A-PAWS: Alternative Approach for PAWS draft-nishida-tcpm-apaws-01

Yoshifumi Nishida

Background

- RFC1323 (RFC7323) requires putting timestamps in all segments

  Once TSopt has been successfully negotiated,
  TSopt MUST be sent in every non-<RST> segment for the duration of the connection

- Timestamp consumes 10-12 bytes in option space
  - 25-30% available option space cannot be used for other options!

Why We Need TS in Every Segment?

- RTT measurements
  - TS in every segment is not necessary
    - Number of samples per RTT does not affect the effectiveness of RTO
- PAWS
  - TS in every segment is necessary
    - Otherwise, TCP might accept old duplicated segments by mistake
- If we have PAWS-like mechanism without TS, we don’t need TS in every segments!

A-PAWS: An Alternative for PAWS

- Design Principle
  - Do not rely on timestamp
  - Provide the same protection as PAWS does
  - Fallback to PAWS if there is a risk
    - Never be worse than PAWS

A-PAWS's Logic

- Basic rules
  - Senders don't put TS until 4GB (2**32 bytes) has been sent
  - Receivers mustn't drop segments without TS until receive 4GB
  - After 4GB transmission, endpoints fallback to PAWS
- Applicability
  - 99.9% TCP connections don't send more than 4GB
- Overhead
  - Requires both endpoints to count sending/receiving bytes, but shouldn't be a problem

Discussions (1)

- PAWS is not only for sequence wrapping, but also used for protection against packets from previous connections
  - This situation may happen due to rebooting or using SO_REUSEADDR
- Solution
  - Don’t use A-PAWS for a MSL upon starting up
  - Don’t use A-PAWS if SO_REUSEADDR is set

Discussions (2)

- PAWS can be used to enhance protection against spoofed packets
  - Receiver can check TS in addition to 5 tuples
- PAWS logic for protection against spoofed packets
  - Compare TS in the received segment (SEG.TSVal) and latest received TS (TS.Recent)
    - SEG.TSval < TS.Recent ... reject
    - SEG.TSval >= TS.Recent ... accept
- This is probably not useful for attacks in 21st century
  - Using random TS can pass PAWS check easily
  - Attackers usually can send multiple packets

Discussions (3)

- A-PAWS requires a signalling mechanism between sender and receiver, how do we do it?

- 3 possible approach
  - Using new TCP option in SYN segments
    - Easy and straightforward, but it consumes option spaces in SYN
  - Using Timestamp values in SYN segments
    - Proposed in draft-scheffenegger-tcpm-timestamp-negotiation
      - Not standardized yet
  - Using new TCP option in Non-SYN segments
    - Sounds better approach, but is it possible?

Signaling With non-SYN Segments

- Design Principal
  - Don’t invent another 3WHS in non-SYN segments
    - Too much complexity!
  - Simple and easy mechanism to be implemented
    - Exchange only 2 segments for feature negotiation
      - Can utilize any DATA and ACK segments exchange

Loose Synchronization in A-PAWS

- A-PAWS doesn't require tight synchronization between senders and receivers
  - A-PAWS receiver can work with PAWS sender

Case #	Sender	Receiver	
1	PAWS	PAWS	◯
2	PAWS	A-PAWS	◯
3	A-PAWS	PAWS	✕
4	A-PAWS	A-PAWS	◯

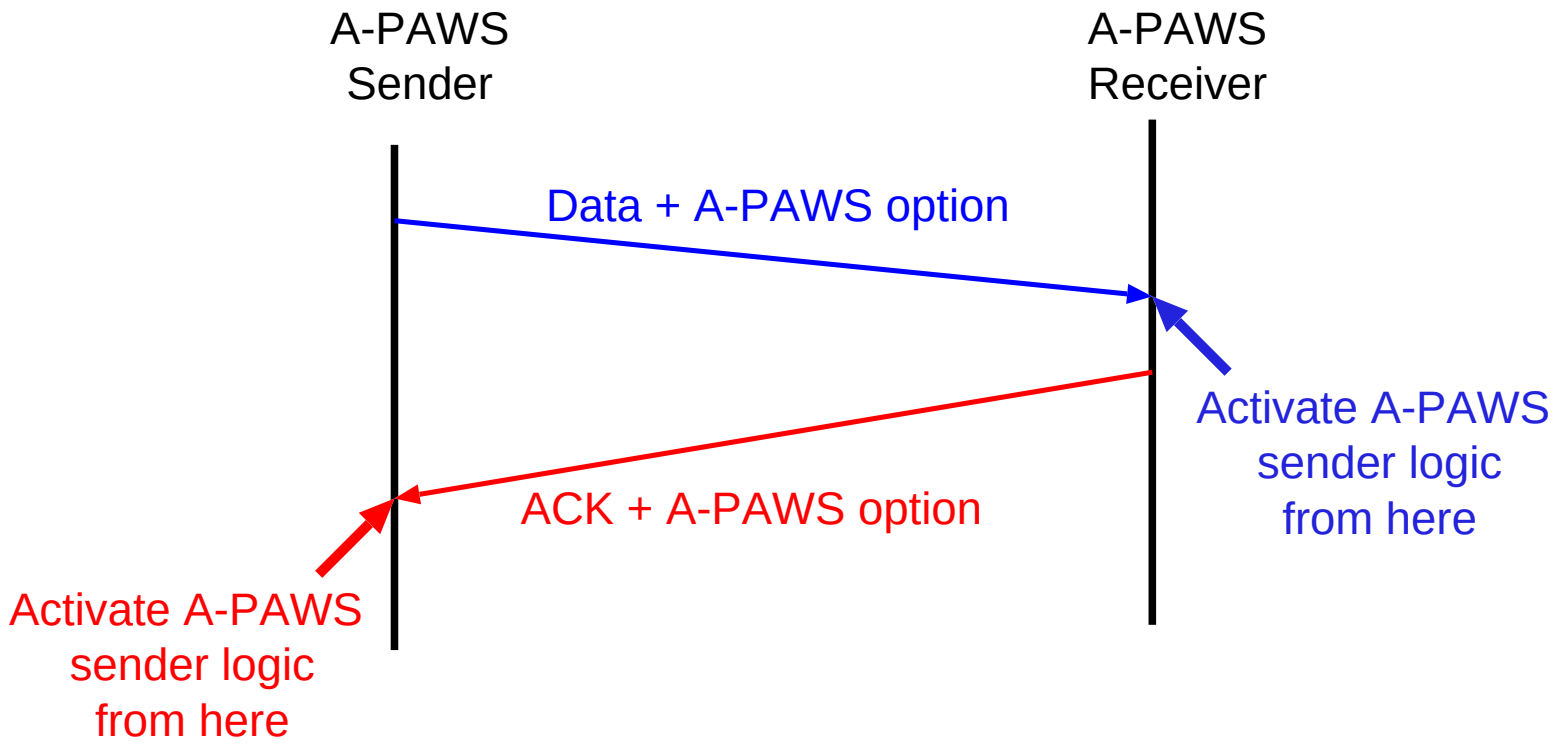
  - We only need to avoid case 3

Signaling Using non-SYN Segments

- Exchange only 2 segments for feature negotiation
- Basic Rules
  - A-PAWS node MUST always activate A-PAWS receiver logic
    - A-PAWS node uses A-PAWS receive logic whether sender uses PAWS or A-PAWS
  - A-PAWS node MUST NOT activate A-PAWS sender logic until it receives A-PAWS signaling
    - A-PAWS node uses sender logic only when peer supports A-PAWS

A-PAWS Signaling Example (1)

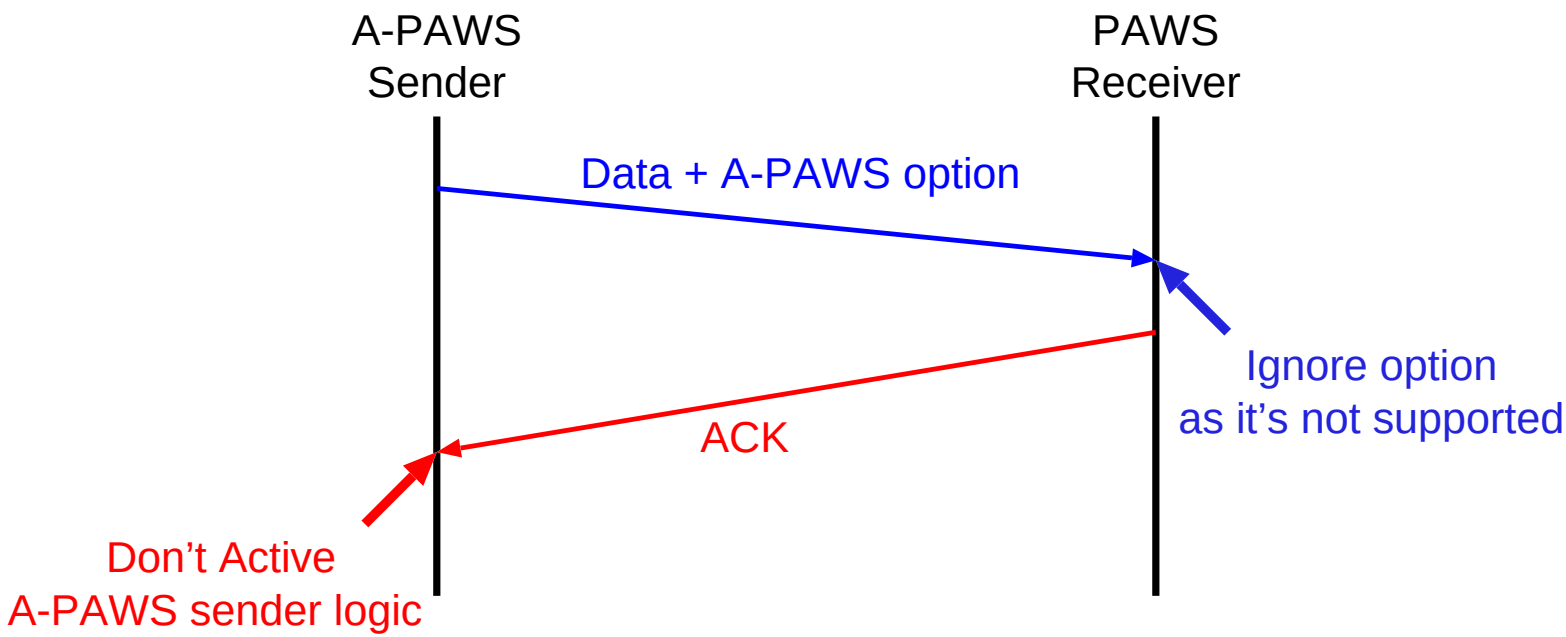
- A-PAWS sender v.s. A-PAWS receiver



- If both endpoints receive A-PAWS options, both activate A-PAWS sender logic (Case 4)

A-PAWS Signaling Example (2)

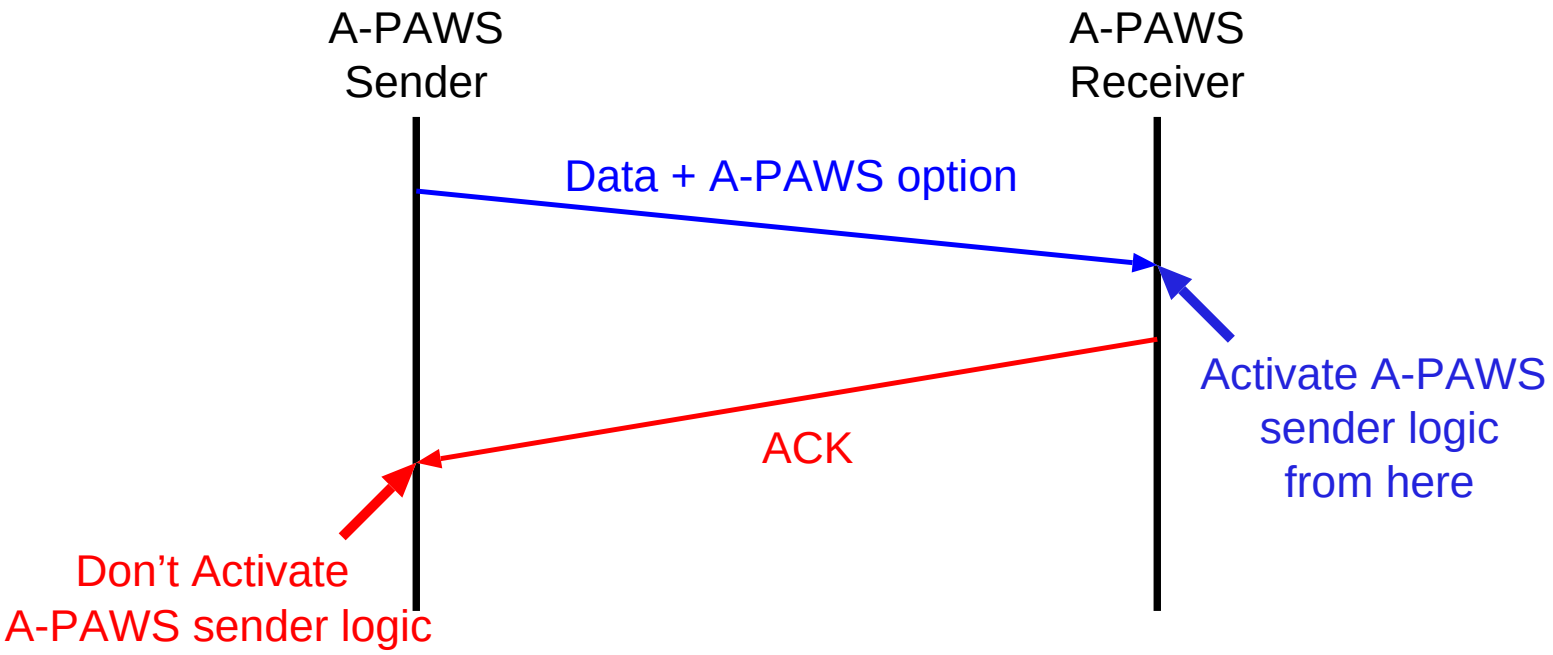
- A-PAWS sender v.s. PAWS receiver



- If receiver doesn't support A-PAWS, both ends don't activate A-PAWS sender logic (Case 1)

A-PAWS Signaling Example (3)

- A-PAWS sender v.s. A-PAWS receiver with signaling error



- If ACK + A-PAWS segment is dropped or A-PAWS option is removed, sender won't activate A-PAWS sender logic
  - Sender uses PAWS and receiver use A-PAWS (Case 2,4)

Conclusion

- What A-PAWS does
  - Provide PAWS-like protection without timestamp
    - Easy to implement because of simple logic
  - Provide the same level of security as PAWS
    - No worse than PAWS
      - Fallback to PAWS when it's necessary
  - Feature negotiation mechanism with non-SYN segments
    - might need more discussion, but it should be worth trying
    - We might be able to use similar techniques in other extensions
- What A-PAWS does not
  - Provide better protection than PAWS
  - Make PAWS obsolete
    - A-PAWS requires PAWS

Questions?

Please check draft-nishida-tcpm-apaws
for more info!

Feedbacks are welcome!