Internet Draft
Expires in six months

R. Friend
R. Monsour
Hi/fn, Inc.

February 6, 1998

IP Payload Compression Using LZS

<draft-ietf-ippcp-lzs-03.txt>

Status of this Memo

Abstract

This document describes a compression method based on the LZS compression algorithm. This document defines the application of the LZS algorithm to the IP Payload Compression Protocol [IPCOMP]. [IPCOMP] defines a method for applying lossless compression to the payloads of Internet Protocol datagrams.

Acknowledgments

The LZS details presented here are similar to those in PPP LZS-DCP Compression Protocol (LZS-DCP), [RFC-1967].

The author wishes to thank the participants of the IPPCP working group mailing list whose discussion is currently active and is working to generate the protocol specification for integrating compression with IP.

Table of Contents

1. Introduction

1.1 General

This document is a submission to the IETF IP Payload Compression Protocol (IPPCP) Working Group. Comments are solicited and should be addressed to the working group mailing list (ippcp@external.cisco.com) or to the editor.

This document specifies the application of LZS compression, a lossless compression algorithm, to IP datagram payloads. This document is to be used in conjunction with the IP Payload Compression Protocol [IPCOMP]. This specification assumes a thorough understanding of the IPComp protocol.

1.2 Background of LZS Compression

Starting with a sliding window compression history, similar to [LZ1], Hi/fn developed a new, enhanced compression algorithm identified as LZS. The LZS algorithm is a general purpose lossless compression algorithm for use with a wide variety of data types. Its encoding method is very efficient, providing compression for strings as short as two octets in length.

The LZS algorithm uses a sliding window of 2,048 bytes. During

compression, redundant sequences of data are replaced with tokens that represent those sequences. During decompression, the original sequences are substituted for the tokens in such a way that the original data is exactly recovered. LZS differs from lossy compression algorithms, such as those often used for video compression, that do not exactly reproduce the original data.

The details of LZS compression can be found in [ANSI94].

The efficiency of the LZS algorithm depends on the degree of redundancy in the original data. A table of compression ratios for the [Calgary] Corpus file set is provided in the appendix in Section 7.

1.3 Licensing

Hi/fn, Inc. holds patents on the LZS algorithm. Licenses for a reference implementation are available for use in IPPCP, IPSec, TLS and PPP applications at no cost. Source and object licenses are available on a non-discriminatory basis. Hardware implementations are also available. For more information, contact Hi/fn at the address listed with the authors' addresses.

1.4 Specification of Requirements

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].

2. Compression Process

2.1 Compression History

The sender MUST reset the compression history prior to processing each datagram's payload. This ensures that each datagram's payload can be decompressed independently of any other, as is needed when datagrams are received out of order.

The sender MUST flush the compressor each time it transmits a compressed datagram. Flushing means that all data going into the compressor is included in the output, i.e., no data is held back in the hope of achieving better compression. Flushing is necessary to prevent a datagram's data from spilling over into a later datagram.

2.2 Anti-expansion of Payload Data

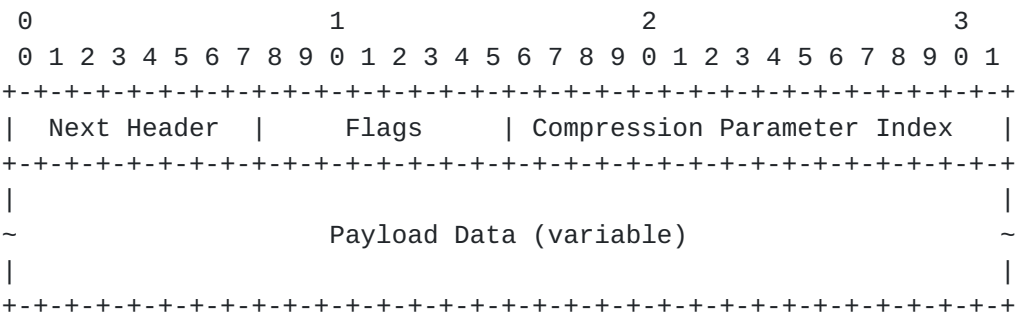
The maximum expansion produced by the LZS algorithm is 12.5%.

If the size of a compressed IP datagram, including the Next Header, Flags, and CPI fields, is not smaller than the size of the original

IP datagram, the IP datagram MUST be sent in the original non-compressed form, as described in [IPCOMP].

2.3 Format of Compressed Datagram Payload

The following is an example datagram that results when using LZS as the compression algorithm for the IP Payload Control Protocol. Note that the IP header has been omitted for clarity.



The Next Header, Flags, and Compression Parameter Index fields are all described in [IPCOMP].

2.4 Compression Encoding Format

The input to the payload compression algorithm is an IP datagram payload. The output of the algorithm is a new (and hopefully smaller) payload. The output payload contains the input payload's data in either compressed or uncompressed format. The input and output payloads are each an integral number of bytes in length.

If the uncompressed form is used, the output payload is identical to the input payload and the IPComp header is omitted. If the compressed form is used, the output payload is prepended with the IPComp header and encoded as defined in [ANSI94], which is repeated here for informational purposes ONLY.

```
<Compressed Stream> := [<Compressed String>] <End Marker>
<Compressed String> := 0 <Raw Byte> | 1 <Compressed Bytes>
<Raw Byte> := <b><b><b><b><b><b><b><b>          (8-bit byte)
<Compressed Bytes> := <Offset> <Length>

<Offset> := 1 <b><b><b><b><b><b><b> |           (7-bit offset)
            0 <b><b><b><b><b><b><b><b><b><b><b> (11-bit offset)
<End Marker> := 110000000

<b> := 1 | 0

<Length> :=
00        = 2     1111 0110      = 14
```

```
   01        = 3     1111 0111      = 15
   10        = 4     1111 1000      = 16
```

```
1100      = 5     1111 1001      = 17
1101      = 6     1111 1010      = 18
1110      = 7     1111 1011      = 19
1111 0000 = 8     1111 1100      = 20
1111 0001 = 9     1111 1101      = 21
1111 0010 = 10    1111 1110      = 22
1111 0011 = 11    1111 1111 0000 = 23
1111 0100 = 12    1111 1111 0001 = 24
1111 0101 = 13     ...
```

2.5 Padding

A datagram payload compressed using LZS always ends with the last compressed data byte (also known as the <end marker>), which is used to disambiguate padding. This allows trailing bits as well as bytes to be considered padding.

The size of a compressed payload MUST be in whole octet units.

3. Decompression Process

If the received datagram is compressed, the receiver MUST reset the decompression history prior to processing the datagram. This ensures that each datagram can be decompressed independently of any other, as is needed when datagrams are received out of order. Following the reset of the decompression history, the receiver decompresses the Payload Data field according to the encoding specified in section 3.2 of [ANSI94].

If the received datagram is not compressed, the receiver needs to perform no decompression processing and the Payload Data field of the datagram is ready for processing by the next protocol layer.

4. IPComp Association (IPCA) Parameters

ISAKMP MAY be used to negotiate the use of the LZS compression method to establish an IPCA, as defined in [IPCOMP].

4.1 ISAKMP Transform ID

The LZS Transform ID as 0x03, as specified in The Internet IP Security Domain of Interpretation [SECDOI]. This value is used to negotiate the LZS compression algorithm under the ISAKMP protocol.

4.2 ISAKMP Security Association Attributes

There are no other parameters required for LZS compression negotiated under ISAKMP.

4.3 Manual configuration

The CPI value 0x03 is used for a manually configured IPComp Security Associations.

4.4 Minimum packet size threshold

As stated in [IPCOMP], small packets may not compress well. Informal tests using the LZS algorithm over the Calgary Corpus data set show that the average payload size that may produce expanded data is approximately 90 bytes. Thus implementations may not want to attempt to compress payloads smaller than 90 bytes.

4.5 Compressibility test

There is no adaptive algorithm embodied in the LZS algorithm, for compressibility testing, as referenced in [IPCOMP].

5. Security Considerations

IP payload compression potentially reduces the security of the Internet, similar to the effects of IP encapsulation [RFC-2003]. For example, IPComp makes it difficult for border routers to filter datagrams based on header fields. In particular, the original value of the Protocol field in the IP header is not located in its normal positions within the datagram, and any transport-layer header fields within the datagram, such as port numbers, are neither located in their normal positions within the datagram nor presented in their original values after compression. A filtering border router can filter the datagram only if it shares the IPComp Association used for the compression. To allow this sort of compression in environments in which all packets need to be filtered (or at least accounted for), a mechanism must be in place for the receiving node to securely communicate the IPComp Association to the border router. This might, more rarely, also apply to the IPComp Association used for outgoing datagrams.

When IPComp is used in the context of IPSec, it is not believed to have an effect on the underlying security functionality provide by the IPSec protocol; i.e., the use of compression is not known to degrade or alter the nature of the underlying security architecture or the encryption technologies used to implement it.

6. References

[AH] Kent, S. and Atkinson, R., "IP Authentication Header", draft-ietf-ipsec-auth-header-01.txt, Work in Progress, July 1997.
[ANSI94] American National Standards Institute, Inc., "Data Compression Method for Information Systems," ANSI X3.241-1994, August 1994.

[Calgary] Text Compression Corpus, University of Calgary, available at ftp://ftp.cpsc.ucalgary.ca/pub/projects/text.compression.corpus.

[IPCOMP] Shacham, A., "IP Payload Compression Protocol (IPComp)", draft-ietf-ippcp-protocol-01.txt, Work in Progress, October 1997.

[LZ1] Lempel, A. and Ziv, J., "A Universal Algorithm for Sequential Data Compression", IEEE Transactions On Information Theory, Vol. IT-23, No. 3, May 1977.

[RFC-1962] Rand, D., "The PPP Compression Control Protocol (CCP)", RFC-1962, June 1996.

[RFC-1967] K. Schneider, R. Friend, "PPP LZS-DCP Compression Protocol (LZS-DCP)", RFC-1967, August, 1996.

[RFC-2003] Perkins, C., "IP Encapsulation within IP", RFC 2003, October 1996.

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

[SECDOI] Piper, D., "The Internet IP Security Domain of Interpretation for ISAKMP", Internet-Draft: draft-ietf-ipsec-ipsec-doi-06.txt, Work in Progress, November 1997.

7. Authors Addresses

Robert Friend
Hi/fn Inc.
5973 Avenida Encinas
Suite 110
Carlsbad, CA 92008
Email: rfriend@hifn.com

Robert Monsour
Hi/fn Inc.
2105 Hamilton Avenue
Suite 230
San Jose, CA 95125
Email: rmonsour@hifn.com

8. Appendix: Compression Efficiency versus Datagram Size

The following table offers some guidance on the compression efficiency that can be achieved as a function of datagram size. Each entry in the table shows the compression ratio that was achieved when LZS was applied to a test file using datagrams of a specified size.

The test file was the University of Calgary Text Compression Corpus [Calgary]. The Calgary Corpus consists of 18 files with a total size (all files) of 3.278MB.

Datagram size, bytes	64	128	256	512	1024	2048	4096	8192	16384
Compression ratio	1.18	1.28	1.43	1.58	1.74	1.91	2.04	2.11	2.14