STATIC-DEF	removed	STATIC-DEF	TS
Destination Address	STATIC-DEF	removed	STATIC-DEF	TS

Table 3: Header classification for IPv6.

Version:
The IP version is specified in the SA and can be copied to the ROHC context, before the first packet is sent/received.

Traffic Class:
Traffic Class can be read from the outer IP header. Therefore the classification is changed to INFERRED.

Flow Label:
Flow Label can be read from the outer IP header. Therefore the classification is changed to INFERRED.

Next Header
The Next Header is stored in the protocol of the Traffic Selector and is fixed. It can be copied to the ROHC context, before the first packet is sent/received.

Hop Limit
The Hop Limit can be read from the outer IP header. Therefore the classification is changed to INFERRED.

Source Address:
The Source Address is fixed in the SA and can be copied to the ROHC context, before the first packet is sent/received.

Destination Address:
The Destination Address is fixed in the SA and can be copied to the ROHC context, before the first packet is sent/received.

7. UDP Transport Layer Compression

This section shows the compression of ESP payload for all ROHC profiles including an UDP header.

The UDP header is classified as shown in Table 4

Field	Class	Compr. Method	Diet-ESP ROHC class	Data origin
Source Port	STATIC-DEF	removed	STATIC-DEF	TS
Destination Port	STATIC-DEF	removed	STATIC-DEF	TS
Length	INFERRED	removed	INFERRED	IP payload length
Checksum	IRREGULAR	LSB	INFERRED	calc.

Table 4: Header classification for UDP.

Source Port:
The Source Port is fixed in the SA and can be copied to the ROHC context, before the first packet is sent/received.

Destination Port:
The Destination Port is fixed in the SA and can be copied to the ROHC context, before the first packet is sent/received.

Length:

The length of the UDP header can be calculated like: IP header - IP header length. Therefore there is no need to send it on the wire and it is defined as INFERRED.

Checksum:
The checksum can be calculated by Diet-ESP and proved by comparing the LSB sent on the wire. The number of bytes sent on the wire can be 0, 1 and 2 stored in TRANSPORT_CHECKSUM_LSB. If 0 LSB is chosen, the checksum MUST be decompressed with the value 0. If the UDP implementation of the sender chose to disable the UDP checksum by setting the checksum to 0 Diet-ESP SHOULD be used with TRANSPORT_CHECKSUM_LSB = 0.

8. UDP-Lite Transport Layer Compression

This section shows the compression of ESP payload for all ROHC profiles including an UDP-Lite header.

The UDP header is classified as shown in Table 5

Field	Class	Compression Method	Diet-ESP ROHC class	Data origin
Source Port	STATIC-DEF	removed	STATIC-DEF	TS
Destination Port	STATIC-DEF	removed	STATIC-DEF	TS
Checksum Coverage	IRREGULAR	LSB	IRREGULAR	calc.
Checksum	IRREGULAR	LSB	INFERRED	calc.

Table 5: Header classification for UDP-Lite.

Source Port:
The Source Port is fixed in the SA and can be copied to the ROHC context, before the first packet is sent/received.

Destination Port:
The Destination Port is fixed in the SA and can be copied to the ROHC context, before the first packet is sent/received.

Checksum Coverage:
The Checksum specifies the number of octets carried by the UDP-Lite checksum. It can have the same value as the UDP length (0 or UDP length) or any value between 8 and UDP length. This field is compressed with TRANSPORT_CHECKSUM_LSB of 0, 1 or 2 bytes. If 0 or 1 LSB is chosen, the field MUST be decompressed with the UDP

length. If 2 LSB is chosen, the checksum has to carry this behaviour.

Checksum:
The checksum can be calculated by Diet-ESP and proved by comparing the LSB sent on the wire. The number of bytes sent on the wire can be 0, 1 and 2 stored in TRANSPORT_CHECKSUM_LSB. If 0 LSB is chosen, the checksum MUST be decompressed with the value 0. If an UDP-lite implementation of the sender chose to disable the UDP checksum by setting the checksum to 0 Diet-ESP SHOULD be used with TRANSPORT_CHECKSUM_LSB = 0.

9. TCP Transport Layer Compression

This section shows the compression of ESP payload for all ROHC profiles including a TCP header. The ROHC context is partly filled while the Diet-ESP context exchange, wherefore some values can be removed. Since TCP is not stateless only fields with the compression methods 'removed' and 'LSB' are allowed to be compressed, the other fields MUST be sent on the wire uncompressed.

The UDP header is classified as shown in Table 6

Field	Class	Compression Method	Diet-ESP ROHC class	Data origin
Source Port	STATIC-DEF	removed	STATIC-DEF	TS
Destination Port	STATIC-DEF	removed	STATIC-DEF	TS
Sequence Number	CHANGING	LSB	CHANGING	ESP SN
Acknowledgement Num	INFERRED	N/A	INFERRED	
Data Offset	CHANGING	N/A	CHANGING	
Reserved	CHANGING	N/A	CHANGING	
CWR flag	CHANGING	N/A	CHANGING	
ECE flag	CHANGING	N/A	CHANGING	
URG flag	CHANGING	N/A	CHANGING	
ACK flag	CHANGING	N/A	CHANGING	
PSH flag	CHANGING	N/A	CHANGING	
RST flag	CHANGING	N/A	CHANGING	
SYN flag	CHANGING	N/A	CHANGING	
FIN flag	CHANGING	N/A	CHANGING	
Window	CHANGING	N/A	CHANGING	
Checksum	IRREGULAR	LSB	INFERRED	calc.
Urgent Pointer	CHANGING	N/A	CHANGING	
Options	CHANGING	N/A	CHANGING	

Table 6: Header classification for TCP.

Source Port:
The Source Port is fixed in the SA and can be copied to the ROHC context, before the first packet is sent/received.

Destination Port:
The Destination Port is fixed in the SA and can be copied to the ROHC context, before the first packet is sent/received.

Sequence Number:
The Sequence Number can be compressed with a LSB by using the SN stored in the SA for the remaining MSB not sent on the wire.

Checksum:
The checksum can be calculated by Diet-ESP and proved by comparing the LSB sent on the wire. The number of bytes sent on the wire can be 0, 1 and 2 stored in TRANSPORT_CHECKSUM_LSB. If 0 LSB is chosen, the checksum MUST be decompressed with the value 0. If an UDP-lite implementation of the sender chose to disable the UDP checksum by setting the checksum to 0 Diet-ESP SHOULD be used with TRANSPORT_CHECKSUM_LSB = 0.

10. IANA Considerations

There are no IANA consideration for this document.

11. Security Considerations

12. Acknowledgment

The current work on Diet-ESP results from exchange and cooperation between Orange, Ludwig-Maximilians-Universitaet Munich, Universite Pierre et Marie Curie. We thank Daniel Palomares and Carsten Bormann for their useful remarks, comments and guidances on the design. We thank Sylvain Killian for implementing an open source Diet-ESP on Contiki and testing it on the FIT IoT-LAB [fit-iot-lab] funded by the French Ministry of Higher Education and Research. We thank the IoT-Lab Team and the INRIA for maintaining the FIT IoT-LAB platform and for providing feed backs in an efficient way.

13. References

13.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3095] Bormann, C., Burmeister, C., Degermark, M., Fukushima, H., Hannu, H., Jonsson, L-E., Hakenberg, R., Koren, T., Le, K., Liu, Z., Martensson, A., Miyazaki, A., Svanbro, K., Wiebke, T., Yoshimura, T., and H. Zheng, "RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed", RFC 3095, July 2001.

[RFC3843] Jonsson, L-E. and G. Pelletier, "RObust Header Compression (ROHC): A Compression Profile for IP", RFC 3843, June 2004.

[RFC4019] Pelletier, G., "RObust Header Compression (ROHC): Profiles for User Datagram Protocol (UDP) Lite", RFC 4019, April 2005.

[RFC5225] Pelletier, G. and K. Sandlund, "RObust Header Compression Version 2 (ROHCv2): Profiles for RTP, UDP, IP, ESP and UDP-Lite", RFC 5225, April 2008.

[RFC5795] Sandlund, K., Pelletier, G., and L-E. Jonsson, "The RObust Header Compression (ROHC) Framework", RFC 5795, March 2010.

[RFC6846] Pelletier, G., Sandlund, K., Jonsson, L-E., and M. West, "RObust Header Compression (ROHC): A Profile for TCP/IP (ROHC-TCP)", RFC 6846, January 2013.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, October 2014.

13.2. Informational References

[I-D.mglt-6lo-diet-esp] Migault, D. and T. Guggemos, "Diet-ESP: a flexible and compressed format for IPsec/ESP", draft-mglt-6lo-diet-esp-00 (work in progress), January 2015.

[RFC5856] Ertekin, E., Jasani, R., Christou, C., and C. Bormann, "Integration of Robust Header Compression over IPsec Security Associations", RFC 5856, May 2010.

[fit-iot-lab] "Future Internet of Things (FIT) IoT-LAB", <https://www.iot-lab.info>.

Appendix A. Interaction with ROHC profiles

Each ROHC profile defines compression rules for a set of protocol headers. Table 7 clarifies how ROHC profiles can be mapped to Diet-ESP payload compression.

Profile Number	ROHC version	Protocol	RFC	Diet-ESP compression
0x0000	ROHC	uncompressed IP	[RFC3095]	no compression
0x0001	ROHC	RTP/UDP/IP	[RFC3095]	not used
0x1001	ROHCv2	RTP/UDP/IP	[RFC5225]	not used
0x0002	ROHC	UDP/IP	[RFC3095]	UDP and IP in Tunnel Mode
0x1002	ROHCv2	UDP/IP	[RFC5225]	UDP and IP in Tunnel Mode
0x0003	ROHC	ESP/IP	[RFC3095]	not used
0x1003	ROHCv2	ESP/IP	[RFC5225]	not used
0x0004	ROHC	IP	[RFC3843]	IP in Tunnel Mode
0x1004	ROHCv2	IP	[RFC5225]	IP in Tunnel Mode
0x0006	ROHC	TCP/IP	[RFC6846]	TCP and IP in Tunnel Mode
0x0007	ROHC	RTP/UDP-Lite/IP	[RFC4019]	not used
0x1007	ROHCv2	RTP/UDP-Lite/IP	[RFC5225]	not used
0x0008	ROHC	UDP-Lite/IP	[RFC4019]	UDP-Lite and IP in Tunnel Mode
0x1008	ROHCv2	UDP-Lite/IP	[RFC5225]	UDP-Lite and IP in Tunnel Mode

Table 7: Overview over currently existing ROHC profiles.

Appendix B. Document Change Log

00-First version published

Authors' Addresses

Daniel Migault (editor)
Ericsson
8400 boulevard Decarie
Montreal, QC H4P 2N2
Canada

Email: mglt.ietf@gmail.com

Tobias Guggemos (editor)
LMU Munich
Am Osteroesch 9
87637 Seeg, Bavaria
Germany

Email: tobias.guggemos@gmail.com