TCPM WG
Internet Draft
Intended status: Experimental
Expires: March 2015

J. Touch
T. Faber
USC/ISI

September 29, 2014

TCP SYN Extended Option Space Using an Out-of-Band Segment
draft-touch-tcpm-tcp-syn-ext-opt-01.txt

Status of this Memo

Copyright Notice

Abstract

This document describes an experimental method to extend the option space for connection parameters within the initial TCP SYN segment, at the start of a TCP connection. This method effectively extends the option space of an initial SYN by using an additional coupled segment that is sent 'out-of-band'. It complements the proposed Extended Data Offset (EDO) option that is applicable only after the initial segment.

Table of Contents

1. Introduction

This document describes a method to extend the option space available in the initial SYN segment of a TCP connection (e.g., SYN

set and ACK not set) [RFC793]. This extension is required to support some combinations of TCP options, notably large ones such as TCP AO [RFC5925], Multipath TCP [RFC6824], and TCP Fast Open [Ch14] with other options already typically used in most TCP connections. This document specifies this TCP SYN extended option space (SYN-EOS) option, and is independent of (and thus compatible with) IPv4 and IPv6. SYN-EOS complements the proposed TCP Extended Data Offset (EDO) option, which increases the space available for options in all segments except the initial SYN [To14].

2. Conventions used in this document

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].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

In this document, the characters ">>" preceding an indented line(s) indicates a compliance requirement statement using the key words listed above. This convention aids reviewers in quickly identifying or finding the explicit compliance requirements of this RFC.

3. Experiment Goals

TCP is critical to the robust functioning of the Internet, therefore any proposed modifications to TCP need to be thoroughly tested. The present specification describes an experimental protocol that initiates a connection using two coupled segments instead of the traditional single one. The intention is to specify the protocol sufficiently so that more than one implementation can be built in order to test its function, robustness, and interoperability with itself, with other variants of TCP and with common network equipment, whether standardized or not.

The following describe the criteria that define success for this

experiment and its expected duration.

Success criteria: The experimental protocol will be considered successful if, in the consensus opinion of the IETF, it functions correctly in a sufficiently wide scope to be useful and it does no harm, which implies that it ought to introduce minimal additional delay or load to either updated or existing implementations and it introduces no new security vulnerabilities. It is also required not

to be unduly difficult or complex to implement correctly, so that it is not likely to lead to additional bugs or vulnerabilities.

Duration: To be credible, the experiment will need to last at least 12 months from publication of the present specification. At that time, a report on the experiment will be written up. If successful, it would then be appropriate to work on a standards track specification.

4. Using Multiple Segments to Establish a Connection

The basis of SYN-EOS is the use of multiple TCP segments to initiate a TCP connection. It is also possible to extend initial SYN option space using context established from prior connections or using separate TCP connections (e.g., using the FTP control channel), this document focuses on a mechanisms that applies to any connection (including the first between two hosts) and do not require prior or established concurrent TCP connections.

There are four examples of such approaches:

- Send a primary SYN and an extension SYN (LOIC [Yo11])
- Send a primary SYN and extension non-SYN data (LO/SLO [Ed08])
- Send two separate SYNs: a legacy SYN and an upgraded SYN on separate port pairs (dual-stack, named "Sister-SYN" [Br14])
- Send a primary SYN and an extension out-of-band segment (OOB, this document)

All four approaches extend the space available in the initial SYN by sending an additional segment during the first phase of the three-

way handshake. The Long Options by Invalid Checksum (LOIC) approach differentiates the two SYNs by using an invalid TCP checksum in the extension SYN [Yo11], which thus cannot traverse NAT/NAPT devices [RFC3234].

The LO/SLO approach extends the three-way handshake into a five-way handshake to include extra options during the third segment, so the traditional SYN/ACK does not complete the active connection [Ed08]. In current implementations, the client TCP state machine transitions to the ESTABLISHED state upon receipt of the SYN/ACK (including transmission of the resulting ACK). In SLO, additional options sent during the third segment are treated as part of the initial SYN and the fourth segment with responses to these options is treated as part of the conventional SYN/ACK. As with conventional TCP, data can

be sent during this handshake as part of any segment, but this data needs to wait for the entire handshake to complete before being forwarded to the application to ensure that all options have been negotiated successfully. This adds an additional round trip of latency which is undesirable in many cases. Connection-splitting middleboxes that merge these segments might also cause long options to be interpreted as data.

The Sister-SYN approach is a dual-stack mechanism. Both legacy servers and upgraded servers process both SYNs; clients terminate the appropriate pending connection based on whether the option is acknowledged for each connection.

The remainder of this document presents the SYN-EOS approach, which overcomes these limitations using an out-of-band segment to extend the option space of the SYN.

5. The TCP SYN-EOS Option

The SYN-EOS approach uses a primary conventional SYN and an additional out-of-band data segment, the latter being a non-SYN packet with the ACK flag not set. Additional options are placed in payload of an out-of-band (OOB) segment, i.e., a segment whose ACK bit is cleared but is not a SYN (i.e., both SYN and ACK are zero). This offers the following advantages:

1. It provide expansion space for options on a SYN, limited only by the default maximum segment size (535 payload bytes for IPv4);

2. It reduces the chances that middleboxes will alter the extra options, given there is a higher bar to altering the payload than header fields.

3. It allows for future structured ways to hide extra options from middleboxes and/or to protect them from being altered.

A client initiating a TCP connection (i.e., issuing an active open) uses the SYN-EOS option flag to indicate the presence of the extended option space (Figure 1). This follows the TCP option format, where Kind is SYN-EOS-OPT and Length is 2.

Figure 1 TCP SYN-EOS OOB option

An upgraded client supporting this feature uses this option only when the space needed for options in the initial SYN exceeds that of legacy TCP. When needed, the client sends the SYN-EOS option in the initial SYN, together with whatever other options are intended for connections to legacy servers (i.e., passive listeners). A legacy server would respond with a SYN/ACK without the SYN-EOS option, while also confirming other supported options, and the connection would proceed without the SYN-EOS extension.

The upgraded client that sends the initial SYN using this option also sends an out-of-band (OOB) data segment with the same option to the same source and destination addresses and ports as the initial SYN. An OOB data segment is herein defined as a TCP segment in which neither SYN nor ACK flag is set. In particular, this looks like a conventional data segment with the ACK field cleared. Current TCP requirements allow the ACK field to be cleared for only the initial SYN, so this segment looks like a data segment that has been transmitted 'out-of-band', before a connection has been established. The entire payload of the segment is used for additional options.

Upgraded servers that receive the TCP SYN with the SYN-EOS option wait for the corresponding OOB segment and treat the entire set of options in both segments as if they arrived with the initial SYN.

Once both have arrived, the server first processes TCP options placed before each SYN-EOS option, applying them solely to their own individual segment. Then the server marshals together all the TCP options placed after each SYN-EOS option. It applies them to the initial SYN only, as if they had all been concatenated after the SYN-EOS OOB option.

>> The server MUST process the options placed after each SYN-EOS option in the following order:

1. Those in the option space of the initial SYN

2. Those in the option space of the OOB segment

3. Those in the payload space of the OOB segment

The upgraded server proceeds with the remainder of the connection as if the SYN-EOS OOB option were a also EDO request option [To14] in the SYN.

>> The SYN/ACK MUST include the SYN-EOS option to confirm the server's support for both the SUN-EOS and EDO capabilities and to confirm receipt of both the SYN and OOB segment. The server MAY also

extend the options space of the SYN/ACK using the EDO option if needed.

>> Any host that supports SYN-EOS MUST also thus support EDO.

>> The OOB segment MUST use the same sequence number as the initial SYN.

>> The client MUST NOT send multiple different OOB segments. If the server receives more than one OOB segment for the same connection it MUST solely use the first.

5.1. Reliable Delivery of Lone Initial Segments

The server acknowledges the initial segment and the OOB segment together by using a SYN/ACK that carries the appropriate SYN-EOS option. The following subsections describe how the server acknowledges initial segments after a certain time if only one has

arrived.

5.2. Reliable Delivery of a Lone SYN with SYN-EOS

If an upgraded server has received only a SYN with the SYN-EOS option but no corresponding OOB segment, after a certain time it MUST proceed with the connection as if the SYN had been received without the SYN-EOS option. I.e. it processes all other TCP options and responds with a SYN/ACK without the SYN-EOS option.

A client will not be able to tell whether this SYN/ACK is from a legacy server or an upgraded server. How the client proceeds on receipt of such a SYN/ACK depends on whether it wishes to retry sending the TCP options in the OOB segment or to proceed without them (e.g. for latency reasons):

>> If the client chooses to proceed without the OOB segment, it MUST proceed as if the SYN-EOS option had never been used, by sending an ACK to complete the three-way handshake.

>> If the client chooses to retry, it MUST retransmit the OOB segment with the same sequence number as the ISN of the SYN, so it is still out-of-band. However, this time it sets the ACK flag and it sets the acknowledgement number to one greater than the sequence number of the SYN/ACK. This effectively acknowledges receipt of the SYN/ACK, but requests a fuller SYN/ACK that also covers the OOB segment. At this stage a client that has chosen to retry the OOB segment MUST NOT send the ACK that would normally complete the three-way handshake.

>> If an upgraded server receives such a retransmitted OOB segment, it MUST process the additional TCP options as if they were placed after those in the initial SYN. Then it MUST send a SYN/ACK containing the SYN-EOS option, as if it had not sent the earlier SYN/ACK .

On receipt of this SYN/ACK, the client sends an ACK to complete the handshake.

>> If an upgraded server receives an ACK to complete the handshake, then later receives an OOB segment, it MUST discard the late OOB segment.

>> If a server, whether upgraded or not, receives only an OOB segment and no corresponding SYN, it MUST discard it and it MUST NOT ever respond (see Security Considerations).

5.3. Interaction with EDO

Successful negotiation of either SYN-EOS option has the same effect as EDO. Successful SYN-EOS negotiation enables EDO for the remainder of the connection.

>> After successful SYN-EOS negotiation, segments after the initial SYN MAY use the EDO option.

Note that a failure to negotiate SYN-EOS has also fails to automatically negotiate EDO for endpoints that support EDO but not SYN-EOS. As a consequence:

>> If EDO is desired when SYN-EOS fails, the initial SYN options MUST include a separate EDO request option.

If SYN-EOS is sent in the initial SYN and confirmed in the SYN/ACK, EDO is available for the remainder of the connection. Segments that need to extend their option space would then include EDO.

>> If SYN-EOS and EDO are sent in the initial SYN and received by SYN-EOS capable server, the server MUST include SYN-EOS in the SYN/ACK, and MAY also include EDO also needed to provide additional option space.

>> If the server agrees to EDO but cannot support SYN-EOS, the SYN/ACK MUST include EDO as per [To14] to confirm the capability.

6. Issues

The following issues are known.

6.1. General Issues

Caching is required because it is unlikely that both segments involved in initiating a SYN-EOS connection will arrive at the same

time:

>> Servers supporting SYN-EOS SHOULD cache received initial SYNs with the SYN-EOS option. Servers MAY decline to cache received initial SYNs if they are under memory constraints.

>> Servers supporting SYN-EOS SHOULD cache received SYN-C segments with the SYN-EOS option. Servers MAY cache received OOB segments but MUST NOT examine or process them further in any way until their corresponding SYN segment arrives.

Similarly, clients need to be able to retransmit supplements to ensure their delivery:

>> Clients MUST retransmit the supplemental segment any time they retransmit the initial SYN segment.

Should this be a new option or just a variant of EDO, and if so, how would it change EDO?

SYN Cookies: An updated server can achieve the same outcome as SYN cookies by putting all the necessary connection state in TCP options in the SYN/ACK (using EDO if extra space is needed). It would then discard its own copy of this state, which it could recover from the TCP options in the final ACK of the 3WHS sent by the client. New TCP options complementary to SYN-EOS might need to be defined to achieve this for some types of TCP option (TBA). A legacy server will not understand the SYN-EOS option whether it uses SYN cookies or not, so it will provide the same legacy service whether or not it uses SYN cookies.

Useful to send SYN, wait shortly, then send OOB

OOB traversal concerns

6.2. Option processing order

TCP options before the EOS-SYN on initial SYN segments are necessarily processed individually when each segment arrives. When

both segments of an EOS-SYN connection establishment arrive, the remaining options are processed in the following sequence:

1. Initial SYN options

2. Supplement options

3. Supplement payload

*** NOTE TO THE WG:

There are two other constraints that might be applied to the supplement options:

>> I. Supplement options MUST exactly match initial SYN options.

>> II. Supplement options MUST contain only the SYN-EOS option.

If either of these is chosen, the supplement options are NOT processed again (i.e., they are discarded).

The former constraint helps the supplement segment share the same fate as the initial SYN. The latter recognizes that the supplement option space is not needed given the supplement payload, because the option space is created from the payload space anyway.

*** END NOTE

6.3. Middlebox Transit Issues

NB: this variant will require an additional 1-byte field on the SYN-EOS option for the EOO field.

Traversal of middleboxes that ensure the payload matches the destination port number. It would be possible to include the facility for SYN-EOS to include an Extra Option Offset (EOO) field. A client setting EOO to a non-zero value would offset the start of the additional TCP options by this number of 4-byte words from the start of the payload.

>> An upgraded SYN-EOS server MUST start reading the additional TCP options from a point within the payload that is offset by this number of 4-byte words from the start of the payload. An upgraded SYN-EOS server MUST ignore all data in the payload up to this point.

The client would then be free to include fake data at the start of the payload consistent with what a middlebox might expect for the

destination port in use. The data to use would be application and implementation dependent, and is not determined in the present specification.

Interaction with TCP Fast Open

6.4. Interaction with Other TCP Options

6.5. TCP Fast Open

TBD.

Notes: SYN-EOS appears to be safe with TFO. Dual-SYN variants appear to have potential problems with both upgraded and legacy servers. With upgraded servers, receipt of a legacy SYN with the SYN extension option flag present might require delayed response. With legacy servers, it may be impossible to safely use TFO with the extended SYN.

6.5.1. TCP Authentication Option and TCP MD5

TBD.

Notes: Likely to be similar to TCP EDO, i.e., requiring authentication processing before extension processing.

7. TCP SYN-EOS Interaction with TCP

The following subsections describe how SYN-EOS interacts with the TCP specification [RFC793].

7.1. TCP User Interface

TBD.

7.2. TCP States and Transitions

TBD.

7.3. TCP Segment Processing

TBD.

7.4. Impact on TCP Header Size

TBD.

8. Error Conditions

8.1. Connectionless Resets

TBD.

8.2. ICMP Handling

TBD [RFC792].

9. Security Considerations

>> By default, a SYN-EOS server must not cache an OOB segment and MUST NOT respond to an OOB segment if it arrives before the corresponding SYN segment, because many legacy firewalls will allow OOB segments into private networks. Caching of OOB segments MAY be enabled explicitly on public servers.

More TBD.

10. IANA Considerations

TBD.

This section is to be removed prior to publication as an RFC.

11. References

11.1. Normative References

[RFC793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[To14] Touch, J., W. Eddy, "TCP Extended Data Offset Option", draft-ietf-tcpm-tcp-edo-00 (work in progress), September 2014.

11.2. Informative References

[Br14] Briscoe, B., "Extended TCP Option Space in the Payload of

an Alternative SYN", draft-briscoe-tcpm-syn-op-sis-02 (work in progress), September 2014.

[Ch14] Cheng, Y., Chu, J., and A. Jain, "TCP Fast Open", draft-ietf-tcpm-fastopen-10, September 2014.

[Ed08] Eddy, W. and A. Langley, "Extending the Space Available for TCP Options", draft-eddy-tcp-loo-04 (work in progress), July 2008.

[RFC792] Postel, J., "Internet Control Message Protocol", RFC 792.

[RFC3234] Carpenter, B. and S. Brim, "Middleboxes: Taxonomy and Issues", RFC 3234, February 2002.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, June 2010.

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, January 2013.

[Yo11] Yourtchenko, A., "Introducing TCP Long Options by Invalid Checksum", draft-yourtchenko-tcp-loic-00 (work in progress), April 2011.

12. Acknowledgments

The authors would like to thank the IETF TCPM WG for their feedback.

The use of multiple segments to extend the option space of a SYN was initially proposed by Bob Briscoe. His initial proposal used complementary SYNs in an earlier version of this document, which evolved into mutually-exclusive "Sister-SYNs" in [Br14].

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Joe Touch
USC/ISI

4676 Admiralty Way
Marina del Rey, CA 90292-6695 USA

Phone: +1 (310) 448-9151
Email: touch@isi.edu
URI: http://www.isi.edu/touch

Ted Faber
USC/ISI
4676 Admiralty Way
Marina del Rey, CA 90292-6695 USA

Phone: +1 (310) 448-9190
Email: faber@isi.edu