SHA-2 Algorithm for the TCP Authentication Option (TCP-AO)

draft-nayak-tcp-sha2-02

Brian Weis

A. Sujeet Nayak

TCP-AO Background

- RFC 2385 (“Protection of BGP Sessions via the TCP MD5 Signature Option”) was published in 1998
  - Used by SPs to integrity-protect BGP sessions when they don’t trust the network in between the BGP speakers
- Ten years later it was recognized that MD5 was no longer considered strong enough and there was interest in finding a stronger integrity algorithm.
- This led to to the publishing of RFC 5925 “The TCP Authentication Option”), which included several improvements in security in addition to allowing the use of higher quality integrity-protection algorithms.

Algorithm Agility

- Algorithm agility is the property of specifying the use of new cryptographic algorithms without changing the base protocol
  - All cryptographic algorithms weaken over time, as attackers have access to more CPU power and better cryptanalytic techniques
  - This is exactly what happened to the “keyed MD5” method used in RFC 2385
- For TCP-AO:
  - RFC 5925 specifies the semantics for the integrity protection, and should not need to be re-published when new algorithms are specified.
  - RFC 5926 (“Cryptographic Algorithms for the TCP Authentication Option (TCP-AO)”) defines algorithm requirements for TCP-AO.

RFC 5926 Algorithms

- RFC 5926 defines two strong integrity methods (known as Message Authentication Codes (MACs)) to integrity-protect TCP segments:
  - HMAC-SHA-1-96
  - AES-128-CMAC-96
- RFC 5926 also defines two strong Key Derivation Functions (KDFs), which derive traffic keys from a master key configured on the communicating devices
  - KDF_HMAC_SHA1
  - KDF_AES_128_CMAC
- These are all mandatory to implement ("MUST"), but other MAC and KDF algorithms can be specified as needed

Adding SHA-256

- Our draft describes the use of SHA-256 for TCP-AO
  - MAC algorithm: HMAC-SHA256-128
  - KDF algorithm: KDF_HMAC_SHA256
- Is there anything wrong with either of the algorithms in RFC 5926?
  - No!
- Then why bother?
  - Because certain user communities have requirements to use SHA-256, as it is considerably stronger than SHA-1
  - Specifying how TCP-AO supports SHA-256 will enable those use communities to adapt TCP-AO

KDF_HMAC_SHA256

- The KDF specifies HMAC-SHA256 and takes inputs specified in RFC 5925

```
traffic_key = HMAC-SHA256(master_key, context)
```

where the context is the TCP socket pair (addresses, ports) and the ISNs (source, dest)

- The resulting traffic_key is a 256-bit value, used as the input to HMAC-SHA256-128

HMAC-SHA256-128

- The MAC specifies HMAC-SHA256 and takes inputs specified in RFC 5925

```
MAC = HMAC-SHA256-128(traffic_key, message)
```

  Where the message is the TCP segment prepended by the IP pseudoheader and TCP header options

- It produces a 256-bit value, which is truncated to 128-bits before placing into the TCP-AO option

Discussion Points

- We appreciate the comments made on the list on earlier drafts
  - We believe we resolved all comments that was possible to resolve
  - The following slides invite further discussion on a few points

Increased TCP-AO option size (1)

	HMAC-SHA-1-96	AES-128-CMAC-96	HMAC-SHA256-128
MAC (octets)	12	12	16
Header (octets)	4	4	4
Total (octets)	16	16	20

- Section 7.6 of RFC 5925 notes that 40 options octets are available
- SYN Segments
  - Section 7.6 states that current implementations expect SACK Permitted, Timestamps, and Window Scale options (15 octets)
  - That would seem to leave TCP-AO a budget of 25 octets
  - TCP-AO with HMAC-SHA256-128 consumes 20 octets, leaving 5 octets free
  - But implementation dependent alignment padding may consume another 2 octets, which leaves 3 octets free

Increased TCP-AO option size (2)

- Non-SYN segments
  - Section 7.6 states that current implementations expect to use either {SACK, Timetamps} (20 octets) or {D-SACK, Timestamps} (28 octets)
  - When SACK is used, there is a budget of 20 octets, which would be just the amount needed for HMAC-SHA256-128
  - Use of D-SACK would not be possible, however. What is the impact of not being able to use D-SACK?

Increased TCP-AO option size (3)

- Other related discussion points
  - Brandon Williams was concerned that the guidance in RFC 5925 did not include the 4-octet MSS option. This would be a problem at least for non-SYN segments
  - There was some discussion on the list regarding whether option packing can be disabled, which at least would free up room for MSS on SYN segments but non-SYN segments seem to be a problem
  - TCP-AO is likely to be used with BGP routers rather than on hosts, so perhaps the set of expected options is different and can accommodate HMAC-SHA256-128. (Investigation is ongoing.)

Updating RFC 5926

- Our Internet-Draft RECOMMENDS using SHA-256
  - There's no need to replace either current of the MACs specified in RFC 5926
  - Since we're not changing the requirements in RFC 5926 we didn't initially see a need to update it.
- But both Joe Touch and Gregory Lebovitz suggested that RFC 5926 should be updated so that all algorithm guidance is together
- We're open to whichever approach the WG prefers.

Next Steps

- Additional comments and suggestions are requested
- Security Area reviews are needed to confirm that the use of SHA-256 is correctly specified.
- We would also like to get some sense as to whether this should be a WG draft or not.