TCPM WG
Internet Draft
Obsoletes: 2385
Intended status: Proposed Standard
Expires: January 2009

J. Touch
USC/ISI
A. Mankin
R. Bonica
Juniper Networks

July 14, 2008

The TCP Authentication Option
draft-ietf-tcpm-tcp-auth-opt-01.txt

Status of this Memo

Abstract

This document specifies a TCP Authentication Option (TCP-AO) which is intended to replace the TCP MD5 Signature option of RFC-2385 (TCP MD5). TCP-AO specifies the use of stronger Message Authentication Codes (MACs) and provides more details on the association of security associations with TCP connections. TCP-AO assumes an external, out-of-band mechanism (manual or via a separate protocol) for session key establishment, parameter negotiation, and rekeying, replicating the

separation of key management and key use as in the IPsec suite. The result is intended to be a simple modification to support current infrastructure uses of TCP MD5, such as to protect BGP and LDP, and to support a larger set of MACs with minimal other system and operational changes. TCP-AO uses a new option identifier, even though it is intended to be mutually exclusive with TCP MD5 on a given TCP connection. It supports IPv6, and is fully compatible with requirements under development for an update to TCP MD5.

Table of Contents

1. Introduction

The TCP MD5 Signature (TCP MD5) is a TCP option that authenticates TCP segments, including the TCP IPv4 pseudoheader, TCP header, and TCP data. It was developed to protect BGP sessions from spoofed TCP segments which could affect BGP data or the robustness of the TCP connection itself [RFC2385][RFC4953].

There have been many recently-documented concerns about TCP MD5. Its use of a simple keyed hash for authentication is problematic because there have been escalating attacks on the algorithm itself [Be05] [Bu06]. TCP MD5 also lacks both key management and algorithm agility. This document proposes to add the latter, but suggests that TCP should not be the framework for cryptographic key management. This document replaces the TCP MD5 option to become a more general TCP Authentication Option (TCP-AO), to support the use of other, stronger hash functions and to provide a more structured recommendation on external key management. The result is compatible with IPv6, and is fully compatible with requirements under development for an update to TCP MD5 [Be07].

This document is not intended to replace the use of the IPsec suite (IPsec and IKE) to protect TCP connections [RFC4301][RFC4306]. In fact, we recommend the use of IPsec and IKE, especially where IKE's level of existing support for parameter negotiation, session key negotiation, or rekeying are desired. TCP-AO is intended for use only where the IPsec suite would not be feasible, e.g., as has been suggested is the case for some routing protocols, or in cases where keys need to be tightly coordinated with individual transport sessions [Be07].

Note that this option is intended to obsolete the use of TCP MD5, although a particular implementation may support both for backward compatibility. For a given connection, only one can be in use. TCP MD5-protected connections cannot be migrated to TCP-AO, since TCP MD5 does not support any changes to a connection's security configuration once established.

1.1. Executive Summary

This document replaces TCP MD5 as follows [RFC2385]:

- Uses a separate option Kind for TCP-AO (TBD-IANA-KIND).

- Allows TCP MD5 to continue to be used for other connections.

- Replaces MD5's one implicit MAC algorithm with two prespecified MACs (TBD-WG-MACS), and allows other MACs at the implementer's discretion.

- Allows rekeying during a TCP connection, assuming that an out-of-band protocol or manual mechanism coordinates the change of key and that incorrectly keyed segments are ignored. In such cases, a key ID makes key selection more efficient.

- Provides more detail in how this option interacts with TCP's states, event processing, and user interface.

- Proposed option is 3 bytes shorter (15 bytes overall, rather than 18) in the default case (assuming a 96-bit MAC, TBD-WG-MACLEN).

This document differs from other proposals to update TCP MD5 in that TCP-AO: [Bo07][We05][We07]:

- Is fully compatible with requirements currently under development.

- Does not support dynamic parameter negotiation.

- Does not support in-band session key negotiation.

- Does not support in-band session rekeying.

- Does not require additional timers.

- Always authenticates the the segment pseudoheader, header, and data.

- Provides more detail in how this option interacts with TCP's states, event processing, and user interface.

- Is shorter than TCP MD5 in the default case.

- Does not expose the MAC algorithm in the header.

- Requires a key ID.

- Supports TCP over either IPv4 or IPv6.

This document differs from an IPsec/IKE solution in that TCP-AO [RFC4301][RFC4306]:

- Does not support dynamic parameter negotiation.

- Does not require a key ID (SPI), but does allow one.
- Does not protect from replay attacks.
- Forces a change of connection key when a connection restarts, even when reusing a TCP socket pair (IP addresses and port numbers) [Be07].
- Does not support encryption.
- Does not authenticate ICMP messages (some may be authenticated in IPsec, depending on the configuration).

1.2. List of TBD Items

[NOTE: to be omitted upon final publication as RFC]

The following items are to be determined (TBD) prior to publication. Once a value is chosen, it should be replaced for the notation below throughout this document and the item removed from this list.

TBD-IANA-KIND new TCP option Kind for TCP-AO, assigned by IANA

TBD-WG-MACS list of default required MAC algorithms

TBD-WG-MACLEN default length of MAC used in the TCP-AO MAF

1.3. List of currently pending issues and to-do items

[NOTE: to be omitted upon final publication as an RFC]

- [IESG] Should this document deprecate TCP MD5?
- [SAAG] Which two MAC algorithms should be required as default? Should one be set as the primary default?
- [TCPM] Should TCP-AO include a negotiation protocol with a backoff, i.e., to allow non-TCP-AO endpoints to connect more quickly (or is this a security problem)? Note that this would be useful only where a rapid failure is useful, or where the TCP might backoff and use another mode (e.g., TCP MD5 or no authentication).
- [EDITORS TO-DO] Add a discussion of the use with manual keys, esp. for connections with dynamic source ports.
- [EDITORS TO-DO] Review need for LISTEN instructions.

1.4. Changes from Previous Versions

[NOTE: to be omitted upon final publication as RFC]

1.4.1. New in draft-ietf-tcp-auth-opt-01

- Require KeyID in all versions. Remove odd/even indicator of KeyID usage.
- Relax restrictions on key reuse: requiring an algorithm for nonce introduction based on ISNs, and suggest key rollover every 2^31 bytes (rather than using an extended sequence number, which introduces new state to the TCP connection).
- Clarify NAT interaction; currently does not support omitting the IP addresses or TCP ports, both of which would be required to support NATs without any coordination. This appears to present a problem for key management - if the key manager knows the received addrs and ports, it should coordinate them (as indicated in Sec 8).
- Options are included or excluded all-or-none. Excluded options are deleted, not just zeroed, to avoid the impact of reordering or length changes of such options.
- Clarified key words to exclude lower case usage.

1.4.2. New in draft-ietf-tcp-auth-opt-00

- List of TBD values, and indication of how each is determined.
- Changed TCP-SA to TCP-AO (removed 'simple' throughout).
- Removed proposed NAT mechanism; cited RFC-3947 NAT-T as appropriate approach instead.
- Made several changes coordinated in the TCP-AUTH-DT as follow:
- Added R. Bonica as co-author.
- Use new TCP option Kind in the core doc.
- Addresses the impact of explicit declines on security.
- Add limits to TSAD size (2 <= TSAD <= 256).
- Allow 0 as a legitimate KeyID.

- Allow the WG to determine the two appropriate required MAC algorithms.
- Add TO-DO items.
- Added discussion at end of Introduction as to why TCP MD5 connections cannot be upgraded to TCP-AO.

1.4.3. New in draft-touch-tcp-simple-auth-03

- Added support for NAT/NAPT.
- Added support for IPv6.
- Added discussion of how this proposal satisfies requirements under development, including those indicated in [Be07].
- Clarified the byte order of all data used in the MAC.
- Changed the TCP option exclusion bit from a bit to a list.

1.4.4. New in draft-touch-tcp-simple-auth-02

- Add reference to Bellovin's need-for-TCP-auth doc [Be07].
- Add reference to SP4 [SDNS88].
- Added notes that TSAD to be externally implemented; this was compatible with the TSAD described in the previous version.
- Augmented the protocol to allow a KeyID, required to support efficient overlapping keys during rekeying, and potentially useful during connection establishment. Accommodated by redesigning the TSAD.
- Added the odd/even indicator for the KeyID.
- Allow for the exclusion of all TCP options in the MAC calculation.

1.4.5. New in draft-touch-tcp-simple-auth-01

- Allows intra-session rekeying, assuming out-of-band coordination.
- MUST allow TSAD entries to change, enabling rekeying within a TCP connection.

- Omits discussion of the impact of connection reestablishment on BGP, because added support for rekeying renders this point moot.

- Adds further discussion on the need for rekeying.

1.5. Summary of RFC-2119 Requirements

[NOTE: a summary will be placed here prior to last call]

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.

3. The TCP Simple Authentication Option

The TCP Simple Authentication Option (TCP-AO) uses a new TCP option Kind value, (TBD-IANA-KIND).

3.1. Review of TCP MD5 Option

For review, the TCP MD5 option is shown in Figure 1.

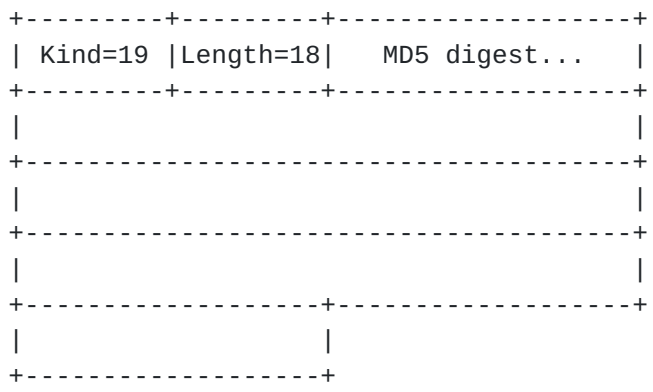

Figure 1 Current TCP MD5 Option [RFC2385]

In the current TCP MD5 option, the length is fixed, and the MD5 digest occupies 16 bytes following the Kind and Length fields, using the full MD5 digest of 128 bits [RFC1321].

The current TCP MD5 option specifies the use of the MD5 digest calculation over the following values in the following order:

1. the TCP pseudoheader (IP source and destination addresses, protocol number, and segment length)

2. TCP header excluding options and checksum

3. TCP data

4. connection key

3.2. TCP-AO Option

The new TCP-AO option is intended to be a superset of the TCP MD5 capability, and to be minimal in the spirit of SP4 [SDNS88]. TCP-AO uses a new Kind field, and similar Length field to TCP MD5, and is shown in Figure 2.

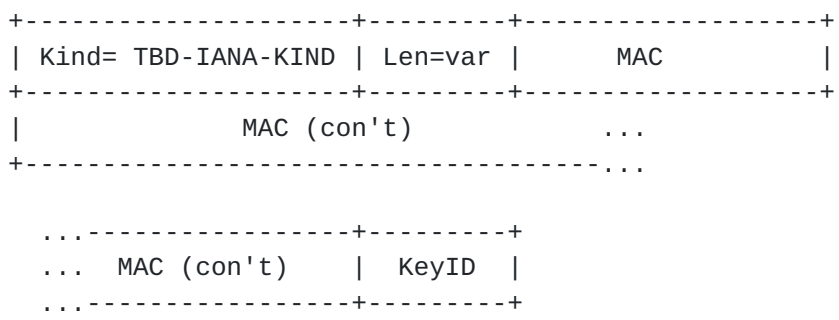

Figure 2 Proposed TCP-AO Option

The TCP-AO defines the following fields:

o Kind: An unsigned field indicating the TCP-AO Option. TCP-AO uses a new Kind value=TBD-IANA-KIND. Because of how keys are managed (see Section 4), an endpoint will not use TCP-AO for the same connection where TCP MD5 is used.

- Length: An unsigned 8-bit field indicating the length of the TCP-AO option in bytes, including the Kind, Length, and KeyID fields.

  >> The Length MUST be greater than or equal to 3.

  >> The Length value MUST be consistent with the TCP header length; this is a consistency check and to avoid overrun/underrun abuse.

  Values of 3 and other small values are of dubious utility (e.g., for MAC=NONE, or for very short MACs) but not specifically prohibited.

- MAC: Message Authentication Field. Its contents are determined by the particulars of the security association. Typical MACs are 96-128 bits (12-16 bytes), but any length that fits in the header of the segment being authenticated is allowed. Because the KeyID is one byte, it may be useful to have odd-length MACs (e.g., to select an odd number of bytes of a computed even-length MAC).

- KeyID: The last byte of the option is a KeyID field. The KeyID is used to support efficient key rollover during a connection and/or to help with key coordination during connection establishment, and will be discussed further in Sections 4.

  >> TCP-AO MUST support TBD-WG-MACS; other MACs MAY be supported [RFC2403].

  >> A single TCP segment MUST NOT have more than one TCP-AO option.

The MAC is computed over the following fields in the following order:

1. the TCP pseudoheader: IP source and destination addresses, protocol number and segment length, all in network byte order, prepended to the TCP header below. The pseudoheader is exactly as used for the TCP checksum in either IPv4 or IPv6 [RFC793][RFC2460]:

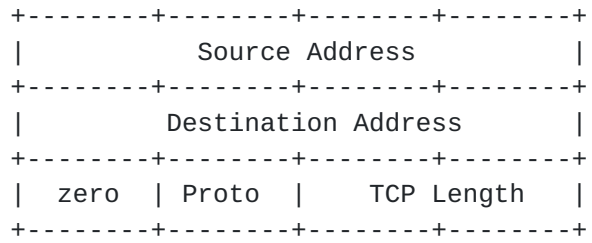

Figure 3 TCP IPv4 pseudoheader [RFC793]

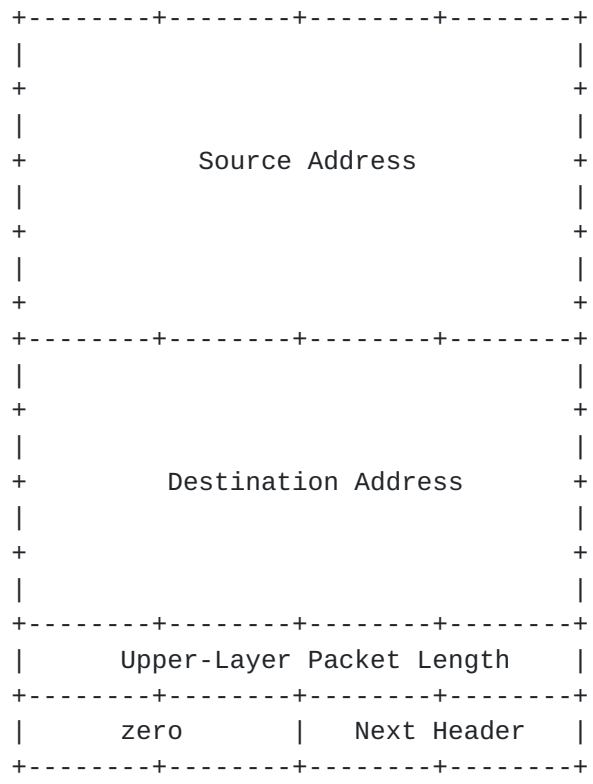

Figure 4 TCP IPv6 pseudoheader [RFC2460]

2. the TCP header, by default including options, and where the TCP checksum and TCP-AO MAC fields are set to zero, all in network byte order

3. TCP data, in network byte order

Note that the connection key is not included here; we expect that the MAC algorithm will indicate how to use the key, e.g., as HMACs do in general [RFC2104][RFC2403].

TCP-AO by default includes the TCP options because these options are intended to be end-to-end and some are required for proper TCP operation (e.g., SACK, timestamp, large windows). Middleboxes that alter TCP options en-route are a kind of attack and would be successfully detected by TCP-AO. In cases where the configuration of the connection's security association state indicates otherwise, the TCP options can be excluded from the MAC calculation. When options are excluded, all options - including TCP-AO - are skipped over during the MAC calculation (rather than being zeroed).

The TCP-AO option does not indicate the MAC algorithm either implicitly (as with TCP MD5) or explicitly. The particular algorithm used is considered part of the configuration state of the connection's security association and is managed separately (see Section 4).

MACs typically benefit from a per-connection nonce, notably in avoiding the impact of key reuse. The presence of TCP's pair of Initial Sequence Numbers presents a nonce that may be useful in that case. Such a nonce could be computed as the concatenation of the ISNs (initiator, responder), and the socket pair (addresses, ports):

o Nonce = ISN_i, ISN_r, IP_address_i, IP_address_r, port_i, port_r

The initial SYN would not know ISN_r, so that packet's nonce would use ISN_r = 0. Use of these nonces avoids the need to avoid key reuse on a per connection basis.

>> ISN and socket pair nonces MUST be used to generate unique per-session keys.

4. Security Association Management

TCP-AO relies on a TCP Security Association Database (TSAD). TSAD entries are assumed to exist at the endpoints where TCP-AO is used, in advance of the connection:

1. TCP connection identifier (ID), i.e., socket pair - IP source address, IP destination address, TCP source port, and TCP destination port [RFC793]. TSAD entries are uniquely determined by their TCP connection ID, which is used to index those entries.

   >> There MUST be no more than one matching TSAD entry per direction for a TCP connection ID.

2. For each of inbound (for received TCP segments) and outbound (for sent TCP segments) directions for this connection (except as noted):

   a. TCP option exclusion flag. When 0, this flag allows default operation, i.e., TCP options are When 1, all options (including TCP-AO) are excluded from all MAC calculations (skipped over, not simply zeroed).

      >> The TCP option exclusion flag MUST default to 0 (i.e., options not excluded).

>> The TCP option flag list MUST NOT change during a TCP connection.

b. An ordered list of zero or more connection key tuples. Each tuple is defined as the set <KeyID, MAC type, key length, connection key> as follows:

>> TSAD key tuple components MUST NOT change during a connection.

>> The set of TSAD key tuples MAY change during a connection, but KeyIDs of those tuples MUST NOT overlap. I.e., tuple parameter changes MUST be accompanied by key changes.

i. KeyID. A single byte used to differentiate overlapping Connection keys.

>> A TSAD implementation MUST support at least two KeyIDs per connection per direction, and MAY support up to 256.

>> A KeyID MAY have any value, 0-255 inclusive.

ii. MAC type. Indicates the MAC used for this connection, as per IKEv2 Transform Type 3 [RFC4306]. This includes the MAC algorithm (e.g., HMAC-MD5, HMAC-SHA1, UMAC, etc.) and the length of the MAC as truncated to (e.g., 96, 128, etc.).

>> A MAC type of "NONE" MUST be supported, to indicate that authentication is not used in this direction; this allows asymmetric use of TCP-AO.

>> At least one direction (inbound/outbound) SHOULD have a non-"NONE" MAC in practice, but this MUST NOT be strictly required by an implementation.

>> When the outbound MAC is set to values other than "NONE", TCP-AO MUST occur in every outbound TCP segment for that connection; when set to NONE or when no tuple exists, TCP-AO MUST NOT occur in those segments.

>> When the inbound MAC is set to values other than "NONE", TCP-AO MUST occur in every inbound TCP segment for that connection; when set to "NONE" or when no tuple exists, TCP-AO SHOULD NOT be added to those segments, but MAY occur and MUST be ignored.

iii. Key length. A byte indicating the length of the connection key in bytes.

iv. Connection key. A byte sequence used for connection keying, this may be derived from a separate shared key by an external protocol over a separate channel. This sequence is used in network standard byte order in MAC calculations.

It is anticipated that TSAD entries for TCP connections in states other than CLOSED can be stored in the TCP Control Block (TCB) or in a separate database (see Section 6.1 for notes on the latter); TSAD entries for pending connections (in passive or active OPEN) may be stored in a separate database. This means that in a single host there should be only a single database which is consulted by all pending connections, the same way that there is only one set of TCBs. Multiple databases could be used to support virtual hosts, i.e., groups of interfaces.

Note that the TCP-AO fields omit an explicit algorithm ID; that algorithm is already specified by the TCP connection ID and stored in the TSAD.

Also note that this document does not address how TSAD entries are created by users/processes; it specifies how they must be destroyed corresponding to connection states, but users/processes may destroy entries as well. It is presumed that a TSAD entry affecting a particular connection cannot be destroyed during an active connection - or, equivalently, that its parameters are copied to TSAD entries local to the connection (i.e., instantiated) and so changes would affect only new connections. The TSAD could be managed by a separate application protocol, and can be stored in a separate database if desired.

5. TCP-AO Interaction with TCP

The following is a description of how TCP-AO affects various TCP states, segments, events, and interfaces. This description is intended to augment the description of TCP as provided in RFC793 [RFC793].

5.1. User Interface

The TCP user interface supports active and passive OPEN, SEND, RECEIVE, CLOSE, STATUS and ABORT commands.

>> TCP OPEN, or the sequence of commands that configure a connection to be in the active or passive OPEN state, MUST be augmented so that a TSAD entry can be configured.

Users are advised to not inappropriately reuse keys [RFC3562]. As noted in Section 3.2, this is accomplished in TCP-AO by the use of unique per-connection nonces in conjunction with conventional keys.

>> TCP STATUS SHOULD be augmented to allow the TSAD entry of a current or pending connection to be read (for confirmation).

>> A TCP-AO implmentation MUST allow TSAD entries for ongoing TCP connections (i.e., not in the CLOSED state) to be modified. Parameters not used to index a connection MAY be modified; parameters used to index a connection MUST NOT be modified.

TSAD entries for TCP connections not in the CLOSED state are deleted indirectly using the CLOSE or ABORT commands.

TCP SEND and RECEIVE are not affected by TCP-AO.

5.2. TCP States and Transitions

TCP includes the states LISTEN, SYN-SENT, SYN-RECEIVED, ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT, and CLOSED.

>> A TSAD entry MAY be associated with any TCP state.

>> A TSAD entry MAY underspecify the TCP connection for the LISTEN state. Such an entry MUST NOT be used for more than one connection progressing out of the LISTEN state.

5.3. TCP Segments

TCP includes control (at least one of SYN, FIN, RST flags set) and data (none of SYN, FIN, or RST flags set) segments. Note that some control segments can include data (e.g., SYN).

>> All TCP segments MUST be checked against the TSAD for matching TCP connection IDs.

>> TCP segments matching TSAD entries with non-NULL MACs without TCP-AO, or with TCP-AO and whose MACs and KeyIDs do not validate MUST be silently discarded.

>> TCP segments with TCP-AO but not matching TSAD entries MUST be silently accepted; this is required for equivalent function with TCPs not implementing TCP-AO.

>> Silent discard events SHOULD be signaled to the user as a warning, and silent accept events MAY be signaled to the user as a warning. Both warnings, if available, MUST be accessible via the STATUS interface. Either signal MAY be asynchronous, but if so they MUST be rate-limited. Either signal MAY be logged; logging SHOULD allow rate-limiting as well.

All TCP-AO processing occurs between the interface of TCP and IP; for incoming segments, this occurs after validation of the TCP checksum. For outgoing segments, this occurs before computation of the TCP checksum.

Note that the TCP-AO option is not negotiated. It is the responsibility of the receiver to determine when TCP-AO is required and to enforce that requirement.

5.4. Sending TCP Segments

The following procedure describes the modifications to TCP to support TCP-AO when a segment departs.

1. Check the segment's TCP connection ID against the TSAD

2. If there is NO TSAD entry, omit the TCP-AO option. Proceed with computing the TCP checksum and transmit the segment.

3. If there is a TSAD entry with zero key tuples, omit the TCP-AO option. Proceed with computing the TCP checksum and transmit the segment.

4. If there is a TSAD entry and a key tuple and the outgoing MAC is NONE, omit the TCP-AO option. Proceed with computing the TCP checksum and transmit the segment.

5. If there is a TSAD entry and a key tuple and the outgoing MAC is not NONE:

    a. Augment the TCP header with the TCP-AO, inserting the appropriate Length and KeyID based on the indexed TSAD entry. Update the TCP header length accordingly.

b. Compute the MAC using the indexed TSAD entry and data from the segment as specified in Section 3.2, including the TCP pseudoheader and TCP header. Include or exclude the options as indicated by the TSAD entry's TCP option exclusion flag.

c. Insert the MAC in the TCP-AO field.

d. Proceed with computing the TCP checksum on the outgoing packet and transmit the segment.

5.5. Receiving TCP Segments

The following procedure describes the modifications to TCP to support TCP-AO when a segment arrives.

1. Check the segment's TCP connection ID against the TSAD.

2. If there is NO TSAD entry, proceed with TCP processing.

3. If there is a TSAD entry with zero key tuples, proceed with TCP processing.

4. If there is a TSAD entry with a key tuple and the incoming MAC is NONE, proceed with TCP processing.

5. If there is a TSAD entry with a key tuple and the incoming MAC is not NONE:

   a. Check that the segment's TCP-AO Length matches the length indicated by the indexed TSAD.

      i. If Lengths differ, silently discard the segment. Log and/or signal the event as indicated in Section 5.3.

   b. Use the KeyID value to index the appropriate key for this connection.

      i. If the TSAD has no entry corresponding to the segment's KeyID, silently discard the segment.

   c. Compute the segment's MAC using the indexed TSAD entry and portions of the segment as indicated in Section 3.2.

      Again, if options are excluded (as per the TCP option exclusion flag), they are skipped over (rather than zeroed) when used as input to the MAC calculation.

i. If the computed MAC differs from the TCP-AO MAC field value, silently discard the segment. Log and/or signal the event as indicated in Section 5.3.

d. Proceed with TCP processing of the segment.

It is suggested that TCP-AO implementations validate a segment's Length field before computing a MAC, to reduce the overhead incurred by spoofed segments with invalid TCP-AO fields.

5.6. Impact on TCP Header Size

The TCP-AO option typically uses a total of 17-19 bytes of TCP header space. TCP-AO is no larger than and typically 3 bytes smaller than the TCP MD5 option (assuming a 96-bit MAC). Although TCP option space is limited, we believe TCP-AO is consistent with the desire to authenticate TCP at the connection level for similar uses as were intended by TCP MD5.

6. Key Establishment and Duration Issues

The TCP-AO option does not provide a mechanism for connection key negotiation or parameter negotiation (MAC algorithm, length, or use of the TCP-AO option) or rekeying during a connection. We assume out-of-band mechanisms for key establishment, parameter negotiation, and rekeying. This separation of key use from key management is similar to that in the IPsec security suite [RFC4301][RFC4306].

We encourage users of TCP-AO to apply known techniques for generating appropriate keys, including the use of reasonable connection key lengths, limited connection key sharing, and limiting the duration of connection key use [RFC3562]. This also includes the use of per-connection nonces, as suggested in Section 3.2.

TCP-AO supports rekeying in which new keys are negotiated out-of-band, either via a protocol or a manual procedure [RFC4808]. New keys use is coordinated using the out-of-band mechanism to update the TSAD at both TCP endpoints. In the default case, where only a single key is used at a time, the temporary use of invalid keys would result in packets being dropped; TCP is already robust to such drops. Such drops may affect TCP's throughput temporarily, as a result TCP-AO benefits from the use of congestion control support for temporary path outages.

>> TCP-AO SHOULD be deployed in conjunction with support for selective acknowledgement (SACK), including support for multiple lost segments in the same round trip [RFC2018][RFC3517].

Note that TCP-AO's support for rekeying is designed to be minimal in the default case. Segments carry only enough context to identify the security association [RFC4301][RFC4306]. In TCP-AO, this context is provided by the socket pair (IP addresses and ports for source and destination). The TSAD can contain multiple concurrent keys, where the KeyID field is used to identify the key that corresponds to a segment, to avoid the need for expensive trial-and-error testing of keys in sequence.

The KeyID field is also useful in coordinating keys for new connections. A TSAD may be configured that matches the unbound source port, which would return a set of possible keys. The KeyID would then indicate which key, allowing more efficient connection establishment; otherwise, the keys could have been tried in sequence. See also Section 6.1.

Implementations are encouraged to keep keys in a suitably private area. Users of TCP-AO are encouraged to use different keys for inbound and outbound MACs on a given TCP connection.

6.1. Implementing the TSAD as an External Database

The TSAD implementation is considered external to TCP-AO. When an external database is used, it would be useful to consider the interface between TCP-AO and the TSAD. The following is largely a restatement of information in Section 4.

The TSAD API is accessed during a connection as follows:

- TCP connection identifier (ID) (The socket pair, sent as 4 byte IP source address, 4 byte IP destination address, 2 byte TCP source port, 2 byte TCP destination port).

- Direction indicator (sent as a single byte, 0x00 = inbound, 0x01 = outbound)

- Number of bytes to be sent/received (two bytes); this is used on the send side to trigger bytecount-based KeyID changes, and on the receive side only for statistics or length-sensitive KeyID selection.

  >> TCP-AO implementations SHOULD change keys for a connection at least every 2^31 bytes, to avoid resending segments with the same TCP sequence number, data, and length under the same key.

- KeyID (single byte); this is provided only by a receiver (i.e., matching the KeyID of the received segment), where a sender would leave this unspecified (and the call would return the appropriate KeyID to use).

The call passes the number of bytes sent/received, and an indication of the direction (send/receive), to enable traffic-based key rollover.

The source port can be 'unbound', indicated by the value 0x0000. In this case, the source port is considered a wildcard, and all corresponding TSAD entries (indexed by the KeyID) are returned as a list. This feature is used during connection establishment.

TSAD calls return the following parameters:

- TCP option exclusion flag (one byte, with 0x00 having the meaning "exclude none" and 0x01 meaning "exclude all").

- An ordered list of zero or more connection key tuples: <KeyID, MAC type, MAC length, connection key>

    - KeyID (one byte)

    - MAC type (four bytes, an IKEv2 Transform Type 3 ID [RFC4306])

    - Key length (one byte)

    - Connection key (byte sequence, indicating the key value)

When the TSAD returns zero keys, it is indicating that there are no currently valid keys for the connection.

7. Interactions with TCP MD5

TCP-AO is intended to be deployed without regard for existing TCP MD5 option support.

>> A TCP implementation MUST NOT use both TCP-AO and TCP MD5 for a particular TCP connection, but MAY support TCP-AO and TCP MD5 simultaneously for different connections.

The Kind value explicitly indicates which of TCP-AO or TCP MD5 is used for a particular connection in TCP segments.

It is possible that the TSAD could be augmented to support TCP MD5, although use of a TSAD-like system is not described in RFC2385.

It is possible to require TCP-AO for a connection or TCP MD5, but it is not possible to require 'either'. Note that when TCP MD5 is required on for a connection, it must be used [RFC2385]. This prevents combined use of the two options for a given connection, to be determined by the other end of the connection.

8. Interactions with NAT/NAPT Devices

TCP-AO can interoperate across NAT/NAPT devices, which modify the IP addresses, and may also modify TCP port numbers and/or TCP options. TCP options can be excluded on a per-connection basis.

IP addresses and port numbers would preferably be coordinated across a NAT/NAPT device, such that the sender and receiver both know the IP address and TCP port numbers of the received packet. In that case, the sender computes the packet as it would be received, i.e., using the receiver's version of the IP pseudoheader and TCP header.

Where such knowledge of the address and port translations are not known, NAT/NAPT traversal can be handled in similar ways to IPsec [RFC2766][RFC3947]. I.e., traversing such a device using a tunnel to avoid the NAT/NAPT from translating fields in the TCP and IP headers TCP-AO uses in its MAC calculation. Such a tunnel may need to coincide with the channel over which keys are exchanged, as in IPsec NAT traversal [RFC3947].

9. Evaluation of Requirements Satisfaction

TCP-AO satisfies all the current requirements for a revision to TCP MD5, as indicated in [Be07] and under current developemt. This should not be a surprise, as the majority of the evolving requirements are derived from its design. The following is a summary of those requirements and notes where relevant.

1. Protected Elements - see Section 3.2.

    a. TCP pseudoheader, including IPv4 and IPv6 versions. Note that we do not allow optional coverage because IP addresses define a connection. If they can be coordinated across a NAT/NAPT, the sender can compute the MAC based on the received values; if not, a tunnel is required.

    b. TCP header. Note that we do not allow optional port coverage because ports define a connection. If they can be coordinated across a NAT/NAPT, the sender can compute the MAC based on the received values; if not, a tunnel is required.

c. TCP options. Allows exclusion of TCP options from coverage, as required.

d. TCP data. Done.

2. Option structure requirements

a. Privacy. TCP-AO exposes only the key index, MAC, and overall option length. Note that short MACs could be obscured by using longer option lengths but specifying a short MAC length (this is equivalent to a different MAC algorithm, and is specified in the TSAD entry). See Section 3.2.

b. Allow optional per connection. Done - see Sections 5.3, 5.4, and 5.5.

c. Require non-optional. Done - see Sections 5.3, 5.4, and 5.5.

d. Standard parsing. Done - see Section 3.2.

e. Compatible with Large Windows. Done - see Section 3.2. The size of the option is intended to allow use with Large Windows and SACK. See also Section 1.1, which indicates that TCP-AO is 4 bytes shorter than TCP MD5 in the default case, assuming a 96-bit MAC.

f. Compatible with SACK. Done - see Section 3.2. The size of the option is intended to allow use with Large Windows and SACK. See also Section 6 regarding key management. See also Section 1.1, which indicates that TCP-AO is 4 bytes shorter than TCP MD5 in the default case.

3. Cryptography requirements

a. Baseline defaults. TCP-AO uses TBD-WG-MACS as the default, as noted in Section 3.2.

b. Good algorithms. TCP-AO uses TBD-WG-MACS as the default, but does not otherwise specify the algorithms used. That would be specified in the key management protocol, and should be limited there.

c. Algorithm agility. TCP-AO allows any desired algorithm, subject to TCP option space limitations, as noted in Section 3.2. The TSAD allows separate connections to use different algorithms.

d. Pre-TCP processing. Done - see Sections 5.3, 5.4, and 5.5. Note that pre-TCP processing is required, because TCP segments cannot be discarded solely based on a combination of connection state and out-of-window checks; many such segments, although discarded, cause a host to respond with a replay of the last valid ACK, e.g. [RFC793].

e. Parameter changes require key changes. TSAD parameters that should not change during a connection (TCP connection ID, receiver TCP connection ID, TCP option exclusion list) cannot change. Other parameters change only when a key is changed, using the key tuple mechanism in the TSAD. See Section 4.

4. Keying requirements. TCP-AO does not specify a key management system, but does indicate a proposed interface to the TSAD, allowing a completely separate key system.

a. Intraconnection rekeying. Supported by the KeyID and multiple key tuples in a TSAD entry; see Section 4.

b. Efficient rekeying. Supported by the KeyID. See Section 6.

c. Automated and manual keying. Supported by the TSAD interface. See Section 6.

d. Key management agnostic. Supported by the TSAD interface. See Section 6.1.

5. Expected constraints

a. Silent failure. Done - see Sections 5.3, 5.4, and 5.5.

b. At most one such option per segment. Done - see Section 3.2.

c. Outgoing all or none. Done - see Section 5.4.

d. Incoming all checked. Done - see Section 5.5.

e. Non-interaction with TCP MD5. Done - see Section 7.

f. Optional ICMP discard. Done - see Section 10.

g. Allows use of NAT/NAPT devices. Done - see Section 8.

h. Maintain TCP connection semantics, in which only the socket pair defines a TCP association and all its security parameters. Done - see Sections 4 and 8.

i. Try to avoid creating a CPU DOS attack opportunity. Done - see Section 10.

10. Security Considerations

Use of TCP-AO, like use of TCP MD5 or IPsec, will impact host performance. Connections that are known to use TCP-AO can be attacked by transmitting segments with invalid MACs. Attackers would need to know only the TCP connection ID and TCP-AO Length value to substantially impact the host's processing capacity. This is similar to the susceptibility of IPsec to on-path attacks, where the IP addresses and SPI would be visible. For IPsec, the entire SPI space (32 bits) is arbitrary, whereas for routing protocols typically only the source port (16 bits) is arbitrary. As a result, it would be easier for an off-path attacker to spoof a TCP-AO segment that could cause receiver validation effort. However, we note that between Internet routers both ports could be arbitrary (i.e., determined a-priori out of band), which would constitute roughly the same off-path antispoofing protection of an arbitrary SPI.

TCP-AO, like TCP MD5, may inhibit connectionless resets. Such resets typically occur after peer crashes, either in response to new connection attempts or when data is sent on stale connections; in either case, the recovering endpoint may lack the connection key required (e.g., if lost during the crash). This may result in time-outs, rather than more responsive recovery after such a crash.

TCP-AO does not include a fast decline capability, e.g., where a SYN-ACK is received without an expected TCP-AO option and the connection is quickly reset or aborted. Normal TCP operation will retry and timeout, which is what should be expected when the intended receiver is not capable of the TCP variant required anyway. Backoff is not optimized because it would present an opportunity for attackers on the wire to abort authenticated connection attempts by sending spoofed SYN-ACKs without the TCP-AO option.

TCP-AO does not expose the MAC algorithm used to authenticate a particular connection; that information is kept in the TSAD at the endpoints, and is not indicated in the header.

TCP-AO is intended to provide similar protections to IPsec, but is not intended to replace the use of IPsec or IKE either for more robust security or more sophisticated security management.

TCP-AO does not address the issue of ICMP attacks on TCP. IPsec makes recommendations regarding dropping ICMPs in certain contexts, or requiring that they are endpoint authenticated in others [RFC4301].

There are other mechanisms proposed to reduce the impact of ICMP attacks by further validating ICMP contents and changing the effect of some messages based on TCP state, but these do not provide the level of authentication for ICMP that TCP-AO provides for TCP [Go07].

>> A TCP-AO implementation MUST allow the system administrator to configure whether TCP will ignore incoming ICMP messages of Type 3 Codes 2-4 intended for connections that match TSAD entries with non-NONE inbound MACs. An implementation SHOULD allow ignored ICMPs to be logged.

This control affects only ICMPs that currently require 'hard errors' which would abort the TCP connection. This recommendation is intended to be similar to how IPsec would handle those messages [RFC4301].

TCP-AO includes the TCP connection ID in the MAC calculation. This prevents connections using the same key (for whatever reason) from potentially enabling a traffic-crossing attack, in which segments to one socket pair are diverted to attack a different socket pair. When multiple connections use the same key, it would be useful to know that packets intended for one ID could not be (maliciously or otherwise) modified in transit and end up being authenticated for the other ID. The ID cannot be zeroed, because to do so would require that the TSAD index was unique in both directions (ID->key and key->ID). That requirement would place an additional burden of uniqueness on keys within endsystems, and potentially across endsystems. Although the resulting attack is low probability, the protection afforded by including the received ID warrants its inclusion in the MAC, and does not unduly increase the MAC calculation or key management system.

The use of any security algorithm can present an opportunity for a CPU DOS attack, where the attacker sends false, random segments that the receiver under attack expends substantial CPU effort to reject. In IPsec, such attacks are reduced by the use of a large Security Parameter Index (SPI) and Sequence Number fields to partly validate segments before CPU cycles are invested validated the Integrity Check Value (ICV). In TCP-AO, the socket pair performs most of the function of IPsec's SPI, and IPsec's Sequence Number, used to avoid replay attacks, isn't needed due to TCP's Sequence Number, which is used to reorder received segments. Unfortunately, it is not useful to validate TCP's Sequence Number before performing a TCP-AO authentication calculation, because out-of-window segments can still cause TCP protocol actions (e.g., ACK retransmission) [RFC793]. It is similarly not useful to add a separate Sequence Number field to the TCP-AO option, because doing so could cause a change in TCP's behavior even when segments are valid.

11. IANA Considerations

The TCP-AO option defines no new namespaces.

The TCP-AO option uses the TCP option Kind value TCP-IANA-KIND, allocated by IANA from the TCP option Kind namespace.

To specify MAC algorithms, TCP-AO uses the 4-byte namespace of IKEv2 Transform Type 3 IDs [RFC4306].

[NOTE: The following to be removed prior to publication as an RFC]

The TCP-AO option requires that IANA allocate a value from the TCP option Kind namespace, to be replaced for TCP-IANA-KIND throughout this document.

12. Acknowledgments

This document was inspired by the revisions to TCP MD5 suggested by Brian Weis and Ron Bonica [Bo07][We05]. Russ Housley suggested L4/application layer management of the TSAD. The KeyID field was motivated by Steve Bellovin. Eric Rescorla suggested the use of ISNs as nonces, and Brian Weis extended the nonce to incorporate the entire connection ID. Alfred Hoenes, Charlie Kaufman, and Adam Langley provided substantial feedback. The document is the result of collaboration with the TCP Authentication Design team (tcp-auth-dt).

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

13. References

13.1. Normative References

[RFC793] Postel, J., "Transmission Control Protocol," STD 007, RFC 793, Standard, Sept. 1981.

[RFC2018] Mathis, M., Mahdavi, J., Floyd, S. and A. Romanow, "TCP Selective Acknowledgement Options", RFC 2018, Proposed Standard, April 1996.

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

[RFC2385] Heffernan, A., "Protection of BGP Sessions via the TCP MD5 Signature Option," RFC 2385, Proposed Standard, Aug. 1998.

[RFC2403] Madson, C., R. Glenn, "The Use of HMAC-MD5-96 within ESP and AH," RFC 2403, Proposed Standard, Nov. 1998.

[RFC2460] Deering, S., Hinden, R., "Internet Protocol, Version 6 (IPv6) Specification," RFC 2460, Proposed Standard, Dec. 1998.

[RFC3517] Blanton, E., Allman, M., Fall, K., and L. Wang, "A Conservative Selective Acknowledgment (SACK)-based Loss Recovery Algorithm for TCP", RFC 3517, Proposed Standard, April 2003.

[RFC4306] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol," RFC 4306, Proposed Standard, Dec. 2005.

13.2. Informative References

[Be05] Bellovin, S., E. Rescorla, "Deploying a New Hash Algorithm," presented at the First NIST Cryptographic Hash Workshop, Oct. 2005. http://csrc.nist.gov/pki/HashWorkshop/2005/program.htm

[Be07] Eddy, W., (ed), S. Bellovin, J. Touch, R. Bonica, "Problem Statement and Requirements for a TCP Authentication Option," draft-bellovin-tcpsec-01, (work in progress), Jul. 2007.

[Bu06] Burr, B., "NIST Cryptographic Standards Status Report," Invited talk at Internet 2 5th Annual PKI R&D Workshop, April 2006. http://middleware.internet2.edu/pki06/proceedings/

[Bo07] Bonica, R., et. al, "Authentication for TCP-based Routing and Management Protocols," draft-bonica-tcp-auth-06 , (work in progress), Feb. 2007.

[Go07] Gont, F., "ICMP attacks against TCP," draft-ietf-tcpm-icmp-attacks-03, (work in progress), Mar. 2008.

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm," RFC-1321, Informational, April 1992.

[RFC2104] Krawczyk, H., Bellare, M., Canetti, R., "HMAC: Keyed-Hashing for Message Authentication," RFC 2104, Informational, Feb. 1997.

[RFC2766] Tsirtsis, G., Srisuresh, P., "Network Address Translation - Protocol Translation (NAT-PT)," RFC 2766, Proposed Standard, Feb. 2000.

[RFC3562] Leech, M., "Key Management Considerations for the TCP MD5 Signature Option," RFC 3562, Informational, July 2003.

[RFC3947] Kivinen, T., B. Swander, A. Huttunen, V. Volpe, "Negotiation of NAT-Traversal in the IKE," RFC 3947, Jan. 2005.

[RFC4301] Kent, S., K. Seo, "Security Architecture for the Internet Protocol," RFC 4301, Proposed Standard, Dec. 2005.

[RFC4808] Bellovin, S., "Key Change Strategies for TCP-MD5," RFC 4808, Informational, Mar. 2007.

[RFC4953] Touch, J., "Defending TCP Against Spoofing Attacks," RFC4953, Jul. 2007.

[SDNS88] Secure Data Network Systems, "Security Protocol 4 (SP4)," Specification SDN.401, Revision 1.2, July 12, 1988.

[We05] Weis, B., "TCP Message Authentication Code Option," draft-weis-tcp-mac-option-00, (expired work in progress), Dec. 2005.

[We07] Weis, B., et al., "Automated key selection extension for the TCP Authentication Option," draft-weis-tcp-auth-auto-ks-03, (work in progress), Oct. 2007.

Author's Addresses

Joe Touch
USC/ISI
4676 Admiralty Way
Marina del Rey, CA 90292-6695
U.S.A.

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

Allison Mankin
Washington, DC
U.S.A.

Phone: 1 301 728 7199
Email: mankin@psg.com
URL: http://www.psg.com/~mankin/

Ronald P. Bonica
Juniper Networks
2251 Corporate Park Drive
Herndon, VA 20171
U.S.A.

Email: rbonica@juniper.net

Full Copyright Statement

Intellectual Property Statement

attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.