LISP Support for TCP Authentication Option

In a LISP deployment, an Egress Tunnel Router (ETR) and a Map Server (MS) exchange LISP control messages through a TCP connection. From Cisco IOS XE Amsterdam 17.2.1, you can use TCP Authentication Option (TCP-AO) to guard against spoofed TCP segments in the sessions between an ETR and an MS.

If you do not configure TCP-AO-based authentication, the TCP segments exchanged between an ETR and an MS are authenticated using a shared key. Cisco IOS XE Amsterdam 17.1.x and earlier releases support only shared-key authentication of TCP messages.

- LISP Support for TCP Authentication Option, on page 1
- How to Configure LISP Support for TCP Authentication Option, on page 2
- Additional References, on page 9

LISP Support for TCP Authentication Option

Overview of LISP Support for TCP Authentication Option

When an ETR detects the first local EID entry, the ETR sends a UDP map-registration message to the MS. MS authenticates the UDP message using a shared authentication key configured on both the ETR and the MS. On successful authentication, the MS creates a TCP listening socket to the ETR for future message exchange.

From Cisco IOS XE Amsterdam 17.2.1, the initial UDP message sent by an ETR also indicates the AO capability.

- If you configure TCP-AO authentication on an ETR, the ETR sets a TCP-AO flag in the UDP message. When the MS detects the TCP-AO flag set in the UDP message, the MS creates the TCP-AO-enabled connection to the ETR. TCP segments that are subsequently exchanged are authenticated using TCP-AO.

  If you do not configure TCP-AO authentication on an ETR, the ETR does not set the TCP-AO flag in the UDP message. When the MS finds that the TCP-AO flag is not set in the UDP message, the MS creates a non-TCP-AO connection to the ETR. TCP messages that are subsequently exchanged are authenticated using the shared key.

- If an ETR is not upgraded to Cisco IOS XE Amsterdam 17.2.1, the ETR sends a UDP message without the TCP-AO flag. When the MS finds no TCP-AO flag in the UDP message, the MS creates a non-TCP-AO connection to the ETR. TCP messages that are subsequently exchanged are authenticated using the shared key.

If you configure peer address locator on a TCP-AO-enabled MS using **map-server session passive-open** [*RLOC*], the MS creates TCP-AO enabled listening sockets. The `accept-ao-mismatch` tcb flag is set to TRUE to maintain backward compatibility with a non-upgraded BR that is not upgraded to Cisco IOS XE Amsterdam 17.2.1 or a later release, or does not have TCP-AO enabled.

Restrictions for LISP Support for TCP Authentication Option

- If you configure TCP-AO authentication on an ETR, but do not configure TCP-AO authentication on the peer MS, a TCP connection is not established between the ETR and the MS. For TCP-AO authentication, you must configure the feature on both the ETR and the MS.
- If you configure or remove TCP-AO authentication for an ETR-MS peer session, the corresponding LISP session flaps as the configuration takes effect.
- You can configure an ETR to use different TCP configurations when communicating with different MSs. When multiple ETRs connect to the same MS, you can configure the ETRs to use different TCP configurations.

  However, on all multi-homing ETRs that are connected to an MS, configure identical TCP-AO behavior. A configuration in which TCP-AO is enabled on some of the multi-homing ETRs connected to the MS but disabled on the others is not supported.

  In a deployment with multiple MSs, a multi-homing ETR can simultaneously have TCP-AO sessions with some of MSs and non-TCP-AO sessions with other MSs. However, all the multi-homing ETRs connected to an MS must have either TCP-AO sessions or non-TCP-AO sessions with the MS.

How to Configure LISP Support for TCP Authentication Option

To establish TCP-AO-based connections between an ETR and an MS, you must configure the following:

1. TCP key-chain and keys on both the ETR and the MS
2. TCP-AO on the MS
3. TCP-AO on the ETR

Configure TCP Key Chain and Keys

Configure TCP-AO key chain and keys on both the peers communicating through a TCP connection.

Note

- 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.
- Do not modify properties of a key in use, except when you need to modify the send-lifetime of the key to trigger rollover. Before modifying properties other than send-lifetime, disassociate the key from the TCP connection.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode. Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **key chain** *key-chain-name* **tcp**

Example:

```
Device(config)# key chain kc1 tcp
```

Creates a TCP-AO key chain of with a specified name and enters the TCP-AO key chain configuration mode.

The key chain name can have a maximum of 256 characters.

Step 4 **key** *key-id*

Example:

```
Device(config-keychain-tcp)# key 10
```

Creates a key with the specified key-id and enters the TCP-AO key chain key configuration mode.

The key-id must be in the range from 0 to 2147483647.

Note
The key-id has only local significance. It is not part of the TCP Authentication Option.

Step 5 **send-id** *send-identifier*

Example:

```
Device(config-keychain-tcp-key)# send-id 218
```

Specifies the send identifier for the key.

The send-identifier must be in the range from 0 to 255.

Step 6 **recv-id** *receiver-identifier*

Example:

```
Device(config-keychain-tcp-key)# recv-id 218
```

Specifies the receive identifier for the key.

The receive-identifier must be in the range from 0 to 255.

Step 7 **cryptographic-algorithm** {**aes-128-cmac** | **hmac-sha-1** | **hmac-sha-256**}

Example:

```
Device(config-keychain-tcp-key)# cryptographic-algorithm hmac-sha-1
```

Specifies the algorithm to be used to compute MACs for TCP segments.

aes-128-cmac	AES-128-CMAC-96: Configures AES-128-CMAC as a cryptographic algorithm with a digest size of 12 bytes.
hmac-sha-1	HMAC-SHA1-96: Configures HMAC-SHA1-96 as a cryptographic algorithm with a digest size of 12 bytes.
hmac-sha-256	HMAC-SHA-256: Configures HMAC-SHA-256 as a cryptographic algorithm with a digest size of 32 bytes.

Step 8 (Optional) **include-tcp-options**

Example:

```
Device(config-keychain-tcp-key)# include-tcp-options
```

This flag indicates whether TCP options other than TCP-AO must be used to calculate MACs.

With the flag enabled, the content of all options, in the order present, is included in the MAC and TCP-AO's MAC field is zero-filled.

When the flag is disabled, all options other than TCP-AO are excluded from MAC calculations.

By default, this flag is disabled.

Step 9 **send-lifetime** [**local**] *start-time* {**infinite** | *end-time* | **duration** *seconds*}

Example:

```
Device(config-keychain-tcp-key)# send-lifetime local 12:00:00 28 Feb 2018 duration 20
```

Specifies the time for which the key is valid to be used for TCP-AO authentication in the send direction.

Use the **local** keyword to specify the start-time in the local time zone. By default, the start-time corresponds to UTC time.

Step 10 **key-string** *master-key*

Example:

```
Device(config-keychain-tcp-key)# key-string abcde
```

Specifies the primary-key for deriving traffic keys.

The primary-keys must be identical on both the peers. If the primary-keys do not match, authentication fails and segments may be rejected by the receiver.

Step 11 (Optional) **accept-ao-mismatch**

Example:

```
Device(config-keychain-tcp-key)# accept-ao-mismatch
```

This flag indicates whether the receiver should accept segments for which the MAC in the incoming TCP AO does not match the MAC generated on the receiver.

Note
Use this configuration with caution. This configuration disables TCP-AO functionality and key rollover on associated connections.

Step 12 **end**

Example:

```
Device(config-keychain-tcp-key)# end
```

Exits TCP-AO key chain key configuration mode and returns to privileged EXEC mode.

Configure TCP Authentication Option on MS

To configure TCP-AO on an MS,

- configure the key chain that the MS uses for TCP-AO authentication of segments received from peer ETRs. Use the command **tcp auth-option** *keychain-name* in the LISP configuration (`config-router-lisp`) mode.

  The MS uses a common key chain for all peer ETRs.

- configure the MS to accept TCP-AO-based connection requests from peer ETRs. Use the **peer accept** command in the TCP-AO(`tcp auth-option`) configuration mode.

- configure the shared authentication key that the MS uses to authenticate the initial UDP message from a peer ETR. The MS authenticates the TCP segments from the ETR using the same key if TCP-AO is not configured on the ETR.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode. Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **router lisp**

Example:

```
Device(config)# router lisp
```

Enters LISP configuration mode.

Step 4 **tcp auth-option** *key-chain*

Example:

```
Device(config-router-lisp)# tcp auth-option kc1
```

Configures the MS to use the specified key chain for TCP-AO and enters the TCP-AO configuration mode.

Step 5 **peer accept**

Example:

```
Device(config-router-lisp-tcp-ao)# peer accept
```

Configures the MS to accept TCP-AO-based connection requests from peer ETRs.

Step 6 **exit**

Example:

```
Device(config-router-lisp-tcp-ao)# exit
```

Exits the TCP-AO configuration mode and returns to LISP configuration mode.

Step 7 **service ipv4**

Example:

```
Device(config-router-lisp)# service ipv4
```

Configures IPv4 as a service type and enters LISP service IPv4 configuration mode.

Step 8 **map-server**

Example:

```
Device(config-lisp-srv-ipv4)# map-server
```

Enables LISP map server functionality for EIDs in the IPv4 address family.

Step 9 **map-resolver**

Example:

```
Device(config-lisp-srv-ipv4)# map-resolver
```

Enables LISP map resolver functionality for EIDs in the IPv4 address family.

Step 10 **exit-service-ipv4**

Example:

```
Device(config-lisp-srv-ipv4)# exit-service-ipv4
```

Exits LISP service IPv4 configuration mode and returns to LISP configuration mode.

Step 11 **site** *site-name*

Example:

```
Device(config-router-lisp)# site site-a
```

Specifies a LISP site named site-a and enters LISP site configuration mode.

Step 12 **eid-record instance-id** *instanace-id EID-prefix*

Example:

```
Device(config-router-lisp-site)# eid-record instance-id 1 172.16.1.0/24
```

Configures an IPv4 prefix associated with this LISP site.

Repeat this step as necessary to configure additional EID prefixes under this LISP sites.

Note
The LISP ETR must be configured with matching EID prefixes.

Step 13 **authentication-key** [*key-type*] *authentication-key*

Example:

```
Device(config-router-lisp-site)# authentication-key some-key
```

Configures the authentication key associated with this site.

Note
Configure an identical authentication key on the LISP ETR.

Step 14 **exit-site**

Example:

```
Device(config-router-lisp-site)# exit-site
```

Exits LISP site configuration mode and returns to LISP configuration mode.

Step 15 Repeat Steps 11 through 14 to configure additional LISP sites.

Configure TCP Authentication Option on ETR

To configure TCP-AO on an ETR,

- configure the key chain that the ETR uses for TCP-AO authentication of segments received from an MS. Use the command **tcp auth-option** *keychain-name* in the LISP configuration (`config-router-lisp`) mode.

  An ETR may use different key chains for different MS peers.

- configure the ETR to establish TCP-AO connection with an MS peer. Use the **peer** *map-server-address* command in the TCP-AO(`tcp auth-option`) configuration mode.

- configure the shared authentication key that the ETR uses to register with the MS.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode. Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **router lisp**

Example:

```
Device(config)# Device lisp
```

Enters LISP configuration mode.

Step 4 **tcp auth-option** *key-chain*

Example:

```
Device(config-router-lisp)# tcp auth-option kc1
```

Configures the ETR to use the specified key chain for TCP-AO and enters the TCP-AO configuration mode.

Step 5 **peer** *map-server-address*

Example:

```
Device(config-router-lisp-tcp-auth-option)# peer 10.10.10.10
```

Configures the ETR to establish a TCP-AO-based connection with the specified MS.

Step 6 **exit**

Example:

```
Device(config-router-lisp-tcp-auth-option)# exit
```

Exits the TCP-AO configuration mode and returns to LISP configuration mode.

Step 7 **service ipv4**

Example:

```
Device(config-router-lisp)# service ipv4
```

Configures IPv4 as a service type and enters LISP service IPv4 configuration mode.

Step 8 **etr**

Example:

```
Device(config-lisp-srv-ipv4)# etr
```

Enables LISP ETR functionality for the IPv4 address family.

Step 9 **map-server** *map-server-address* **key** [*key-type*] *authentication-key*

Example:

```
Device(config-lisp-srv-ipv4)# map-server 10.10.10.10 key some-key
```

Configures a locator address for the LISP map server and an authentication key that this router, acting as an IPv4 LISP ETR, will use to register with the LISP mapping system.

Step 10 **exit-service-ipv4**

Example:

```
Device(config-lisp-srv-ipv4)# exit-service-ipv4
```

Exits LISP service IPv4 configuration mode and returns to LISP configuration mode.

Verifying LISP Support for TCP Authentication Option

Use the command **show lisp vrf default session** *peer-address* to verify that TCP-AO is enabled and the correct TCP key chain is in use.

The following example shows the output of the **show lisp vrf default session** *peer-address* command. The highlighted line is included in the output only if TCP-AO is enabled.

```
Router#show lisp vrf default session 4.4.4.4

Peer address:     4.4.4.4:4342
Local address:    2.2.2.2:34316
Session Type:     Active
Session State:    Up (00:01:12)
Messages in/out:  6/5
Bytes in/out:     245/292
Fatal errors:     0
Rcvd unsupported: 0
Rcvd invalid VRF: 0
Rcvd override:    0
Rcvd malformed:   0
Sent deferred:    0
SSO redundancy:   unsynchronized
Auth type:        TCP-Auth-Option, keychain: kc1

Accepting Users:  0
Users:            6
  Type                      ID                                      In/Out    State
  ETR Reliable Registration lisp 0 IID 102 AFI IPv4                  2/2       TCP
  ETR Reliable Registration lisp 0 IID 108 AFI IPv4                  2/2       TCP
  Capability Exchange       N/A
```

Debugging LISP Support for TCP Authentication Option

You can use the following commands to debug TCP-AO operation with LISP:

- **debug lisp control-plane session**
- **debug ip tcp mkt**
- **debug ip tcp transactions**

Additional References

Related Documents

Related Topic	Document Title
TCP Authentication Option	IP Routing: Protocol-Independent Configuration Guide

Standards and RFCs

Standard/RFC	Title
RFC 5925	The TCP Authentication Option