Extending Option Space Discussion

Overview and its requirements

Tcpm chairs

Introduction

- 40 bytes option space is becoming serious problems for TCP
  - Without extending option space, TCP cannot enable some features simultaneously
  - Option space limitation affects the design of some features (e.g. MPTCP, TCPCrypt)
- Current WG item for option space solution
  - draft-ietf-tcpm-tcp-edo
    - Extend option space for non-SYN packets

What about extending option space in SYN?

Non-SYN option space extension

- Current proposal: EDO (draft-ietf-tcpm-tcp-edo)
  - Use a part of payload as option space
    - Override DO (Data Offset) field
  - Use new option type for signaling
  - Negotiate the feature during SYN exchanges
  - No specific mechanism for middlebox issue
- Enable feature only after receiving confirmation signals from both sides

Difficulty in SYN option space extension

- Need to use option space extension feature during SYN exchanges
  - Packet with extension will be different from standard SYN packets
  - How to send non-standard packets while negotiating features?
    - Can we send these packets without negotiation?
    - Can we negotiate the feature without extra delay?

Middlebox Issues

- Some middleboxes affects non-standard TCP packets (drop or strip options or modify flags, etc)
  - TCP segments with unknown options, special flags may look non-standard
- Some middleboxes modify segments
  - TCP segments might be coalesced or split or updated
  - If we override DO field in TCP segments, this will be serious problems!
- How much these issues can affect option space extension design?

Alternative Option Extension Proposals

- draft-touch-tcpm-tcp-syn-ext-opt
  - Use special packets for the feature
    - Send extended packets without negotiation
- draft-briscoe-tcpm-inner-space
  - Use two connections for the feature
    - Send extended packets without negotiation + encoding mechanisms for middlebox protection
- draft-borman-tcpm-tcp4way
  - Introduce 4way handshake mechanism
    - Modify negotiation mechanism to allow extended packets
- draft-leslie-tcpm-checksum-option
  - Check if middleboxes modify the packets
    - Supplemental feature for option space extension. Can be combined to other solutions

draft-touch-tcpm-tcp-syn-ext-opt

- Client sends SYN with SYN-EOS option
  - Indicate to send OOB (Out-Of-Band) packet
- Client sends OOB packet after sending SYN
  - OOB packet: packet with both SYN and ACK are not set
  - Payload in OOB packet can be used to store TCP options as well as TCP header
- Server processes all TCP options in the following places
  - TCP header in SYN, TCP header in OOB, TCP payload in OOB
- Server sends back SYN-EOS option in SYN/ACK for confirmation

draft-briscoe-tcpm-inner-space

- Client sends two SYN packets (same dest port, but different source port)
  - One SYN has normal format, the other has upgraded format to store TCP options in payload
- Client resets one of the connection based on the response from server
  - If server correctly respond to upgraded format, reset normal connection (upgrade)
  - If not, reset upgraded connection (fall back)

draft-borman-tcpm-tcp4way

- Client sends SYN with an indication (TCP flag or option) for 4way handshake
- If server responses back the indication, client sends second SYN to perform 4way handshake
- EDO option can be negotiated in the first SYN exchange
- The second SYN can use EDO option if both sides agree in the first SYN exchange

draft-leslie-tcpm-tcp-checksum-option-00

- TCP Option to checksum various fields
- No handshake (changes nothing about TCP)
- Typical use: checksum over all Option bytes
- Diagnostic only: Remediation not part of spec
  - (details will change: byte-count vs. checksum, etc.)

How to move forward?

- How radically we want to change TCP for option space?
  - What kinds of changes to be allowed in TCP?

- This will be an important update for TCP
  - No need to hurry, but wants many feedbacks!

Criteria for the solution

- What are the requirements for this? How much extent do we need or allow?
  - Robustness against middleboxes
  - Latency
  - Design complexity
  - What others?

Discussion Points

- How to address middlebox issues?
- How we should structure SYN option space extension mechanism?
- What to do with non-SYN (EDO) solution?

Q1: How to address middlebox issues?

- Which issues should be addressed or ignored (How we should be robust against middlebox)
  - Removing new options
  - Modifying flags
  - Dropping non-standard segments
  - Splitting, Coalescing segments
  - Combinations of above
- How to approach the problem?
  - In-bound encoding like inner-space proposal
  - checksum like mechanism to detect middlebox interventions
- Do we need something for this?
  - Experiments? Survey? Publishing docs?

Q2: How we should structure SYN option space extension mechanism?

A. Publish one mechanism as single doc
B. Publish multiple docs as experimental and encourage experiments
C. Publish single doc that describes possible approach
D. Something else
    - Do nothing for now?

Q3: What to do with non-SYN solution?

- Should we merge non-SYN solution (EDO) and other solutions?
  - Current non-SYN draft is simple and straightforward
  - Alternative proposals may use different mechanisms to extend option space used in non-SYN draft

  - Pros
    - Can provide single mechanism for option space extension
  - Cons
    - Will takes more time. More complicated. Should be experimental.