SSE BOF
Internet-Draft
Intended status: Standards Track
Expires: December 29, 2017

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

June 27, 2017

Session Security Envelope
draft-moskowitz-sse-05

Abstract

This memo specifies the details of the Session Security Envelope (SSE). SSE is a session protocol aiming to guarantee confidentiality, integrity and authentication completely independently by the underlying context, namely network and transport layers. A single session using the SEE protocol can include a single transport session or multiple transport sessions. This mean that SSE can survive the break-down in network and transport layers or to attacks carried against them. SSE is also applicable in networks lacking in classic inter-networking and transport protocols SSE relies on modern AEAD block cipher modes of operations, a class of block cipher modes which allows, at the same time, to authenticate the message while encrypting a part of it.

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

This memo specifies the details of the Session Security Envelope (SSE). SSE is a session protocol aiming to guarantee confidentiality, integrity and authentication completely independently by the underlying context, namely network and transport layers. A single SSE session can span a single transport session or multiple transport sessions. These transport sessions can use the same transport layer protocol (E.g. TCP) or use different transport protocols. SSE can survive the break-down in network and transport layers or to attacks carried against them. Moreover SSE will relies on modern AEAD block cipher modes of operations, a class of block cipher modes which allows, at the same time, to authenticate the message while encrypting a part of it.

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. Notations

This section will contain notations

2.3. Definitions

AEAD Block Cypher: (definition needed)

SSE: Session Specific Envelope

3. SSE Security Boundary

The security boundary comes at layer above the IP transport layers (TCP, SCTP, UDP). This security allows the data to be secure prior to entering into a specific transport layer. A single SSE session can span 1 or N transport protocol connections. The multiple transport connections running under an SSE session may all use one protocol (e.g. TCP) or multiple protocols (e.g. TCP, SCTP, UDP). The higher layer security boundary provides a common security layer.

4. API

The initial API is part of a shim with socket call over a TCP socket.

s = int socket(int domain, int type, int protocol)

```
   where:

      domain: AF_INET and AF_INET6 supported

      type: SOCK_SECURE

      protocol: Transport protocol (TCP (6), UDP (6), SCTP (132))


    int setsockopt(int sockfd, int level, int optname,
                const void *optval, socklen_t optlen);

   int getsocketopt(int sockfd, int level, int optname
                      const void *optval, socket
    where:
     sockfd:      # socket file descriptor
     optname:     # option name (see below)
     optval;          # points to *sse_transport structure;
     optlen;          # length of option

     optval values:
     ADD_SSE_Transport[1];     # add transport to SSE
     DELETE_SSE_Transport[2];  # delete transport to SSE
     Query_SSE_Transport [3];  # Query transport

     optval        *sse_transport[MAX_SSE_TRANSPORTS]; - for add/deletes


    struct *sse_add_transport
           int nt_sockfd;   # new transport socket
           int protocol;    # new protocol
           );


  int getsockopt(int sockfd, int level, int optname,
                void *optval, socklen_t *optlen);

  int setsockopt(int sockfd, int level, int optname,
                const void *optval, socklen_t optlen);
```

Figure 1 - Example SSE Socket API

Note: The prototype for this SECURE_SOCKET is on a FREEBSD OS.

5. Packet format

An SSE PDU is a Session Layer PDU (SPDU). In order to accommodate various use cases three formats are available for the PDU. The only difference between those formats is the size of length and sequence number fields. Following these fields is the encrypted payload and Integrity Check Value (ICV). Encrypted payload and ICV has a substructure depending on the choice of encryption algorithm and mode.

5.1. SSE compact format

SSE compact format aims to provide a Session Security Layer to applications leveraging on constrained network media with packet size limitations or high cost per bit transport.

In the SSE compact format:

SPI is 24 bits.

FLAGS is 8 bits.

Length is 12 bits

Sequence Number is 20 bits

12 bits of Length allows (2^12) 4096 bytes in the Encrypted Payload (does not include the ICV). 20 bits in the Sequence Number allows to send (2^20) 1048576 packets before renegotiating the key. (The ICV length is set by the KMP parameters, so the length is known and therefore is not included in the length calculation)

The SPI internally is 32 bits to maintain SPI length consistancy. The high order 8 bits are always ZERO, allowing for only sending the lower 24 bits in the header.

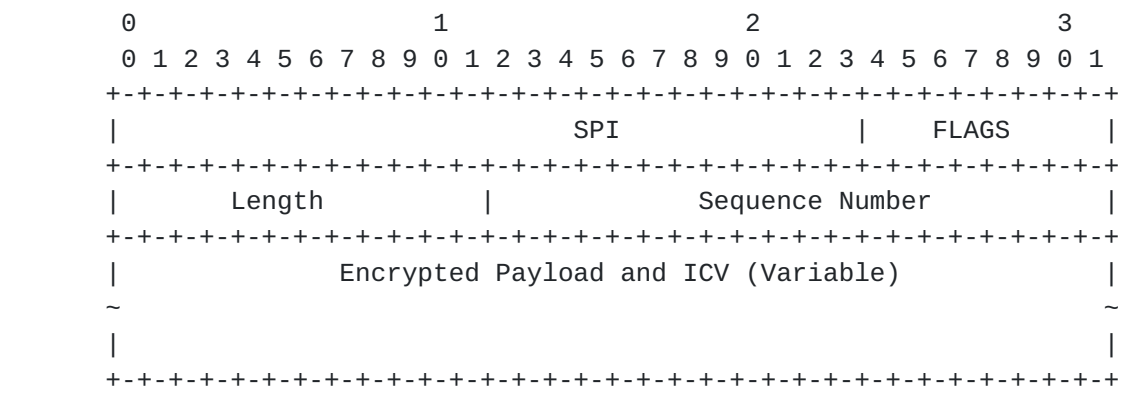

Figure 2 - Compact format

5.2. SSE Large Format

SSE large format aims provide a Session Security Layer to applications which have common sizes of transport packets.

In the SSE compact format:

SPI is 32 bits.

FLAGS is 8 bits.

Length is 32 bits

Sequence Number is 32 bits

32 bits of Length allows (2^32)or ~4Gbytes in the Encrypted Payload (does not include the ICV). 32 bits in the Sequence Number allows to send (2^32) ~40 billion packets packets before renegotiating the key.

The 32 bits of length allows an IPv6 jumbogram to be included as in the SSE Large Format Payload

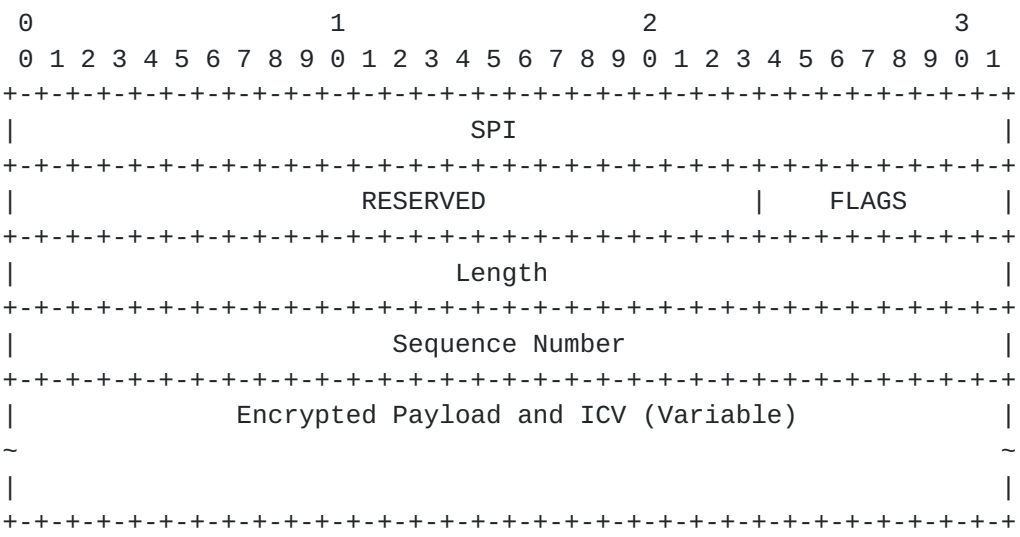

Figure 3 - Large Format

5.3. SSE Extreme Format

SSE large format aims provide a Session Security Layer to high performance networks.

In the SSE compact format:

SPI is 32 bits.

FLAGS is 8 bits.

Length is 32 bits

Sequence Number is 64 bits

32 bits of Length allows (2^32) 4294967296 bytes (4Gbytes) in the Encrypted Payload (excluding the ICV). 32 bits in the Sequence Number allows to send (2^64) 18446744073709551616 (around 18 * 10^18) packets before renegotiating the key.

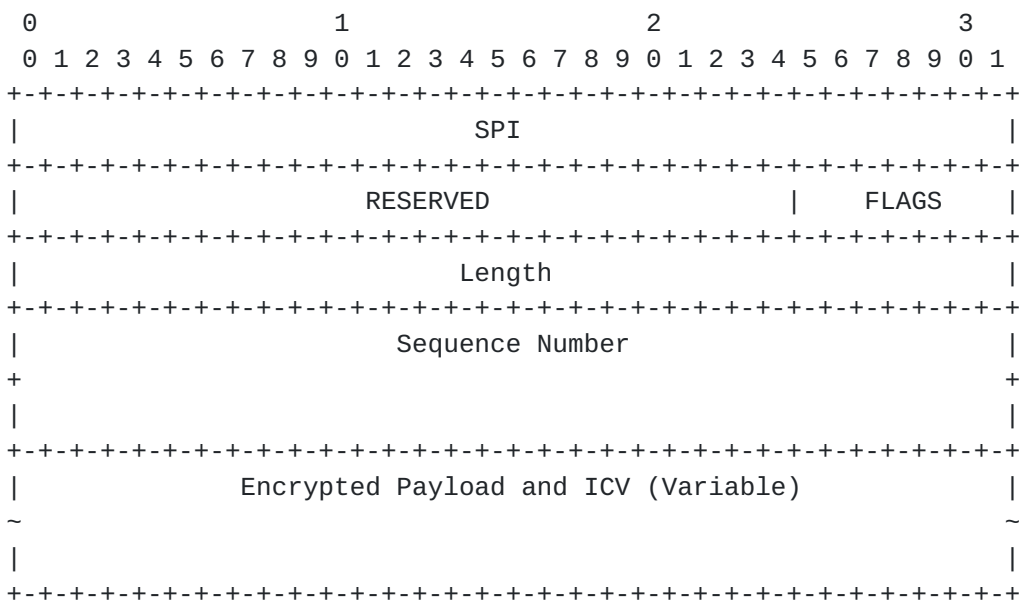

Figure 4 - Extreme Format

5.4. Header Fields

SPI is the Security Parameter Index, a 32 bit number received from the external KMP. It is the index into the Security Association and is typically unidirectional. That is each direction in has its own SPI. A KMP for a unicast communication would provide the two SPIs. Multicast is different. Depending on the requirements, there can be one SPI for all transmitters or one per transmitter.

The compact format only transmits 24 bits of the 32 bit SPI. The SPI is internally kept as both the 32 bits SPI from the KMP and a 24 bit truncated SPI (with the 8 high order bits of zero). If this truncation results in a duplicate SPI, the negotiation is rejected and the KMP is called again.

Length is the length in bytes of the encrypted payload. This does not include the ICV. The length of the ICV depends on the block cipher settings.

FLAGS is a set of 8 options flags. Bit 7 is the GPComp [I-D.moskowitz-gpcomp] bit compression option bit.

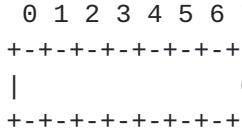

Figure 5 - FLAGS field

Sequence Number is a, strictly increasing by 1, counter. When the field cannot be increased without wrapping a key renegotiation MUST be performed. Please note that this Sequence Number has not the same meaning and implications of a Transport Layer sequence number, hence increasing by 1 is a good idea.

Note: It is common practice to rekey some time BEFORE the number space is exhausted.

5.5. AEAD integration

SSE MUST use AEAD block cipher modes. AEAD block cypher modes will ensure confidentiality on the payload and integrity of both the payload and the headers (SPI, length and sequence number).

6. Packet processing and State Machine

SSE will spawn across several ports and protocols, hence each listened port and protocol can be a different SSE instance. See Architecture Draft.

6.1. Establishing a session

An application can establish a session via the SSE API, which in turn will interact with a KMP daemon. SSE instance will get all parameters related to the session from the KMP daemon.

Editorial note: Is this a local vulnerability?

6.2. Processing Outgoing Application Data

After having established an SSE session, an application can send application- level data using the normal socket calls. The SSE layer will encapsulate the packet, and send it on the appropriate transport session. The application doesn't need to know SPI, sequence number or key. The local SSE knows these facts, and keeps it within the SSE data associated with a set of transport connections.

6.3. Processing Incoming Application Data

After having established an SSE session, the packets will be sent to the transport layer for de-encapsulation. After header removal, the socket processing will hand it to the SEE processing for security check. If the packet is deemed secure, the socket will remove the SSE envelope. The application see the byte stream as data from a transport connection.

The application doesn't need to know SPI, sequence number or key, relying on a fake connection. (but its local SSE instance knows it, hence the application own memory where those are stored).

7. Negotiating SSE

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

7.1. Using IKEv2

At set up, and application may call IKEv2 [RFC7296]. Currently there are no defined options for SSE in IKEv2 and it have to be amended. It should be able to follow ESP in Transport Mode [RFC4303].

7.2. Using HIP

At set up, and application may call HIPv2 [RFC7401] or HIP-DEX [I-D.ietf-hip-dex].

HIP does not currently include a negotiation for SSE. SSE can be added by assigning a HIP parameter value for an SSE Transform that is higher than ESP. A value of 4101 can be used for this purpose. The negotiation will mirror the ESP transform negotiation [RFC7402] and be carried in the R1 and I2 payloads as is ESP transform. This parameter and negotiation may be explicitly expanded here at in a later revision.

8. IANA Considerations

IANA is requested to assign a HIP parameter value for the SSE Transform. This parameter value should be higher than ESP. A value of 4101 is recommended.

9. Security Considerations

As SSE uses an AEAD block cipher, it is vulnerable to attack if a sequence number is reused for a given key. Thus implementations of SSE MUST provide for rekeying prior to Sequence Number rollover. An implementation should never assume that for a given context, the sequence number space will never be exhausted. Key Management Protocols like IKEv2 [RFC7296] or HIP [RFC7401] could be used to provide for rekeying management. The KMP SHOULD not create a network layer fate-sharing limitation.

As any security protocol can be used for a resource exhaustion attack, implementations should consider methods to mitigate flooding attacks of messages with valid SPIs but invalid content. Even with the ICV check, resources are still consumed to validate the ICV.

SSE makes no attempt to recommend the ICV length. For constrained network implementations, other sources should guide the implementation as to ICV length selection. The ICV length selection SHOULD be the the responsibility of the KMP.

As with any layered security protocol, SSE makes no claims of protecting lower or higher processes in the communication stack. Each layer's risks and liabilities need be addressed at that level.

10. References

10.1. 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>.

10.2. 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-gpcomp] Moskowitz, R., Hares, S., Faynberg, I., Lu, H., and P. Giacomin, "GPCOMP", draft-moskowitz-gpcomp-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>.

[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>.

Authors' Addresses

Robert Moskowitz
HTT Consulting
Oak Park, MI 48237

Email: rgm@labs.htt-consult.com

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

Email: igorfaynberg@gmail.com

Huilan Lu
Retired

Email: huilanlu2@gmail.com

Susan Hares
Hickory Hill Consulting
7453 Hickory Hill
Saline, MI 48176
USA

Email: shares@ndzh.com

Pierpaolo Giacomin
FreeLance

Email: yrz@anche.no