Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 29, 2017

R. Moskowitz
S. Hares
Huawei
I. Faynberg
Stargazers Consulting, LLC
H. Lu
Retired
P. Giacomin
FreeLance

June 27, 2017

GPCOMP
draft-moskowitz-gpcomp-02.txt

Abstract

This document describes a protocol intended to provide lossless compression for use within any datagram. It is particularly intended for use in encrypted datagrams where lower-level compression is ineffective.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 29, 2017.

Table of Contents

1. Introduction

Generic payload compression is a protocol to reduce the size of most datagrams. This protocol will increase the overall communication performance by compressing the datagrams, provided the participating devices have sufficient computation power, through either CPU capacity or a compression coprocessor, and the communication is over constrained links.

Generic payload compression is especially useful when encryption is applied to datagrams. Encrypting a datagram causes the data to be random in nature, rendering compression at lower protocol layers ineffective.

This document defines the Generic payload compression protocol (GPComp), a GPComp packet structure, the GPComp Association (GPCA), and several methods to negotiate the GPCA.

Other documents shall specify how a specific compression algorithm can be used with the Generic payload compression protocol. Such algorithms are beyond the scope of this document.

This document draws heavily on IPCOMP [RFC3173].

2. Terms and Definitions

2.1. Requirements 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 [RFC2119].

2.2. Definitions

GPCA: The Generic Payload Compression Protocol Association. This is the collection of attributes and values that define how GPComp operates.

3. Compression Process

The compression processing has two phases: compressing of outbound datagrams ("compression") and decompressing of inbound datagrams ("decompression"). The compression processing MUST be lossless, ensuring that the datagram, after being compressed and decompressed, is identical to the original datagram.

Each datagram is compressed and decompressed by itself without any relation to other datagrams ("stateless compression"), as datagrams may arrive out of order or not arrive at all.

Processing of inbound datagrams MUST support both compressed and non-compressed datagrams, in order to meet the non-expansion policy requirements, as defined in Section 3.3.

3.1. Compressed Payload

Compression is applied to a single datagram. The size of a compressed payload, generated by the compression algorithm, MUST be in whole octet units.

As compression is optional for each datagram associated within the GPCA, an identification mechanism is REQUIRED for each datagram. Minimally this can be a single option bit within the datagram's header (if it has one). Alternatively, the GPComp header, defined in Section 4.2, is inserted immediately preceding the compressed

payload. The receiving side MUST be able to distinguish between compressed and uncompressed payloads.

3.2. Uncompressing Conundrum

The receiver MUST be able to recognize the condition of no compression for the case where there is no datagram header option flag for compression and only the presense of the GPComp header indicates a compressed payload. In this case, the payload itself has no indication that GPComp is enabled for the payload, but there is nothing to decompress. The receiving process has to be able to identify the payload as lacking the GPComp header and act appropriately. Thus it is best if there is a datagram header compression flag (for example in SSE [I-D.moskowitz-sse]) and the GPComp header is not even used.

3.3. Non-Expansion Policy

If the total size of a compressed payload and the GPComp header (if present) is not smaller than the size of the original payload, the datagram MUST be sent in the original non-compressed form. To clarify: If an datagram is sent non-compressed, no GPComp header is added to the datagram. This policy ensures saving the decompression processing cycles and avoiding incurring datagram fragmentation if the expanded datagram is larger than the MTU. It does present a potential conundrum Section 3.2 to the receiver.

Small datagrams are likely to expand as a result of compression. Therefore, a numeric threshold should be applied before compression, where datagrams of size smaller than the threshold are sent in the original form without attempting compression. The numeric threshold is implementation dependent.

A datagram payload with compressed content tends not to compress any further. The previously compressed payload may be the result of external processes, such as compression applied by an upper layer in the communication stack, or by an off-line compression utility. An adaptive algorithm should be implemented to avoid the performance hit. For example, if the compression of i consecutive IP datagrams of an GPCA fails, the next several datagrams, say k, are sent without attempting compression. If then the next j datagrams also fail to compress, a larger number of datagrams, say k+n, are sent without attempting compression. Once a datagram is compressed successfully, the normal process of IPComp restarts. Such an adaptive algorithm, including all the related thresholds, is implementation dependent.

During the processing of the payload, the compression algorithm MAY periodically apply a test to determine the compressibility of the

processed data, similar to the requirements of [V42BIS]. The nature of the test is algorithm dependent. Once the compression algorithm detects that the data is non-compressible, the algorithm SHOULD stop processing the data, and the payload is sent in the original non-compressed form.

4. Compressed Datagram Structure

The compressed datagram structure for GPComp can be implied or explicit. The implied structure is used with datagrams that have a header field with option flags and a length field or end-of-datagram identifier. The explicit structure uses the GPComp header.

4.1. Implied Structure

The implied structure takes one option flag bit in the datagram header. This bit is ONE if that datagram is compressed or ZERO if not compressed. The compression algorithm is specified within the GPCA. The implied structure can be used within SSE.

4.2. GPComp header for Explicit Structure

The GPComp header is used for datagrams that do not have a defined header with an options field, or do not have an available bit in the header to flag compression status. IPFIX [RFC7011] and NETCONF [RFC6536] use such a datagram.

The GPComp header is identical to the IPComp header [RFC3173]. This is for done for simplicity sake. Although it is possible to design a GPComp header of only 2 bytes, this would break the typical 32 bit word alignment in Internet Protocol headers. In many uses, the Next Header field will be NULL; this is set by the GPCA.

5. Negotiating GPComp

The use of GPComp and its options (e.g. compression algorithm) should be part of the communication start up process. Although GPComp can be manually set up, this may result in a lack of agility in compression algorithm selection. That is, only one algorithm is used and cannot easily be changed. Thus manual set up for GPComp should be limited to testing needs.

An application may use any internal set up mechanism for negotiating GPComp. However, as compression is frequently used in conjunction with encryption, the application may call a Key Management Protocol (KMP) and request that the KMP set up GPComp.

5.1. The GPCA

The GPCA is a data structure that controls the operation of GPComp. The content of the GPCA is application dependent but it will always include the Compression Parameter Index (CPI) as defined in IPCOMP.

5.2. Using IKEv2

At set up, and application may call IKEv2 [RFC7296]. This may be to enable ESP in Transport Mode [RFC4303] or SSE for secure communications. It the same time, IKE may be instructed to negotiate IPCOMP, but the application will use the negotiated IPCOMP CPI for GPComp.

5.3. Using HIP

At set up, and application may call HIPv2 [RFC7401] or HIP-DEX [I-D.ietf-hip-dex]. This may be to enable ESP in BEET Mode [RFC7402] or SSE for secure communications.

HIP does not currently include a negotiation for compression. A GPCOMP_INFO parameter is proposed in [I-D.moskowitz-ssls-hip]. It is unclear at this time if this could also be used for IPCOMP, or if a separate parameter is needed for it.

6. IANA Considerations

In [I-D.moskowitz-ssls-hip], IANA is requested to assign a HIP parameter value for the Compression Transform.

7. Security Considerations

TBD

8. Contributors

TBD

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3173] Shacham, A., Monsour, B., Pereira, R., and M. Thomas, "IP Payload Compression Protocol (IPComp)", RFC 3173, DOI 10.17487/RFC3173, September 2001, <http://www.rfc-editor.org/info/rfc3173>.

9.2. Informative References

[I-D.ietf-hip-dex]
Moskowitz, R. and R. Hummen, "HIP Diet EXchange (DEX)", draft-ietf-hip-dex-05 (work in progress), February 2017.

[I-D.moskowitz-sse]
Moskowitz, R., Faynberg, I., Lu, H., Hares, S., and P. Giacomin, "Session Security Envelope", draft-moskowitz-sse-04 (work in progress), October 2016.

[I-D.moskowitz-ssls-hip]
Moskowitz, R., Xia, L., Faynberg, I., Hares, S., and P. Giacomin, "Secure Session Layer Services KMP via HIP", draft-moskowitz-ssls-hip-01 (work in progress), October 2016.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <http://www.rfc-editor.org/info/rfc4303>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <http://www.rfc-editor.org/info/rfc6536>.

[RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <http://www.rfc-editor.org/info/rfc7011>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <http://www.rfc-editor.org/info/rfc7296>.

[RFC7401] Moskowitz, R., Ed., Heer, T., Jokela, P., and T. Henderson, "Host Identity Protocol Version 2 (HIPv2)", RFC 7401, DOI 10.17487/RFC7401, April 2015, <http://www.rfc-editor.org/info/rfc7401>.

[RFC7402] Jokela, P., Moskowitz, R., and J. Melen, "Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP)", RFC 7402, DOI 10.17487/RFC7402, April 2015, <http://www.rfc-editor.org/info/rfc7402>.

[V42BIS] CCITT, "Data Compression Procedures for Data Circuit Terminating Equipment (DCE) Using Error Correction Procedures", Recommendation V.42 bis, January 1990.

Authors' Addresses

Robert Moskowitz
Huawei
Oak Park, MI 48237

Email: rgm@labs.htt-consult.com

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

Email: shares@ndzh.com

Igor Faynberg
Stargazers Consulting, LLC
East Brunswick, NJ 08816
USA

Email: igorfaynberg@gmail.com

Huilan Lu
Retired

Email: huilanlu2@gmail.com

Pierpaolo Giacomin
FreeLance

Email: yrz@anche.no