Implementing Master Key Tuple Configuration

This feature specifies TCP Authentication Option (TCP-AO), which replaces the TCP MD5 option. TCP-AO uses the Message Authentication Codes (MACs), which provides the following:

Master Key Tuple Configuration

This feature specifies TCP Authentication Option (TCP-AO), which replaces the TCP MD5 option. TCP-AO uses the Message Authentication Codes (MACs), which provides the following:

- Protection against replays for long-lived TCP connections
- More details on the security association with TCP connections than TCP MD5
- A larger set of MACs with minimal other system and operational changes

TCP-AO is compatible with Master Key Tuple (MKT) configuration. TCP-AO also protects connections when using the same MKT across repeated instances of a connection. TCP-AO protects the connections by using traffic key that are derived from the MKT, and then coordinates changes between the endpoints.

Note TCPAO and TCP MD5 are never permitted to be used simultaneously. TCP-AO supports IPv6, and is fully compatible with the proposed requirements for the replacement of TCP MD5.

Cisco provides the MKT configuration via the following configurations:

- keychain configuration
- tcp ao keychain configuration

The system translates each key, such “key_id” that is under a keychain, as MKT. The keychain configuration owns part of the configuration like secret, lifetimes, and algorithms. While the “tcp ao keychain” mode owns the TCP AO-specific configuration for an MKT (send_id and receive_id).

Keychain Configurations

Configuration Guidelines

In order to run a successful configuration, ensure that you follow the configuration guidelines:

- An allowed value range for both Send_ID and Receive_ID is 0 to 255.
- You can link only one keychain to an application neighbor.
- Under the same keychain, if you configure the same send_id key again under the keys that have an overlapping lifetime, then the old key becomes unusable until you correct the configuration.
- The system sends a warning message in the following scenarios:
  - If there is a change in Send_ID or Receive_ID.
  - If the corresponding key is currently active, and is in use by some connection.
- BGP neighbor can ONLY use one of the authentication options:
  - MD5
  - EA
  - AO

Note If you configure one of these options, the system rejects the other authentication options during the configuration time.

Configuration Guidelines for TCP AO BGP Neighbor

The configuration guidelines are:

- Configure all the necessary configurations (key_string, MAC_algorithm, send_lifetime, accept_lifetime, send_id, receive_id) under key_id with the desired lifetime it wants to use the key_id for.
- Configure a matching MKT in the peer side with exactly same lifetime.
- Once a keychain-key is linked to tcp-ao, do not change the components of the key. If you want TCP to consider another key for use, you can configure that dynamically. Based on the ‘start-time’of send lifetime, TCP AO uses the key.
- Send_ID and Receive_ID under a key_id (under a keychain) must have the same lifetime range. For example, send-lifetime==accept-lifetime.

  TCP considers only expiry of send-lifetime to transition to next active key and it does not consider accept-lifetime at all.
- Do not configure a key with send-lifetime that is covered by another key’s send-lifetime.

For example, if there is a key that is already configured with send-lifetime of “04:00:00 November 01, 2017 07:00:00 November 01, 2017” and the user now configures another key with send-lifetime of “05:00:00 November 01, 2017 06:00:00 November 01, 2017”, this might result into connection flap.

TCP AO tries to transition back to the old key once the new key is expired. However, if the new key has already expired, TCP AO can’t use it, which might result in segment loss and hence connection flap.

- Configure minimum of 15 minutes of overlapping time between the two overlapping keys. When a key expires, TCP does not use it and hence out-of-order segments with that key are dropped.
- We recommend configuring send_id and receive_id to be same for a key_id for simplicity.
- TCP does not have any restriction on the number of keychains and keys under a keychain. The system does not support more than 4000 keychains, any number higher than 4000 might result in unexpected behaviors.

Keychain Configuration

```
key chain <keychain_name>
   key <key_id>
      accept-lifetime <start-time> <end-time>
      key-string <master-key>
      send-lifetime <start-time> <end-time>
      cryptographic-algorithm <algorithm>
   !
!
```

TCP Configuration

TCP provides a new tcp ao submode that specifies SendID and ReceiveID per key_id per keychain.

```
tcp ao
    keychain <keychain_name1>
        key-id <key_id> send_id <0-255> receive_id <0-255>
        !
```

Example:

```
tcp ao
 keychain bgp_ao
  key 0 SendID 0 ReceiveID 0
  key 1 SendID 1 ReceiveID 1
  key 2 SendID 3 ReceiveID 4
 !
 keychain ldp_ao
  key 1 SendID 100 ReceiveID 200
  key 120 SendID 1 ReceiveID 1
 !
```

BGP Configurations

Applications like BGP provide the tcp-ao keychain and related information that it uses per neighbor. Following are the optional configurations per tcp-ao keychain:

- include-tcp-options
- accept-non-ao-connections

```
router bgp <AS-number>
neighbor <neighbor-ip>
  remote-as <remote-as-number>
  ao <keychain-name> include-tcp-options enable/disable  <accept-ao-mismatch-connections>
!
```

XML Configurations

BGP XML

TCP-AO XML

```
<?xml version="1.0" encoding="UTF-8"?>
<Request>
 <Set>
  <Configuration>
   <IP_TCP>
    <AO>
      <Enable>
       true
      </Enable>
      <KeychainTable>
        <Keychain>
         <Naming>
          <Name> bgp_ao_xml </Name>
         </Naming>
         <Enable>
          true
         </Enable>
          <KeyTable>
           <Key>
            <Naming>
             <KeyID> 0 </KeyID>
            </Naming>
             <SendID> 0 </SendID>
             <ReceiveID> 0 </ReceiveID>
           </Key>
          </KeyTable>
        </Keychain>
      </KeychainTable>
    </AO>
   </IP_TCP>
  </Configuration>
 </Set>
 <Commit/>
</Request>
```

Verification

To verify the keychain database, use the show tcp authentication keychain <keychain-name> command in EXEC mode. The following output displays all the keychain database details:

```
Keychain name: tcp_ao_keychain1, configured for tcp-ao
Desired key: 1
Detail of last notification from keychain:
Time: 'Jan 23 12:07:39.128', event: Config update, attr: Crypto algorithm, key: 1
Total number of keys: 1
Key details:
    Key ID: 1, Active, Valid
```

```
    Active_state: 1, invalid_bits: 0x0, state: 0x110
    Key is configured for tcp-ao, Send ID: 1, Receive ID: 1
    Crypto algorithm: AES_128_CMAC_96, key string chksum: 00028222
    Detail of last notification from keychain:
    Time: 'Jan 23 12:07:39.128', event: Config update, attr: Crypto algorithm
    No valid overlapping key
    No keys invalidated

Total number of usable (Active & Valid) keys: 1
    Keys: 1,
Total number of peers: 24
Peer details:
    Peer: 0x7fc2f00242f8,
    Current key not yet available
    RNext key: 1
    Traffic keys: send_non_SYN: 00000000, recv_non_SYN: 00000000

    Peer: 0x7fc2f0024618,
    Current key not yet available
    RNext key: 1
    Traffic keys: send_non_SYN: 00000000, recv_non_SYN: 00000000

    Peer: 0x7fc2f00247f8,
    Current key not yet available
    RNext key: 1
    Traffic keys: send_non_SYN: 00000000, recv_non_SYN: 00000000

    Peer: 0x7fc2f00249d8,
    Current key not yet available
    RNext key: 1
    Traffic keys: send_non_SYN: 00000000, recv_non_SYN: 00000000

    Peer: 0x7fc2f0024bb8,
    Current key not yet available
    RNext key: 1
    Traffic keys: send_non_SYN: 00000000, recv_non_SYN: 00000000

    Peer: 0x7fc320037a08,
    Current key not yet available
    RNext key: 1
    Traffic keys: send_non_SYN: 00000000, recv_non_SYN: 00000000

    Peer: 0x7fc320037d78,
    Current key not yet available
    RNext key: 1
    Traffic keys: send_non_SYN: 00000000, recv_non_SYN: 00000000

    Peer: 0x7fc3200386d8,
    Current key not yet available
    RNext key: 1
    Traffic keys: send_non_SYN: 00000000, recv_non_SYN: 00000000

    Peer: 0x7fc3200388b8,
    Current key not yet available
    RNext key: 1
    Traffic keys: send_non_SYN: 00000000, recv_non_SYN: 00000000

    Peer: 0x7fc320038a98,
    Current key not yet available
    RNext key: 1
    Traffic keys: send_non_SYN: 00000000, recv_non_SYN: 00000000

    Peer: 0x7fc35000d3f8,
    Current key: 1
```

```
Traffic keys: send_non_SYN: 00476017, recv_non_SYN: ffd520f9
RNext key: 1
Traffic keys: send_non_SYN: 00000000, recv_non_SYN: 00000000
Last 1 keys used:
    key: 1, time: Jan 23 12:07:41.953, reason: Peer requested rollover

Peer: 0x7fc320038e78,
Current key not yet available
RNext key: 1
Traffic keys: send_non_SYN: 00000000, recv_non_SYN: 00000000

Peer: 0x7fc350012758,
Current key not yet available
RNext key: 1
Traffic keys: send_non_SYN: 00000000, recv_non_SYN: 00000000

Peer: 0x7fc2f0026bc8,
Current key not yet available
RNext key: 1
Traffic keys: send_non_SYN: 00000000, recv_non_SYN: 00000000

Peer: 0x7fc320048b08,
Current key: 1
Traffic keys: send_non_SYN: 004a05b5, recv_non_SYN: fff639b2
RNext key: 1
Traffic keys: send_non_SYN: 00000000, recv_non_SYN: 00000000
Last 1 keys used:
    key: 1, time: Jan 23 12:07:44.209, reason: No current key set

Peer: 0x7fc2f4008388,
Current key: 1
Traffic keys: send_non_SYN: 0029837c, recv_non_SYN: 002af030
RNext key: 1
Traffic keys: send_non_SYN: 00000000, recv_non_SYN: 00000000
Last 1 keys used:
    key: 1, time: Jan 23 12:07:44.229, reason: No current key set

Peer: 0x7fc350017198,
Current key: 1
Traffic keys: send_non_SYN: ffdb7322, recv_non_SYN: fff1fb23
RNext key: 1
Traffic keys: send_non_SYN: 00000000, recv_non_SYN: 00000000
Last 1 keys used:
    key: 1, time: Jan 23 12:07:45.419, reason: Peer requested rollover

Peer: 0x7fc320049098,
Current key: 1
Traffic keys: send_non_SYN: ffed0d67, recv_non_SYN: ffe4f959
RNext key: 1
Traffic keys: send_non_SYN: 00000000, recv_non_SYN: 00000000
Last 1 keys used:
    key: 1, time: Jan 23 12:07:55.180, reason: No current key set

Peer: 0x7fc32005d2a8,
Current key: 1
Traffic keys: send_non_SYN: 0021b461, recv_non_SYN: fffe679e
RNext key: 1
Traffic keys: send_non_SYN: 00000000, recv_non_SYN: 00000000
Last 1 keys used:
    key: 1, time: Jan 23 12:07:56.894, reason: No current key set

Peer: 0x7fc350035c88,
Current key: 1
Traffic keys: send_non_SYN: 00296167, recv_non_SYN: fff1c236
```

```
    RNext key: 1
    Traffic keys: send_non_SYN: 00000000, recv_non_SYN: 00000000
    Last 1 keys used:
        key: 1, time: Jan 23 12:07:57.859, reason: Peer requested rollover

    Peer: 0x7fc35003fb18,
    Current key: 1
    Traffic keys: send_non_SYN: ffc95844, recv_non_SYN: ffcdfd4f
    RNext key: 1
    Traffic keys: send_non_SYN: 00000000, recv_non_SYN: 00000000
    Last 1 keys used:
        key: 1, time: Jan 23 12:08:00.754, reason: Peer requested rollover

    Peer: 0x7fc350049638,
    Current key: 1
    Traffic keys: send_non_SYN: 002ff48b, recv_non_SYN: ffbe71b9
    RNext key: 1
    Traffic keys: send_non_SYN: 00000000, recv_non_SYN: 00000000
    Last 1 keys used:
        key: 1, time: Jan 23 12:08:10.014, reason: Peer requested rollover

    Peer: 0x7fc350053928,
    Current key: 1
    Traffic keys: send_non_SYN: 00206914, recv_non_SYN: 001df9bc
    RNext key: 1
    Traffic keys: send_non_SYN: 00000000, recv_non_SYN: 00000000
    Last 1 keys used:
        key: 1, time: Jan 23 12:08:12.422, reason: Peer requested rollover

    Peer: 0x7fc2f401f3b8,
    Current key not yet available
    RNext key: 1
    Traffic keys: send_non_SYN: 00000000, recv_non_SYN: 00000000

Total number of Send IDs: 1
Send ID details:
    SendID: 1, Total number of keys: 1
        Keys: 1,
Total number of Receive IDs: 1
Receive ID details:
    ReceiveID: 1, Total number of keys: 1
        Keys: 1,

RP/0/0/CPU0:stoat#
```