Workgroup: Network Working Group
Internet-Draft:
draft-tuexen-tsvwg-sctp-zero-checksum-03
Published: 10 May 2023
Intended Status: Standards Track
Expires: 11 November 2023
Authors: M. Tüxen — Münster Univ. of Appl. Sciences
V. Boivie — Google
F. Castelli — Google
R. Jesup — Mozilla

Zero Checksum for the Stream Control Transmission Protocol

Abstract

The Stream Control Transmission Protocol (SCTP) uses a 32-bit checksum in the common header of each packet to provide some level of data integrity. When the lower layer used by SCTP provides already the same or a higher level of data integrity, computing this checksum does not provide any additional protection, but does require computing resources. This document provides a simple extension to SCTP allowing to save these computing resources by using the constant 0 as the checksum in a backwards compatible way.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

SCTP as specified in [RFC9260] uses a CRC32c to provide some level of data integrity. When using, for example, Datagram Transport Layer Security (DTLS) as the lower layer for SCTP as specified in [RFC8261], using the CRC32c does not provide any additional protection over the one already provided by DTLS. However, computing the CRC32c at the sender and receiver side does require computationally resources for no benefit. This is in particular important for computational limited end points using SCTP encapsulated in DTLS.

The extension described in this document allows an SCTP end point to declare that it accepts SCTP packets with a checksum of zero. This declaration happens during the setup of the SCTP association and allows end points supporting this extension to be interoperable with end points not supporting the extension described in this document. To provide this backwards compatibility, end points using this extension still need to implement the CRC32c checksum algorithm.

2. Conventions

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and

"**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. A New Chunk Parameter

The Zero Checksum Acceptable Chunk Parameter is defined by the following figure.

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 0x8001 (suggested)   |          Length = 4           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: Zero Checksum Acceptable Chunk Parameter

Type: 16 bits (unsigned integer)
This field holds the IANA defined parameter type for the "Zero Checksum Acceptable" chunk parameter. IANA is requested to assign the value 32769 (0x8001) (suggested) for this parameter type.

Length: 16 bits (unsigned integer)
This field holds the length in bytes of the chunk parameter; the value **MUST** be 4.

All transported integer numbers are in "network byte order" a.k.a., Big Endian.

The Zero Checksum Acceptable Chunk Parameter **MAY** appear in INIT and INIT ACK chunks. It **MUST NOT** appear in any other chunk.

If an end point not supporting the extension described in this document receives this parameter in an INIT or INIT ACK chunk, it skips this parameter and continues to process further parameters in the chunk. This behavior is **REQUIRED** by [RFC9260] because the highest-order two bits of the Type are '10'.

4. Procedures

4.1. Declaration of Feature Support

If the lower layer of SCTP provides an equal or better level of data integrity protection than the one provided by using the CRC32c algorithm, the computation of the CRC32c checksum requires computational resources without providing any benefit. To avoid this, an SCTP end point **MAY** be willing to accept SCTP packets with an incorrect CRC32c checksum value of zero in addition to SCTP packets with correct CRC32c checksum values. An SCTP endpoint **SHOULD NOT** be willing to accept SCTP packets with an incorrect CRC32c checksum value of zero, if the lower layer does not provide at least the level of data integrity the CRC32c checksum algorithm provides.

One example of such a lower layer is the use of SCTP over DTLS as described in [RFC8261] (as used in the WebRTC context). Counter examples include:

* SCTP over IPv4 or IPv6 as specified in [RFC9260].
* SCTP over UDP as specified in [RFC6951].
* The use of SCTP Authentication as specified in [RFC4895].

Therefore using an incorrect zero checksum, in particular when using SCTP over DTLS, will not result in problems with the middle boxes expecting correct CRC32c checksums in SCTP packets.

An SCTP implementation **MAY** also require the upper layer to indicate that it is fine to accept SCTP packets with an incorrect CRC32c value of zero.

An end point willing to accept SCTP packets with a checksum of zero **MUST** include the Zero Checksum Acceptable Chunk Parameter in the INIT or INIT ACK chunk it sends.

4.2. Sender Side Considerations

If an end point has received an INIT or INIT ACK chunk containing a Zero Checksum Acceptable Chunk Parameter from its peer during the association setup, it **SHOULD** use zero as the checksum for all packets sent in this association with the following three exceptions:

* When an end point sends a packet containing an INIT chunk, it **MUST** include a correct CRC32c checksum in the packet containing the INIT chunk.
* When an end point sends a packet containing a COOKIE ECHO chunk, it **MUST** include a correct CRC32c checksum in the packet containing the COOKIE ECHO chunk.
* When an end point supports the dynamic address reconfiguration specified in [RFC5061] and sends a packet containing an ASCONF chunk, it **MUST** include a correct CRC32c checksum in the packet containing the ASCONF chunk.

The first exception allows backwards compatibility and the second and third exception allow a simpler implementation of the extension defined in this document.

When an end point responds to an "Out of the Blue" (OOTB) SCTP packet, it **MUST** include a correct CRC32c checksum in the response packet.

An SCTP end point **MAY** only send packets with an incorrect checksum of zero, if the upper layer allowed the reception of SCTP packets with an incorrect zero checksum.

4.3. Receiver Side Considerations

Zero is a valid result of the CRC32c algorithm. Therefore, a receiver of an SCTP packet containing a checksum value of zero cannot determine whether the sender included an incorrect CRC32c of zero to reduce the CPU cost or the result of the CRC32c computation was actually zero. However, if the receiver has sent the Zero Checksum Acceptable Chunk Parameter during the handshake, this ambiguity is irrelevant, since the receiver is fine with not using the CRC32c to protect incoming packets.

If an end point has sent the Zero Checksum Acceptable Chunk Parameter in an INIT or INIT ACK chunk, it **MUST** accept SCTP packets using an incorrect checksum value of zero in addition to SCTP packets containing the correct CRC32c checksum value for this association.

An SCTP implementation **MAY** process OOTB SCTP packets having an incorrect zero checksum in addition to OOTB packets with a correct CRC32c checksum.

5. Socket API Considerations

This section describes how the socket API defined in [RFC6458] needs to be extended to provide a way for the application to control the acceptance of a zero checksum.

Please note that this section is informational only.

A socket API implementation based on [RFC6458] is extended by supporting one new write-only IPPROTO_SCTP-level socket option.

5.1. Set Accepting a Zero Checksum (SCTP_ACCEPT_ZERO_CHECKSUM)

This socket option can be used to control the acceptance of a zero checksum. It is a write-only socket option and applies only to future SCTP associations on the socket.

This option expects an integer boolean flag, where a non-zero value enables the option, and a zero value disables the option. An implementation might only send packets with an incorrect checksum of zero, if this option is enabled.

This option is disabled by default.

6. IANA Considerations

[NOTE to RFC-Editor: "RFCXXXX" is to be replaced by the RFC number you assign this document.]

[NOTE to RFC-Editor: The suggested value for the parameter type is tentative and to be confirmed by IANA.]

This document (RFCXXXX) is the reference for the registration described in this section.

A new chunk parameter type has to be assigned by IANA. This requires an additional line in the "Chunk Parameter Types" registry for SCTP:

ID Value	Chunk Parameter Type	Reference
32769 (suggested)	Zero Checksum Acceptable (0x8001 (suggested))	[RFCXXXX]

Table 1: New entry in "Chunk Parameter Types" registry

7. Security Considerations

This document does not change the considerations given in [RFC9260].

8. References

8.1. 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, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, DOI 10.17487/RFC5061, September 2007, <https://www.rfc-editor.org/info/rfc5061>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8261] Tuexen, M., Stewart, R., Jesup, R., and S. Loreto, "Datagram Transport Layer Security (DTLS) Encapsulation of SCTP Packets", RFC 8261, DOI 10.17487/RFC8261, November 2017, <https://www.rfc-editor.org/info/rfc8261>.

[RFC9260]
Stewart, R., Tüxen, M., and K. Nielsen, "Stream Control Transmission Protocol", RFC 9260, DOI 10.17487/RFC9260, June 2022, <https://www.rfc-editor.org/info/rfc9260>.

8.2. Informative References

[RFC4895] Tuexen, M., Stewart, R., Lei, P., and E. Rescorla, "Authenticated Chunks for the Stream Control Transmission Protocol (SCTP)", RFC 4895, DOI 10.17487/RFC4895, August 2007, <https://www.rfc-editor.org/info/rfc4895>.

[RFC6458] Stewart, R., Tuexen, M., Poon, K., Lei, P., and V. Yasevich, "Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)", RFC 6458, DOI 10.17487/RFC6458, December 2011, <https://www.rfc-editor.org/info/rfc6458>.

[RFC6951] Tuexen, M. and R. Stewart, "UDP Encapsulation of Stream Control Transmission Protocol (SCTP) Packets for End-Host to End-Host Communication", RFC 6951, DOI 10.17487/RFC6951, May 2013, <https://www.rfc-editor.org/info/rfc6951>.

Acknowledgments

The authors wish to thank Gorry Fairhurst, Mike Heard, Nils Ohlmeier, Claudio Porfiri, and Magnus Westerlund for their invaluable comments.

Authors' Addresses

Michael Tüxen
Münster University of Applied Sciences
Stegerwaldstrasse 39
48565 Steinfurt
Germany

Email: tuexen@fh-muenster.de

Victor Boivie
Google
Kungsbron 2
SE-11122 Stockholm
Sweden

Email: boivie@google.com

Florent Castelli
Google

Kungsbron 2
SE-11122 Stockholm
Sweden

Email: orphis@google.com

Randell Jesup
Mozilla Corporation
1835 Horse Shoe Trl
Malvern, PA 19355
United States of America

Email: randell-ietf@jesup.org