A-PAWS: Alternative Approach for PAWS

draft-nishida-tcpm-apaws-00

Yoshifumi Nishida

Background

- RFC1323 (RFC1323bis) 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?

- Timestamp
  - 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
  - No worse than PAWS
    - Fallback to PAWS when if there is a risk

What Does PAWS Do?

- Protection against packets that has the same seqno, but has different payload

- How does this happen?
  - Case 1: Packets belong to the same connection
    - Seqno circulates every 2**32 bytes
  - Case 2: Packets belong to previous connections which have the same 5 tuples
    - May happen due to rebooting or using SO_REUSEADDR
  - Case 3: Spoofed Packets or broken implementation

Protection Logic of PAWS

- Presume that TS is monotonically increased
- 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 might not be useful for malicious attack
  - Using random TS can pass PAWS check easily

A-PAWS's Logic (1)

- Protection against packets belong to the same connection
  - Seqno circulates every 2**32 bytes
- Approach
  - Count sending/receiving bytes at endpoints
  - Receiver's logic
    - If receiving bytes < 2**32, accept
    - If receiving bytes >= 2**32, do PAWS check
  - Sender's logic
    - If sending bytes < 2**32, don't put TS
    - If sending bytes >= 2**32, put TS (fallback to PAWS)

A-PAWS's Logic (2)

- Protection against packets belong to previous connections
  - May happen due to rebooting or using SO_REUSEADDR
- Approach
  - Don't use A-PAWS for a MSL upon starting up
  - Don't use A-PAWS if SO_REUSEADDR is set

Signalling

- A-PAWS requires signalling before used
  - If sender uses A-PAWS and receiver uses PAWS, packet might be discarded
- Possible Signalling Method
  - Using new TCP Option in SYN
  - Using new TCP Option in Non-SYN
  - Using Timestamp values in SYN
    - Proposed in draft-scheffenegger-tcpm-timestamp-negotiation

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
- 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!