TCP Maintenance & Minor Extensions (tcpm)
Internet-Draft
Updates: 793, 2018, 5925, 7323 (if approved)
Intended status: Standards Track
Expires: September 14, 2017

J. Looney
Netflix

March 13, 2017

64-bit Sequence Numbers for TCP
draft-looney-tcpm-64-bit-seqnos-00

Abstract

This draft updates RFC 793 to allow the optional use of 64-bit sequence numbers. It also updates other standards to support the extended sequence number space.

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 September 14, 2017.

Copyright Notice

Table of Contents

1. Introduction

RFC 793 [RFC0793] specifies the sequence number space as a 32-bit space. This means that the sequence number space will wrap in 2**32 bytes. On a 10-Gb/s network, this can occur in approximately 3.5 seconds. On a 100-Gb/s network, this can occur in approximately 350 milliseconds. While sequence number wrapping is a basic feature of TCP, the specified wrapping mechanism only supports having a theoretical maximum of 2**31 bytes outstanding at any given time. Additionally, when you are re-using sequence number space in such a short timeframe, it is unclear that the existing mechanisms for

detecting duplicate packets will be sufficient. To practically support these very high-speed networks, it is necessary to expand the sequence number space.

In addition to the base TCP specification, a number of other specifications have made assumptions about sequence numbers being 32 bits long. This document updates some of those specifications and provides guidance on interaction with other specifications.

1.1. Design Goals

This document assumes the following design goals:

- Support 64-bit sequence numbers.
- Maintain the existing header format.
- Maintain backwards compatibility with TCP implementations (including middle boxes) that only support 32-bit sequence numbers.
- Require minimal changes for any features that assume 32-bit sequence numbers.
- Use minimal TCP option space.

1.2. Overview of Implementation

This document specifies that the least significant 32 bits of the sequence number will continue to be carried in the Sequence Number and Acknowledgment Number fields of the standard TCP header. The most significant 32 bits will be carried in a new TCP option.

This mechanism provides an easy way to negotiate the option on startup: hosts that understand 64-bit sequence numbers can include the option with the SYN. If the other host does not understand the 64-bit Sequence Number Option, it will ignore the option and use the 32-bit sequence number already contained in the standard TCP header. When the initiating host receives a SYN/ACK that does not contain the 64-bit sequence number option, it simply reverts to normal 32-bit operation.

This method of negotiation and operation bears some similarity to the TCP Timestamp Option [RFC7323], which has been widely deployed without evident problems.

1.3. Backwards Compatibility

This document proposes a mechanism for providing backwards compatibility with existing TCP implementations that only support 32-bit sequence numbers. The document takes advantage of the fact that the least-significant 32-bits of the 64-bit sequence number should have the same properties as the normal 32-bit sequence number: it is the same size, should be seeded to be as random as 32-bit sequence numbers, and should continue to wrap as expected.

1.4. Requirements Language

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. Extended Sequence Numbers

This document allows the use of 64-bit sequence numbers if both endpoints of a TCP connection agree to use them. If both endpoints agree to use them, the endpoints should store 64 bits of sequence number and acknowledgment number information and should conduct all operations on these values using modulo 2**64 arithmetic.

Although a host is free to store the 64-bit sequence number information in whatever format it desires, this document make a logical distinction between the most-significant 32 bits and the least-significant 32 bits of sequence number information. This division is represented in Figure 1.

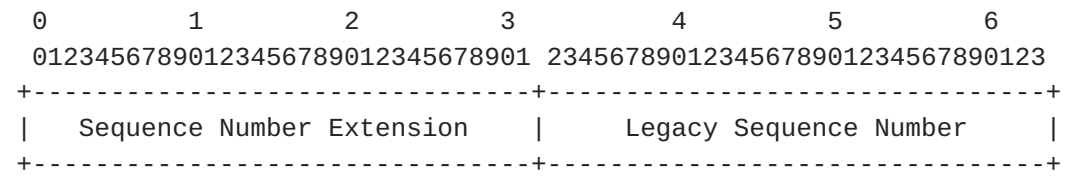

Figure 1: Logical division of a 64-bit sequence number

In Figure 1, the least-significant 32 bits of the sequence number are labeled the "Legacy Sequence Number". This is the portion of the sequence number that is stored in the Sequence Number field of the standard TCP header defined in [RFC0793]. In Figure 1, the most-significant 32 bits of the sequence number are labeled the "Sequence Number Extension". This is the portion of the sequence number that is stored in the 64-bit Sequence Number Option, which is defined in this document.

The 64-bit acknowledgment number is divided in the same way. For completeness, the acknowledgment number logical divisions are shown in Figure 2.

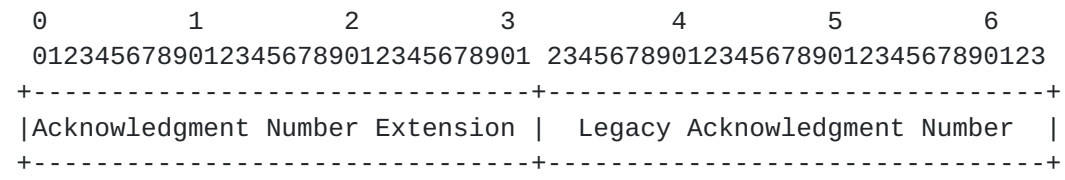

Figure 2: Logical division of a 64-bit acknowledgment number

In Figure 2, the least-significant 32 bits of the acknowledgment number are labeled the "Legacy Acknowledgment Number". This is the portion of the acknowledgment number that is stored in the Acknowledgment Number field of the standard TCP header defined in [RFC0793]. In Figure 2, the most-significant 32 bits of the acknowledgment number are labeled the "Acknowledgment Number Extension". This is the portion of the acknowledgment number that is stored in the 64-bit Sequence Number Option, which is defined in this document.

2.1. The 64-bit Sequence Number Option

The 64-bit Sequence Number Option is used to carry the most-significant 32 bits of the 64-bit sequence number information. It is also used to signal support for 64-bit sequence numbers in TCP segments with the SYN flag set.

The 64-bit Sequence Number Option will use TCP Option Kind TBD1. Its general form is shown in Figure 3.

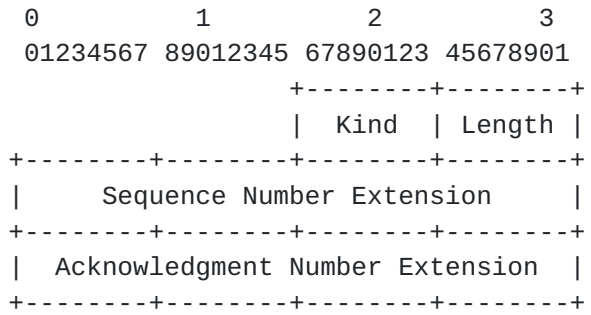

Figure 3: The 64-bit Sequence Number Option

Prior to standardization action, implementations should use the mechanism described in [RFC6994] to encode the option. IANA has reserved experiment ID (ExID) TBD2 for the option described in this document. This option format is shown in Figure 4.

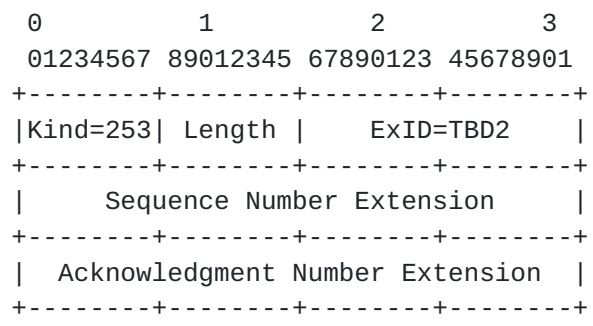

Figure 4: The 64-bit Sequence Number Option with ExID

In both cases, the fields are described in more detail below:

Length
: The total length (in octets) of the option (including Kind, Length, and, if applicable, ExID), as specified in [RFC0793].

Sequence Number Extension
: The most-significant 32 bits of the 64-bit sequence number.

Acknowledgment Number Extension
: For a segment with the ACK flag set, this field contains the most-significant 32 bits of the 64-bit sequence number. If the ACK flag is not set, this field is omitted (and, consequently, the option is 4 octets shorter).

2.2. Operation of the 64-bit Sequence Number Option

In order to use 64-bit sequence numbers, it is necessary for both hosts to negotiate the use of 64-bit sequence numbers. Further, it is necessary to ensure that no middlebox that is unaware of 64-bit sequence numbers is going to modify sequence number information. This section describes the initial negotiation to satisfy these parameters.

2.2.1. Choice of Initial Sequence Numbers

In order to detect when a middlebox has modified the initial sequence numbers (ISNs) in the three-way handshake, each host must choose an ISN such that the Sequence Number Extension is the bitwise inverse of the Legacy Sequence Number. In C pseudo-code:

```
sequence_number_extension = ~(legacy_sequence_number);
```

This property is only a restriction on a choice of ISN. Subsequent to the selection of an ISN, 64-bit sequence numbers behave as normal 64-bit numbers.

2.2.2. Negotiation of the 64-bit Sequence Number Option

When a host ("the client") desires to use 64-bit sequence numbers for a TCP connection it is initiating, it includes the 64-bit Sequence Number Option in the initial segment. It places the least-significant 32 bits of the initial sequence number (ISN) in the Sequence Number field of the TCP header. It places the most-significant 32 bits of the ISN in the Sequence Number Extension field of the 64-bit Sequence Number Option.

When a host ("the server") receives a request to initiate a TCP connection (that is, a segment with the SYN flag set and the ACK flag not set) and the segment contains a valid 64-bit Sequence Number Option, the server MAY choose to use 64-bit sequence numbers for that TCP connection. If the server chooses to use 64-bit sequence numbers for that connection, the server includes the 64-bit sequence number option in its reply (that is, a segment with both the SYN and ACK flags set). The server MUST NOT include a 64-bit Sequence Number Option unless the client included the 64-bit Sequence Number Option in its request to initiate a TCP connection.

When the client receives the initial reply (that is, a segment with both the SYN and ACK flags set), it checks for a valid 64-bit Sequence Number Option. If it finds a valid 64-bit Sequence Number Option, it MUST include the 64-bit Sequence Number Option on all subsequent segments it sends for this connection.

When the server receives an acknowledgement to its initial segment, it chcks for a valid 64-bit Sequence Number Option. If it finds a valid 64-bit Sequence Number Option, it MUST include the 64-bit Sequence Number Option on all subsequent segments it sends for this connection.

For purposes of this section, a 64-bit Sequence Number Option is considered "valid" if (and only if):

- If the segment's SYN flag is set, the Sequence Number Extension must be the bitwise inverse of the Legacy Sequence Number.

- If the ACK flag is set, the full 64-bit acknowledgment number exactly matches the expected value.

A host is said to have negotiated to use 64-bit sequence numbers is it has sent a 64-bit Sequence Number Option in the first segment it sent to the remote host and the first reply it received from the remote host contained a valid 64-bit Sequence Number Option.

If a host successfully negotiates the use of 64-bit sequence numbers, the host proceeds using 64-bit sequence numbers for the remainder of the session. If a host fails to successfully negotiate the use of 64-bit sequence numbers, the host uses backwards compatibility mode (see Section 2.2.4).

2.2.3. Detecting Middle Boxes

If a middle box is present which is modifying sequence numbers or proxying TCP connections, and that middle box does not support 64-bit sequence numbers, it is probable that either the ISN will not follow the rule specified in Section 2.2.1 or the Acknowledgment Number will not match the expected values. (The probability that these will exactly match accidentally is approximately 1 in 2**32. And, it is hard to conceive of a reasonable scenario where the 32-bit sequence numbers will exactly match, the first three segments will all also contain valid 64-bit Sequence Number Options, and yet the two sides will be unable to communicate using 64-bit sequence numbers.) That is why both sides MUST follow the validation rules specified in Section 2.2.2 for the first first three packets in the session (the so-called "three-way handshake"). And, this is also why both sides MUST fallback to using 32-bit sequence numbers if an invalid 64-bit Sequence Number Option is detected in one of the first three frames.

2.2.4. Backwards Compatibility Mode

If a host finds a missing or invalid 64-bit Sequence Number Option in one of the first three segments of a connection (the so-called "three-way handshake"), it MUST process the segment using 32-bit sequence numbers. Specifically, it ignores any 64-bit Sequence Number Option and only pays attention to the 32-bit Sequence Number and Acknowledgement Number fields found in the standard TCP header. Additionally, the host only considers the Legacy Sequence Number portion of the 64-bit sequence number and/or acknowledgement number it stored for the session. If the host finds that the segment is still not valid (e.g. the Acknowledgment Number does not match the expected value), it ignores the segment. (NOTE: This specifically means that the segment does NOT determine whether the host has successfully negotiated, or failed to negotiate, the use of 64-bit sequence numbers on the session.)

However, if the host finds that the segment is valid when processed using 32-bit sequence numbers, the 64-bit sequence number negotiation has failed and the host MUST proceed for the rest of the session using only 32-bit sequence numbers. In this case, it MUST NOT send the 64-bit Sequence Number Option on any further segments for that connection.

If a host has NOT successfully negotiated to use 64-bit sequence numbers for a particular connection and it receives a 64-bit Sequence Number Option in a TCP segment for that connection, it MUST treat the segment as if it contained an out-of-window sequence number.

If a host has successfully negotiated to use 64-bit sequence numbers for a particular connection and it receives a segment without a 64-bit Sequence Number Option, it MUST treat the segment as if it contained an out-of-window sequence number.

3. Changes to Other Features

Over time, other features have built upon the base TCP protocol specification. Many, if not all, of these features have assumed the existence of 32-bit sequence numbers. This document updates some of the features. It also provides a general rule for the operation of other features.

3.1. Window Size

[RFC7323] specifies the Window Scale Option. It specifies a maximum window shift of 14. This document updates [RFC7323] by specifying that the maximum window shift is 46 if the hosts successfully negotiate using 64-bit sequence numbers for a connection. If the 64-bit sequence number negotiation fails, both hosts must enforce the maximum window shift of 14 specified by [RFC7323].

3.2. SACK Blocks

[RFC2018] defines a way to acknowledge receipt of out-of-order segments. [RFC2018] specifies that the segments are defined by 32-bit sequence numbers. This document updates the way SACK blocks defined in [RFC2018] are interpreted when used on 64-bit environments. It also defines a new option used to hold 64-bit SACK blocks.

3.2.1. 32-bit SACK Blocks

When a host has successfully negotiated the use of 64-bit sequence numbers on a session and has also negotiated the use of SACK as described in [RFC2018], the host may append a TCP SACK option as defined in [RFC2018]. When these options are used, the sequence numbers in the option are interpreted as follows: the Acknowledgment Number Extension from the 64-bit Sequence Number Option is used as the most-significant 32 bits of the 64-bit sequence numbers, while the sequence numbers from the TCP SACK option are used as the least-significant 32 bits of the 64-bit sequence numbers. Otherwise, the operation of the TCP SACK option remains unchanged.

3.2.2. 64-bit SACK Blocks

When a host has successfully negotiated the use of 64-bit sequence numbers on a session and has also negotiated the use of SACK as described in [RFC2018], the host may append a 64-bit SACK Option.

The 64-bit SACK Option will use TCP Option Kind TBD3. Its general form is shown in Figure 5.

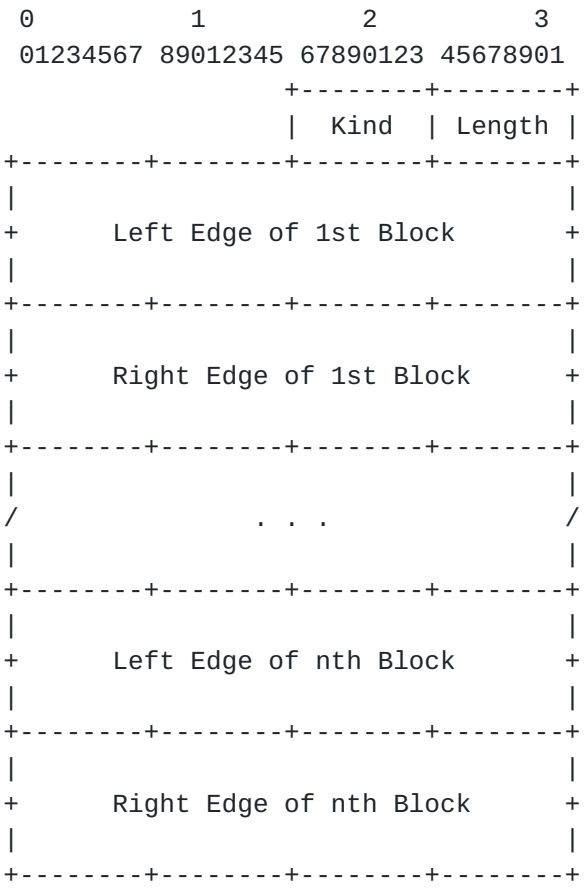

Figure 5: The 64-bit SACK Option

Prior to standardization action, implementations should use the mechanism described in [RFC6994] to encode the option. IANA has reserved experiment ID (ExID) TBD4 for the option described in this document. This option format is shown in Figure 6.

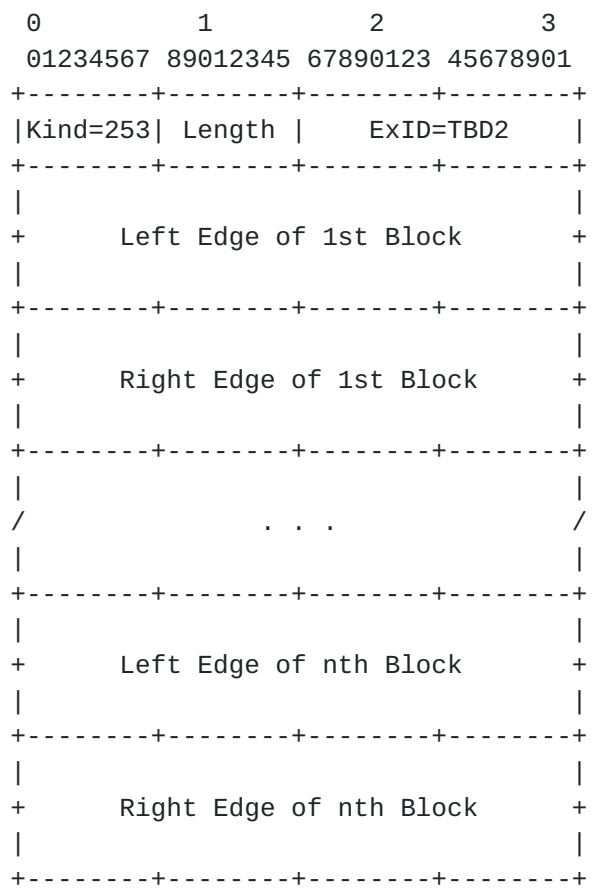

Figure 6: The 64-bit SACK Option with ExID

The meaning of the option fields, and the operation of the option, is unchanged from the TCP SACK option described in [RFC2018], except that the sequence numbers are 64-bit values in network byte order.

3.3. TCP Authentication Option

[RFC5925] defines the TCP Authentication Option. The TCP Authentication Option uses sequence numbers in two places: the Key Derivation Function (KDF) context and the data input to the Message Authentication Code (MAC) algorithm.

For purposes of the KDF context, this document updates [RFC5925] to specify that the Source ISN and Dest. ISN fields (shown in Figure 7 of [RFC5925]) are defined to be the least-significant 32 bits of the initial sequence numbers.

For purposes of the input to the Message Authentication Code (MAC) algorithm, this document updates [RFC5925] to specify that the Sequence Number Extension field is the Sequence Number Extension field from the 64-bit Sequence Number Option, if the option is

present, in any packet that does not carry the SYN flag. Otherwise, the Sequence Number Extension field is calculated as specified in [RFC5925].

Note that the Sequence Number Extension field will always be formulated as specified in [RFC5925] for the first two packets of the so-called "three-way handshake". This ensures that hosts will be able to correctly calculate MACs whether or not they support 64-bit sequence numbers.

3.4. Other Features

Anywhere that another RFC specifies the use of sequence numbers without specifying the way 64-bit sequence numbers should be handled, the RFC shall be interpreted as using the least-significant 32 bits of the sequence number.

4. Implementation Considerations

During the 64-bit sequence number negotiation, it is important for security purposes (as described in Section 7.3) that the server check the third packet of the "three-way handshake" when determining whether the connection has negotiated to use 64-bit sequence numbers. If another in-sequence packet is received prior to the third packet of the "three-way handshake", it must either be discarded or queued for processing after the third packet of the "three-way handshake" is received.

It may be useful to provide a way for applications to know whether a given connection uses 32-bit or 64-bit sequence numbers. It may also be useful to provide a way for applications to force the use of 32-bit or 64-bit sequence numbers.

It will be essential to properly handle 32-bit and 64-bit sequence numbers concurrently for different connections. This will require providing two sets of arithmetic and comparison functions. For various reasons, it probably makes sense to store the data as a union of a single 64-bit value and a two-member array of 32-bit values.

Due to the limited option space, it may be impossible to deploy this feature concurrently with some other features on a given connection. This limitation may change if the option space is expanded by a future standardization change. However, implementers should pay attention to the possible combinations of options and order them in such a way to fit the maximum number of options in a single segment. Further, implementations will need to prioritize which features actually appear in the option space if they will not all fit.

Careful consideration will need to be paid to various offload technologies, such as TCP segmentation offload (TSO) or large receive offload (LRO). If the network interface card (NIC) drivers or hardware do not support 64-bit sequence numbers, the endpoint MUST NOT try to use 64-bit sequence numbers. Otherwise, sessions may not work correctly in practice, even if they appear to work correctly in small-scale tests.

Implementations must ensure that the least-significant 32 bits of 64-bit initial Sequence Numbers (ISNs) must serve as sufficiently random 32-bit ISNs. (See Section 7.4.)

5. Acknowledgements

Jana Iyengar, Randall Stewart, and Michael Tuexen provided valuable feedback on this document. Michael Tuexen suggested the mechanism that currently appears in Section 2.2.1.

6. IANA Considerations

IANA has assigned an option code value of TBD1 to the 64-bit Sequence Number Option (defined in Section 2.1) and an option code value of TBD3 to the 64-bit SACK Option (defined in Section 3.2.2) from the TCP Option Kind Numbers space defined in Section 9.3 of RFC 2780 [RFC2780].

[Note to editor: I think this paragraph and the following table can be removed.]The requested options are summarized below:

Value	Description	Reference
TBD1	64-bit Sequence Number	[RFCXXXX]
TBD3	64-bit SACK	[RFCXXXX]

IANA has assigned an identifer value of TBD2 to the 64-bit Sequence Number experiment and an identifier value of TBD4 to the 64-bit SACK experiment from the TCP Experimental Option Experiment Identifiers space defined in Section 8 of RFC 6994 [RFC6994] [NOTE: If this is standardized with an option number, the experimental IDs should be deprecated, which will require change to this text.]

[Note to editor: I think this paragraph and the following table can be removed.]The assigned ExIDs are summarized below:

Value	Description	Reference
TBD2	64-bit Sequence Number	[draft-looney-tcpm-64-bit-seqnos-00]
TBD4	64-bit SACK	[draft-looney-tcpm-64-bit-seqnos-00]

7. Security Considerations

The security properties of TCP are largely unchanged (at least in a negative way) by 64-bit sequence numbers. However, a few things are worth discussing.

7.1. Attacks Due to Sequence-Number Guessing

With 32-bit sequence numbers and the maximum window shift, an attacker has approximately a 25% chance of accurately guessing an in-window Sequence Number. If a host checks for both the acceptability of Sequence Numbers and Acknowledgment Numbers prior to acting on a segment, in the worst-case scenario (where the full window size is in flight, allowing for a full window size worth of acceptable Acknowledgment Numbers), this allows a 6.25% chance of accurately guessing a combination of in-window Sequence Number and acceptable Acknowledgment Number.

Because this document specifies a maximum window shift that is 32 bits larger than the maximum window shift used for 32-bit sequence numbers, these security properties are essentially unchanged with 64-bit sequence numbers. (The major change is that an out-of-band attacker may not be able to guess whether a connection uses 64-bit sequence numbers. This may require that they try both 32-bit and 64-bit sequence number semantics, decreasing the chance that they would accurately guess appropriate sequence numbers.)

However, if you compare the use of 32-bit and 64-bit sequence numbers with the same amount of outstanding traffic and the same window size, the chance of guessing acceptable sequence numbers is much smaller with 64-bit sequence numbers than 32-bit sequence numbers.

7.2. Downgrade Attacks

A man-in-the-middle (for example, a middlebox or proxy) can conduct a downgrade attack. This is actually a feature, as it allows two endpoints that understand 64-bit sequence numbers to communicate through a middlebox or proxy that does not understand 64-bit sequence numbers. However, it is important that operators be cognizant of the differing performance and security properties of 32-bit and 64-bit

sequence numbers. It may be appropriate to provide a mechanism for applications to require the use of 64-bit sequence numbers (and reset a session that cannot be established with 64-bit sequence numbers).

7.3. Denial-of-Service Attacks

This mechanism introduces one additional denial-of-service attack possibility. Assume a session where both sides have sent and received valid 64-bit Sequence Number Options in the SYN segments. If an attacker correctly guesses the appropriate Sequence Number and Acknowledgment Number to use in the third packet of the so-called "three-way handshake" and they can inject a packet with the correct Sequence Number and Acknowledgment Number without a 64-bit Sequence Number Option and ensure the server receives the spoofed packet prior to the valid packet, this will prevent communication between the two hosts. The server will use 32-bit sequence numbers for the session, while the client will use 64-bit sequence numbers for the session. However, the requirement that the server must verify the actual third packet of the "three-way handshake" (and not merely some in-window segment) requires that the attacker EXACTLY guess both the 32-bit Legacy Sequence Number and the 32-bit Legacy Acknowledgment Number. With completely random sequence numbers, the chance of doing this is 1 in 2**64.

7.4. 32-bit Sequence Numbers

Because a session that is started with 64-bit sequence numbers may fallback to using 32-bit sequence numbers, implementations MUST choose ISNs such that the least-significant 32 bits of the ISN must be at least as random as the 32-bit ISNs that the system uses for connections that only support 32-bit sequence numbers.

Further, the mechanism chosen to detect middleboxes results in only 2**32 possible 64-bit ISNs. This provides the same level of security provided with 32-bit sequence numbers.

8. References

8.1. Normative References

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <http://www.rfc-editor.org/info/rfc793>.

[RFC2018] Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, "TCP Selective Acknowledgment Options", RFC 2018, DOI 10.17487/RFC2018, October 1996, <http://www.rfc-editor.org/info/rfc2018>.

[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>.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, DOI 10.17487/RFC5925, June 2010, <http://www.rfc-editor.org/info/rfc5925>.

[RFC6994] Touch, J., "Shared Use of Experimental TCP Options", RFC 6994, DOI 10.17487/RFC6994, August 2013, <http://www.rfc-editor.org/info/rfc6994>.

[RFC7323] Borman, D., Braden, B., Jacobson, V., and R. Scheffenegger, Ed., "TCP Extensions for High Performance", RFC 7323, DOI 10.17487/RFC7323, September 2014, <http://www.rfc-editor.org/info/rfc7323>.

8.2. Informative References

[RFC2780] Bradner, S. and V. Paxson, "IANA Allocation Guidelines For Values In the Internet Protocol and Related Headers", BCP 37, RFC 2780, DOI 10.17487/RFC2780, March 2000, <http://www.rfc-editor.org/info/rfc2780>.

Appendix A. Design Choices

This section attempts to document the reasoning behind some of the design choices.

A.1. Detecting Middle Boxes

An earlier version of this draft specified a different mechanism for detecting middlebox changes: a checksum of the 64-bit Sequence Number and Acknowledgment Number. This had the benefit of allowing the full 64-bit sequence number to be random. However, it had the negative effects of requiring an additional two bytes of option space and requiring additional processing on input and output. Michael Tuexen suggested the mechanism that currently appears in Section 2.2.1.

The mechanism that currently appears in Section 2.2.3 may still fail to detect a middlebox in one case. If there is a middlebox (such as a "transparent proxy") that passes TCP segments unchanged between the client and server, rewriting only IP addresses, this mechanism will not detect such a middlebox. However, it is not really necessary to detect such a middlebox: if the middlebox literally leaves the TCP portion of the packet unchanged, it should be perfectly acceptable to use 64-bit sequence numbers.

A.2. SACK Blocks

An earlier version of this draft reused the existing TCP SACK option and specified that the option should contain sequence numbers of the same length as the sequence numbers in use for the connection. However, this was suboptimal for two reasons. First, a middle box might misinterpret the meaning of the 64-bit sequence numbers. Second, it always required the use of 64-bit values. The current mechanism means that the existing TCP SACK option will always contain 32-bit values. This mechanism also allows the use of 32-bit values instead of full 64-bit values in some cases. However, this may still suffer from being too complex.

Author's Address

Jonathan Looney
Netflix
100 Winchester Circle
Los Gatos, CA 95032
USA

Email: jtl.ietf@gmail.com