Key Coordination enhancement for TCP-AO
draft-duan-tcpm-tcp-ao-rekeying-00.txt

Abstract

TCP-AO technology was proposed to obsolete the TCP-MD5 option which was developed to protect the BGP sessions between routers. Besides of allowing users to choose which cryptographic algorithm(s) they want to use to meet their security needs, TCP-AO provides key coordination mechanism giving the ability to move from one key to another within the same connection with zero segment loss by using two ID fields i.e. KeyID and RNextKeyID. The sender uses the RNextKeyID to indicate the receiver using the preferred MKT which will authenticate the next incoming segments. However, if the sender finds its MKT which is used to authenticate the outgoing segments has been attacked and should be changed into a new one, it can do nothing but wait for receiver to send a segment which carries a different RNextKeyID.

In this case, the communication becomes dangerous probably because the sender always authenticates outgoing segments by an attacked key before the receiver wants to change the incoming key. This document provides a method giving the sender ability to inform the other part change the RNextKeyID when the sender finds the key used in outgoing segment is not safe any longer.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."
Internet-Draft   Key Coordination enhancement for TCP-AO      March 2010

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on September 3, 2010.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the BSD License.
Table of Contents

1. Introduction ........................................... 4
   1.1. Requirements Language ............................. 4
2. Review of the TCP-AO option ............................. 4
3. The main idea ........................................... 6
4. Operation ............................................... 6
   4.1. Per-Connection TCP-AO Parameters .................. 7
   4.2. Sending TCP Segments ............................... 7
   4.3. Receiving TCP Segments ............................. 8
5. Security Considerations ................................. 10
6. References .............................................. 10
   6.1. Normative References ............................... 10
   6.2. Informative References ............................. 10
Appendix A. Additional Stuff ............................... 10
Authors’ Addresses ......................................... 10
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].

However, due to the fatal attack on MD5 and the difficulty to changing the long-term key, TCP-AO [tcp-ao] was proposed to obsolete the TCP-MD5 option. In TCP-AO, more strong authentication algorithms are supported and two kinds of ID fields i.e. KeyID and RNextKeyID are explicitly given. The KeyID and RNextKeyID form a simple key coordination mechanism giving the ability to move from one key to another within the same connection with zero segment loss. If one part wants to change the key used to authenticate the incoming segments, it can insert the preferred ID of MKT into the RNextKeyID to notify the receiver. And the receiver could set current key to the RNextKeyID MKT when the RNextKeyID MKT is ready. It works well for changing the incoming key. However, if a sender finds its MKT which is used to authenticate outgoing segments has been attacked and should change into a new one, it can do nothing but has to wait for the receiver sending a segment which carries a different RNextKeyID.

In this case, the communication becomes dangerous probably because the sender always authenticates outgoing segment by an attacked key before the receiver changes the incoming key. This paper proposes a method which the sender can trigger the other part to change the RNextKeyID when the sender finds the former outgoing key is not safe any longer.

When the sender wants to change the key which is used to authenticate the outgoing segment, it sends outgoing segment with 0Xff in RNextKeyID field. The 0Xff in RNextKeyID field is used to trigger the receiver to use a new key ID in the RNextKeyID.

1.1. Requirements Language

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 [RFC2119].

2. Review of the TCP-AO option

The TCP-AO option is shown in Figure 1.
The TCP-AO defines the fields as follows:

- **Kind**: An unsigned 1-byte field indicating the TCP-AO Option. TCP-AO uses a new Kind value of TBD-IANA-KIND.
  
  An endpoint MUST NOT use TCP-AO for the same connection in which TCP MD5 is used. When both options appear, TCP MUST silently discard the segment.

  A single TCP segment MUST NOT have more than one TCP-AO option. When multiple TCP-AO options appear, TCP MUST discard the segment.

- **Length**: An unsigned 1-byte field indicating the length of the TCP-AO option in bytes, including the Kind, Length, KeyID, RNExtKeyID, and MAC fields.
  
  The Length value MUST be consistent with the TCP header length; this is a consistency check and avoids overrun/underrun abuse. When the Length value is invalid, TCP MUST discard the segment.

  Values of 4 and other small values larger than 4 (e.g., indicating MAC fields of very short length) are of dubious utility but are not specifically prohibited.

- **KeyID**: An unsigned 1-byte field indicating the MKT used to generate the traffic keys which were used to generate the MAC that authenticates this segment.
  
  It supports efficient key changes during a connection and/or to help with key coordination during connection establishment. Note that the KeyID has no cryptographic properties – it need not be random, nor are there any reserved values.
KeyID values MAY be the same in both directions of a connection, but do not have to be and there is no special meaning when they are.

- **RNextKeyID**: An unsigned 1-byte field indicating the MKT that the sender is ready to receive authenticated segments, i.e., the desired ‘receive next’ KeyID.

  It supports efficient key change coordination. Note that the RNextKeyID has no cryptographic properties - it need not be random, nor are there any reserved values.

- **MAC**: Message Authentication Code. 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.

  Required support for TCP-AO MACs are defined in [ao-crypto]; other MACs MAY be supported.

3. The main idea

The problem is that if a sender finds its MKT which is used to authenticate the outgoing segment has been attacked and should change into a new one, it can do nothing but has to wait for the receiver sending a segment which carries a different RNextKeyID.

The problem stems from the limitation of the RNextKeyID, that is, the RNextKeyID is only used to notify the receiver to change the key which is used to authenticate the next incoming segments but has no ability to change the sender’s key which is used to authenticate its outgoing segments.

The main idea of this method is that the sender can trigger receiver to change Rnext_key pointer. RNextKeyID may define 0xff as reserved message which is used for sender to trigger receiver changing Rnext_key. The sender sends a segment where the RNextKeyID field is 0Xff which means that it wants to use a new MKT to authenticate the future outgoing segments. Receiving this segment, the receiver changes its Rnext_key pointer, sets the RNextKeyID field to the MKT ID which is indicated by the Rnext_key pointer and sends the next segment.

4. Operation
4.1. Per-Connection TCP-AO Parameters

TCP-AO uses a small number of parameters associated with each connection that uses the TCP-AO option, once instantiated. These values can be stored in the Transport Control Block (TCP) [RFC793]. These values are explained in subsequent sections of this document as noted; they include:

1. Current_key – the MKT currently used to authenticate outgoing segments, whose SendID is inserted in outgoing segments as KeyID. Incoming segments are authenticated using the MKT corresponding to the segment and the KeyID in its TCP-AO header, as matched against the MKT TCP connection identifier and the MKT RecvID. There is only one Current_key at any given time on a particular connection.

2. Rnext_key – the MKT currently preferred for incoming (received) segments, whose RecvID is inserted in outgoing segments as RNExtKeyID. This parameter is changed only by manual user intervention or MKT management protocol operation. The configuration is as following:
   1. If the sender wants to change a new key to authenticate his next incoming segment, then he sets the Rnext_key to a new MKT which he is ready for.
   2. If the sender wants to change a new key to authenticate his future outgoing segment, then he sets the Rnext_key to a special reserved message 0Xff.

3. A pair of Sequence Numbers Extensions (SNEs). SNEs are used to prevent replay attacks. Each SNE is initialized to zero upon connection establishment.

4. One or more MKTs. These are the MKTs that match this connection’s socket pair.

4.2. Sending TCP Segments

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

1. Find the per-connection parameters for the segment:
   a. If the segment is a SYN, then this is the first segment of a new connection. Find the matching MKT for this segment based on the segment’s socket pair.
Internet-Draft   Key Coordination enhancement for TCP-AO      March 2010

i.  If there is no matching MKT, omit the TCP-AO option. Proceed with transmitting the segment.

ii. If there is a matching MKT, then set the per-connection parameters as needed. Proceed with the step 2.

b. If the segment is not a SYN, then determine whether TCP-AO is being used for the connection and use the MKT as indicated by the Current_key value from the per-connection parameters and proceed with the step 2.

2. Using the per-connection parameters:

a. Augment the TCP header with the TCP-AO, inserting the appropriate Length and KeyID based on the MKT indicated by Current_key (using the Current_key MKT’s SendID as the TCP-AO KeyID). Update the TCP header length accordingly.

b. Determine SND.SNE.

c. Determine the appropriate traffic key, i.e., as pointed to by Current_key. I.e., use the send_SYN_traffic_key for SYN segments, and the send_other_traffic_key for other segments.

d. Determine the RNExtKeyID indicated by the Rnext_key pointer, and insert it in the TCP-AO option (using the Rnext_key MKT’s RecvID as the TCP-AO RNExtKeyID).

e. Compute the MAC using the MKT (and cached traffic key) and data from the segment as specified in Section 7.1.

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

g. Proceed with transmitting the segment.

4.3. Receiving TCP Segments

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

1. Find the per-connection parameters for the segment:

Find the matching MKT for this segment, using the segment’s socket pair and its TCP-AO KeyID, matched against the MKT’s TCP connection identifier and the MKT’s RecvID.

i. If there is no matching MKT, remove the TCP-AO option from the segment. Proceed with further TCP handling of the
ii. If there is a matching MKT, then set the per-connection parameters as needed. Proceed with the step 2.

2. Using the per-connection parameters:
   a. Check that the segment’s TCP-AO Length matches the length indicated by the MKT.
      i. If lengths differ, silently discard the segment.
   b. Determine the segment’s RCV.SNE.
   c. Determine the segment’s traffic key from the MKT. I.e., use the receive_SYN_traffic_key for SYN segments, and receive_other_traffic_key for other segments.
   d. Compute the segment’s MAC using the MKT (and its derived traffic key).
      i. If the computed MAC differs from the TCP-AO MAC field value, silently discard the segment.
   e. Compare the received RNExtKeyID value to the currently active outgoing KeyID value (Current_key MKT’s SendID).
      i. If they match, no further action is required.
      ii. If they differ, determine whether the RNExtKeyID MKT is ready.

If(RNExtKeyID is not 0Xff)
   1. If the MKT corresponding to the segment’s socket pair and RNExtKeyID is not available, no action is required (RNExtKeyID of a received segment needs to match the MKT’s SendID).
   2. If the matching MKT corresponding to the segment’s socket pair and RNExtKeyID is available:
      a. Set Current_key to the RNExtKeyID MKT.

If(RNExtKeyID is 0Xff)

Set the Rnext_key to another new MKT.
f. Proceed with TCP processing of the segment.

5. Security Considerations

This document inherits all of the security considerations of the TCP-AO.

6. References

6.1. Normative References


6.2. Informative References


Appendix A. Additional Stuff

Authors’ Addresses

Shili Duan (editor)
ZTE Corporation
No. 68, Zijinghua Road, Yuhuatai District
Nanjing, Jiangsu 210012
China

Phone: +86 25 52877648
Email: duan.shili@zte.com.cn