Configuring TCP Authentication Option

This document describes how to configure TCP authentication option on Cisco NX-OS devices.

About TCP Authentication Option

With TCP Authentication Option (TCP-AO), defined in RFC 5925, you can protect long-lived TCP connections against replays using stronger Message Authentication Codes (MACs).

TCP-AO is the proposed replacement for TCP MD5, defined in RFC 2385. Unlike TCP MD5, TCP-AO is resistant to collision attacks and provides algorithmic agility and support for key management.

TCP-AO has the following distinct features:

- TCP-AO supports the use of stronger Message Authentication Codes (MACs) to enhance the security of long-lived TCP connections.
- TCP-AO protects against replays for long-lived TCP connections, and coordinates key changes between endpoints by providing a more explicit key management.

The TCP-AO feature deprecates TCP MD5. Cisco NX-OS devices will continue to support the TCP-MD5 option for legacy BGP peers. However, a configuration in which one end of the peering is configured with the TCP MD5 option and the other with the TCP-AO option is not supported

TCP-AO Key Chain

TCP-AO is based on traffic keys and Message Authentication Codes (MACs) generated using the keys and a MAC algorithm. The traffic keys are derived from master keys that you can configure in a TCP-AO key chain. Use the **key chain** *key-chain-name* **tcp** command in the global configuration mode to create a TCP-AO key chain and configure keys in the chain. The TCP-AO key chain must be configured on both the peers communicating via a TCP connection.

Keys in a TCP-AO key chain have the following configurable properties:

Configurable Property	Description
send-id	Key identifier of the TCP-AO option of the outgoing segment. The send identifier configured on a router must match the receive identifier configured on the peer.
recv-id	Key identifier compared with the TCP-AO key identifier of the incoming segment during authentication. The receive identifier configured on a router must match the send identifier configured on the peer.
cryptographic-algorithm	The MAC algorithm to be used to create MACs for outgoing segments. The algorithm can be one of the following: • AES-128-CMAC authentication algorithm • HMAC-SHA-1 authentication algorithm • HMAC-SHA-256 authentication algorithm
include-tcp-options	This flag indicates whether TCP options other than TCP-AO will be used to calculate MACs. With this flag enabled, the contents of all options along with a zero-filled authentication option, is used to calculate the MAC. When the flag is disabled, all options other than TCP-AO are excluded from MAC calculations. This flag is disabled by default. **Note** The configuration of this flag is overridden by the application configuration when the application configuration is available.
send-lifetime	This configuration determines the time for which a key is valid and can be used for TCP-AO-based authentication of TCP segments. When the lifetime of key elapses and the key expires, the next key with the youngest lifetime is selected.
key-string	The key string is a pre-shared master key configured on both peers and is used to derive the traffic keys.

TCP-AO Format

```
+------------+------------+------------+------------+
|  Kind=29   |   Length   |   KeyID    | RNextKeyID |
+------------+------------+------------+------------+
|                     MAC           ...
+-----------------------------------...

   ...-----------------+
```

```
...  MAC (con't)    |
...-----------------+
```

The fields of the TLV format are as follows:

- Kind: Indicates TCP-AO with a value of 29.
- Length: Indicates the length of the TCP-AO sequence.
- KeyID: The send identifier of the Master Key Tuple (MKT) that was used to generate the traffic keys.
- RNextKeyID: The receive identifier of the MKT that is ready to be used to authenticate received segments.
- MAC: The MAC computed for the TCP segment data and the prefixed pseudo header.

Master Key Tuples

Traffic keys are the keying material used to compute the message authentication codes of individual TCP segments.

Master Key Tuples (MKTs) enable you to derive unique traffic keys, and to include the keying material required to generate those traffic keys. MKTs indicate the parameters under which the traffic keys are configured. The parameters include whether TCP options are authenticated, and indicators of the algorithms used for traffic key derivation and MAC calculation.

Each MKT has the following two identifiers:

- **SendID**: The **SendID** identifier is inserted as the KeyID identifier of the TCP AO option of the outgoing segments.
- **RecvID**: The **RecvID** is matched against the TCP AO KeyID of the incoming segments.

TCP-AO Key Rollover

TCP-AO keys are valid for a defined duration configured using the send-lifetime. If send-lifetime is not configured the key is considered inactive. Key rollover is initiated based on the send lifetimes of keys.

TCP-AO coordinates the use of new MKTs using the RNextKeyID and KeyID field on the TCP-AO option field. For hitless key rollovers, new and old keys in keychain configurations need to have at least 15 minutes of overlap. This is required so that the TCP-AO has enough time to coordinate use of the new MKT.

When key rollover is initiated, one of the peer routers, say Router A, indicates that the rollover is necessary. To indicate that the rollover is necessary, Router A sets the RNextKeyID to the receive identifier (recv-id) of the new MKT to be used. On receiving the TCP segment, the peer router, say Router B, looks up the send identifier (send-id) in its database to find the MKT indicated by the RNextKeyID in the TCP-AO payload. If the key is available and valid, Router B sets the current key to the new MKT. After Router B has rolled over, Router A also sets the current key to the new Primary Key Tuples.

Key rollover is initiated with overlapping send-lifetimes and send-lifetime expiry

If you do not configure a new key that can be activated before the expiry of the current key, the key may time out and expire. Such an expiry can cause retransmissions with the peer router rejecting segments authenticated with the expired key. The connection may fail due to Retransmission Time Out (RTO). When new valid keys are configured and usable, the connection can be re-established.

Guidelines and Limitations

- The send-id and recv-id of each key in the key chain must be unique. Because send-id and recv-id must be chosen from the range 0 to 255, the TCP-AO key chain can have a maximum of 256 keys.
- Only one keychain can be associated with an application connection. Rollover is always performed within the keys in this keychain.
- If the key in use expires, expect segment loss until a new key that has a valid lifetime is configured on each side and keys rollover.
- All the following configurations must be done for a TCP-AO keychain key to be considered active: send-id, recv-id, key-string, send-lifetime and cryptographic-algorithm.
- The key chain software process will use the newest key (youngest key) based on the send-lifetime configuration. Or, whichever key was configured last if the same send-lifetime is configured for two different keys in the same key chain. Configuring two keys with identical send-lifetimes is not best practice or recommended.
- User MUST configure minimum 15 minutes overlapping time between the two overlapping keys.
- Modifying the configuration of a key in use such as key-string, send-id, recv-id, cryptographic-algorithm or send-lifetime will result in TCP connection flap.
- A keychain's configuration type must match the type it has been linked to within the client protocol. If an attempt is made to mismatch these types, a syslog message is generated to notify the user. For example: It is not supported if a keychain named keychain_abc is configured as a Macsec keychain but is associated as a TCP keychain with BGP. Similarly, the case where the keychain is first associated with the client (a process known as forward-referencing) and then configured as a different keychain type, is also not supported.

Configure TCP Key Chain and Keys

Before you begin

- Ensure that the key-string, send-lifetimes, cryptographic-algorithm, and ids of keys match on both peers.
- Ensure that the send-id on a router matches the recv-id on the peer router. We recommend using the same id for both the parameters unless there is a need to use separate key spaces.
- The send-id and recv-id of a key cannot be reused for another key in the same key chain.
- The key-string is encrypted and stored in the Type-6 format if the AES password encryption feature is enabled and a primary key is configured. Otherwise, the password will be stored in the Type-7 encrypted format.
- For more details, see Configuring a Primary Key and Enabling the AES Password Encryption Feature

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal`	Enters global configuration mode.
Step 2	**key chain** *name* **tcp** **Example:** `switch(config)# key chain bgp-keys tcp`	Enters keychain configuration mode for the keychain that you specified.
Step 3	**key** *key-ID* **Example:** `switch(config-tcpkeychain)# key 13`	Enters key configuration mode for the key that you specified. The *key-ID* argument must be a whole number between 0 and 65535.
Step 4	**send-id** *send-ID* **Example:** `switch(config-tcpkeychain-tcpkey)# send-id 2`	Specifies the send identifier for the key. The send-ID must be in the range from 0 to 255 and unique value per key chain.
Step 5	**recv-id** *recv-ID* **Example:** `switch(config-tcpkeychain-tcpkey)# recv-id 2`	Specifies the recieve identifier for the key. The recv-ID must be in the range from 0 to 255 and unique value per key chain.
Step 6	**key-string** *[encryption-type] text-string* **Example:** `switch(config-tcpkeychain-tcpkey)# key-string 0 AS3cureStr1ng`	Configures the text string for the key. The text-string argument is alphanumeric, case-sensitive, and supports special characters. The encryption-type argument can be one of the following values: • 0—The text-string argument that you enter is unencrypted text. This is the default. • 6—Beginning with Cisco NX-OS Release 10.3(3)F, the Cisco proprietary (Type-6 encrypted) method is supported on Cisco Nexus 9000 Series platform switches. • 7—The text-string argument that you enter is encrypted. The encryption method is a Cisco proprietary method. This option is useful when you are entering a text string based on the encrypted output of a **show key chain** command that you ran on another Cisco NX-OS device.

<table>
<tr><th></th><th>Command or Action</th><th>Purpose</th></tr>
<tr><td></td><td></td><td>The key-string command has limitations on using the following special characters in the text-string:
<table>
<tr><th>Special Character</th><th>Description</th><th>Comments</th></tr>
<tr><td>|</td><td>Vertical bar or pipe</td><td>Unsupported at start of key-string</td></tr>
<tr><td>></td><td>Greater than</td><td>Unsupported at start of key-string</td></tr>
<tr><td>\</td><td>Backslash</td><td>Unsupported start or end of a key-string</td></tr>
<tr><td>(</td><td>Left parenthesis</td><td>Unsupported at start of key-string</td></tr>
<tr><td>'</td><td>Apostrophe</td><td>Unsupported at start of key-string</td></tr>
<tr><td>"</td><td>Quotation mark</td><td>Unsupported at start of key-string</td></tr>
<tr><td>?</td><td>Question mark</td><td>Supported. However, press Ctrl-V before entering a question mark (?).</td></tr>
</table>
For more information on the special characters usage in commands, see Understanding the Command-Line Interface section.</td></tr>
<tr><td>Step 7</td><td>[no] cryptographic-algorithm {HMAC-SHA-1 | HMAC-SHA-256 | AES-128-CMAC }
Example:
<code>switch(config-tcpkeychain-tcpkey)# cryptographic-algorithm HMAC-SHA-1</code></td><td>Specifies the algorithm to be used to compute MACs for TCP segments. You can configure only one cryptographic algorithm per key.</td></tr>
<tr><td>Step 8</td><td>send-lifetime [local] start-time duration [duration-value | infinite | end-time]
Example:
<code>switch(config-tcpkeychain-tcpkey)# send-lifetime local 01:01:01 Jan 01 2023 01:01:01 Jan 10 2023</code></td><td>Configures a send lifetime for the key. By default, the device treats the start-time and end-time arguments as UTC. If you specify the local keyword, the device treats these times as local times.
The start-time argument is the time of day and date that the key becomes active.
You can specify the end of the send lifetime with one of the following options:
• duration duration-value —The length of the lifetime in seconds. The maximum</td></tr>
</table>

	Command or Action	Purpose
		length is 2147483646 seconds (approximately 68 years). • **infinite**—The send lifetime of the key never expires. • end-time —The end-time argument is the time of day and date that the key becomes inactive.
Step 9	(Optional) **include-tcp-options** **Example:** `switch(config-tcpkeychain-tcpkey)# include-tcp-options`	An optional configuration to specify if the full 'TCP Options" part of the TCP header (other than TCP AO option) needs to be included while computing the 'MAC' digest of the packets.

Verifying the TCP Keychain

Command	Purpose
show key chain *[name] [detail]*	Displays the keychains configured on the device.

```
switch# show key chain
Key-Chain bgp_keys tcp
  Key 2 -- text 7 "070e234f"
    send-id 2
    recv-id 2
    cryptographic-algorithm AES_128_CMAC
    send lifetime UTC (08:17:00 May 29 2023)-(08:21:00 May 29 2023)
    include-tcp-options
  Key 3 -- text 7 "070c2058"
    send-id 3
    recv-id 4
    cryptographic-algorithm HMAC-SHA-1
    send lifetime UTC (08:20:00 May 29 2023)-(always valid) [active]
    include-tcp-options
  Key 12 -- text ""
    send lifetime UTC (08:20:00 May 29 2023)-(always valid)
```

Note [active] indicates that the key is valid and active otherwise the key is inactive. In the above example only key 3 is active and usable.

The **show key chain detail** command will explicitly display any active and inactive key(s). In the case of Type 6 encryption, the show key chain detail command will display if the type 6 key-string is decryptable or not. It will also display the newest (youngest) active send key that the client is currently using to authenticate packets.

```
switch# show key chain detail
Key-Chain bgp_keys tcp
  Key 1 -- text 6 "JDYk9k4kmaciqaH6Eu2+9C0tmCRl9k7JAMYs/fXGbW1lmHP88PAA=="
    Type6 Decryptable: yes
```

```
    send-id 1
    recv-id 1
    cryptographic-algorithm HMAC-SHA-1
    send lifetime local (18:15:42 May 15 2023)-(always valid) [active]
    include-tcp-options
    accept-ao-mismatch
  Key 2 -- text 6 "JDYkB+Fs8u3ujRDpFSu4tH6H7iTS45JJA6sKeGsBD0L3HjGDeg9AA=="
    Type6 Decryptable: yes
    send-id 2
    recv-id 2
    cryptographic-algorithm AES_128_CMAC
    send lifetime local (17:10:47 May 15 2023)-(18:15:42 May 15 2023) [inactive]

  youngest active send key: 1
```

Configuration Example for a TCP Keychain

This example shows how to configure a TCP keychain named bgp_keys. Each key text string is encrypted. The keys have overlapping lifetime configurations:

```
key chain bgp_keys tcp
  key 1
    send-id 1
    recv-id 1
    key-string 7 070e234f
    send-lifetime 01:00:00 Oct 10 2023 01:00:00 Oct 11 2023
    cryptographic-algorithm AES-128-CMAC
  key 2
    send-id 2
    recv-id 2
    key-string 7 075e731f
    send-lifetime 00:45:00 Oct 11 2023 01:00:00 Oct 12 2023
    cryptographic-algorithm HMAC-SHA-256
    include-tcp-options
```