Stream: Internet Engineering Task Force (IETF)
RFC: 9648
Category: Standards Track
Published: August 2024
ISSN: 2070-1721
Authors: M. Scharf, *Hochschule Esslingen*; M. Jethanandani, *Kloud Services*; V. Murgai, *F5, Inc.*

RFC 9648
A YANG Model for Transmission Control Protocol (TCP) Configuration and State

Abstract

This document specifies a minimal YANG data model for TCP on devices that are configured and managed by network management protocols. The YANG data model defines a container for all TCP connections and groupings of authentication parameters that can be imported and used in TCP implementations or by other models that need to configure TCP parameters. The model also includes basic TCP statistics. The model is compliant with Network Management Datastore Architecture (NMDA) (RFC 8342).

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9648.

Copyright Notice

Table of Contents

1. Introduction

The Transmission Control Protocol (TCP) [RFC9293] is used by many applications in the Internet, including control and management protocols. As such, TCP is implemented on network elements that can be configured and managed via network management protocols such as Network Configuration Protocol (NETCONF) [RFC6241] or RESTCONF [RFC8040].

This document specifies a minimal YANG 1.1 [RFC7950] data model for configuring and managing TCP on network elements that support YANG, a TCP connection table, a TCP listener table containing information about a particular TCP listener, and an augmentation of the YANG data model for key chains [RFC8177] to support authentication. The YANG module specified in this document is compliant with Network Management Datastore Architecture (NMDA) [RFC8342].

The YANG module has a narrow scope and focuses on a subset of fundamental TCP functions and basic statistics. It defines a container for a list of TCP connections that includes definitions from "YANG Groupings for TCP Clients and TCP Servers" [RFC9643]. The model adheres to the recommendation in "BGP/MPLS IP Virtual Private Networks (VPNs)" [RFC4364]. Therefore, it allows enabling of TCP Authentication Option (TCP-AO) [RFC5925] and accommodates the installed base that makes use of MD5. The module can be augmented or updated to address more advanced or implementation-specific TCP features in the future.

This specification does not deprecate the Management Information Base (MIB) for the Transmission Control Protocol (TCP) [RFC4022]. The basic statistics defined in this document follow the model of the TCP MIB. A TCP extended statistics MIB [RFC4898] is also available, but this document does not cover such extended statistics. The YANG module also omits some selected parameters included in TCP MIB, most notably Retransmission Timeout (RTO) configuration and a maximum connection limit. This is a conscious decision as these parameters hardly matter in a state-of-the-art TCP implementation. It would also be possible to translate a MIB into a YANG module, for instance, using "Translation of Structure of Management Information Version 2 (SMIv2) MIB Modules to YANG Modules" [RFC6643]. However, this approach is not used in this document, because a translated model would not be up-to-date.

There are other existing TCP-related YANG data models, which are orthogonal to this specification. Examples are:

- TCP header attributes are modeled in other security-related models, such as those described in "YANG Data Model for Network Access Control Lists (ACLs)" [RFC8519], "Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel Specification" [RFC8783], "I2NSF Capability YANG Data Model" [NSF-CAP-YANG], or "I2NSF Network Security Function-Facing Interface YANG Data Model" [NSF-FACING-YANG].
- TCP-related configuration of a NAT (e.g., NAT44, NAT64, or Destination NAT) is defined in "A YANG Module for Network Address Translation (NAT) and Network Prefix Translation (NPT)" [RFC8512] and "A YANG Data Model for Dual-Stack Lite (DS-Lite)" [RFC8513].

- TCP-AO and TCP MD5 configuration for Layer 3 VPNs is modeled in "A YANG Network Data Model for Layer 3 VPNs" [RFC9182]. This model assumes that TCP-AO-specific parameters are preconfigured in addition to the key chain parameters.

2. Requirements Language

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. YANG Module Overview

3.1. Scope

TCP is implemented on different system architectures. As a result, there are many different and often implementation-specific ways to configure parameters of the TCP engine. In addition, in many TCP/IP stacks, configuration exists for different scopes:

- System-wide configuration: Many TCP implementations have configuration parameters that affect all TCP connections from or to this TCP stack. Typical examples include enabling or disabling optional protocol features. For instance, many implementations can turn on or off use of window scaling the Transmission Control Protocol (TCP) [RFC9293] for all TCP connections.
- Interface configuration: It can be useful to use different TCP parameters on different interfaces, e.g., different device ports or IP interfaces. In that case, TCP parameters can be part of the interface configuration. Typical examples are the Maximum Segment Size (MSS) or configuration related to hardware offloading.
- Connection parameters: Many implementations have means to influence the behavior of each TCP connection, e.g., on the programming interface used by applications. Typical examples are socket options in the socket API, such as disabling the Nagle algorithm (as described in "Transmission Control Protocol (TCP)" [RFC9293]) by TCP_NODELAY. If an application uses such an interface, it is possible that the configuration of the application or application protocol includes TCP-related parameters. An example is the BGP YANG module for service provider networks [BGP-MODEL].
- Application preferences: Setting of TCP parameters can also be part of application preferences, templates, or profiles. An example would be the preferences defined in "An Abstract Application Layer Interface to Transport Services" [TAPS-INTERFACE].

As a result, there is no ground truth for setting certain TCP parameters, and traditionally different TCP implementations have used different modeling approaches. For instance, one implementation may define a given configuration parameter globally, while another one uses per-interface settings, and both approaches work well for the corresponding use cases. Also, different systems may use different default values. In addition, TCP can be implemented in different ways and design choices by the protocol engine often affect configuration options.

Nonetheless, a number of TCP stack parameters require configuration by YANG data models. This document therefore defines a minimal YANG data model with fundamental parameters. An important use case is the TCP configuration on network elements, such as routers, which often use YANG data models. The model therefore specifies TCP parameters that are important on such TCP stacks.

In particular, this applies to the support of the TCP Authentication Option (TCP-AO) [RFC5925] and the corresponding cryptographic algorithms [RFC5926]. TCP-AO is used on routers to secure routing protocols such as BGP. In that case, a YANG data model for TCP-AO configuration is required. The model defined in this document includes the required parameters for TCP-AO configuration, such as the values of SendID and RecvID. The key chain for TCP-AO can be modeled by the YANG data model for key chains [RFC8177]. The groupings defined in this document can be imported and used as part of such a preconfiguration.

Given an installed base, the model also allows enabling of the legacy TCP MD5 [RFC2385] signature option. The TCP MD5 signature option was obsoleted by TCP-AO in 2010. If current implementations require TCP authentication, it is **RECOMMENDED** to use TCP-AO [RFC5925].

Similar to the TCP MIB [RFC4022], this document also specifies basic statistics, a TCP connection list, and a TCP listener list.

- Statistics: Counters for the number of active/passive opens, sent and received TCP segments, errors, and possibly other detailed debugging information.
- TCP connection list: Access to status information for all TCP connections. Note that the connection table is modeled as a list that is read-writeable, even though a connection cannot be created by adding entries to the table. Similarly, deletion of connections from this list is implementation-specific.
- TCP listener list: A list containing information about TCP listeners, i.e., applications willing to accept connections.

This allows implementations of TCP MIB [RFC4022] to migrate to the YANG data model defined in this memo. Note that the TCP MIB does not include means to reset statistics, which are defined in this document. This is not a major addition, as a reset can simply be implemented by storing offset values for the counters.

This version of the module does not model details of Multipath TCP [RFC8684]. This could be addressed in a later version of this document.

3.2. Model Design

The YANG data model defined in this document includes definitions from "YANG Groupings for TCP Clients and TCP Servers" [RFC9643]. Similar to that model, this specification defines YANG groupings. This allows reuse of these groupings in different YANG data models. It is intended that these groupings will be used either standalone or for TCP-based protocols as part of a stack of protocol-specific configuration models. An example could be the one described in "YANG Model for Border Gateway Protocol (BGP-4)" [BGP-MODEL].

3.3. Tree Diagram

This section provides an abridged tree diagram for the YANG module defined in this document. Annotations used in the diagram are defined in "YANG Tree Diagrams" [RFC8340]. A complete tree diagram can be found in Appendix B.

```
module: ietf-tcp
  +--rw tcp!
     +--rw connections
     |     ...
     +--ro tcp-listeners* [type address port]
     |     ...
     +--ro statistics {statistics}?
           ...

  augment /key-chain:key-chains/key-chain:key-chain/key-chain:key:
    +--rw authentication {authentication}?
       +--rw keychain?    key-chain:key-chain-ref
       +--rw (authentication)?
             ...
```

4. TCP YANG Data Model

This YANG module references "The TCP Authentication Option" [RFC5925], "Protection of BGP Sessions via the TCP MD5 Signature Option" [RFC2385], and "Transmission Control Protocol (TCP)" [RFC9293] and imports "Common YANG Data Types" [RFC6991], "Network Configuration Access Control Model" [RFC8341], and "YANG Groupings for TCP Clients and TCP Servers" [RFC9643].

```
<CODE BEGINS> file "ietf-tcp@2022-09-11.yang"

module ietf-tcp {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tcp";
  prefix tcp;

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types.";
  }
  import ietf-tcp-common {
    prefix tcpcmn;
    reference
      "RFC 9643: YANG Groupings for TCP Clients and TCP Servers.";
  }
  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Data Types.";
```

```
  }
  import ietf-netconf-acm {
    prefix nacm;
    reference
      "RFC 8341: Network Configuration Access Control Model.";
  }
  import ietf-key-chain {
    prefix key-chain;
    reference
      "RFC 8177: YANG Data Model for Key Chains.";
  }

  organization
    "IETF TCPM Working Group";

  contact
    "WG Web:   https://datatracker.ietf.org/wg/tcpm/about
     WG List:  TCPM WG <tcpm@ietf.org>

     Authors:  Michael Scharf <michael.scharf@hs-esslingen.de>
               Mahesh Jethanandani <mjethanandani@gmail.com>
               Vishal Murgai <vmurgai@gmail.com>";

  description
    "This module focuses on fundamental TCP functions and basic
     statistics.  The model can be augmented to address more advanced
     or implementation-specific TCP features.

     The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
     NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
     'MAY', and 'OPTIONAL' in this document are to be interpreted as
     described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
     they appear in all capitals, as shown here.

     Copyright (c) 2024 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject to
     the license terms contained in, the Revised BSD License set
     forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (https://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC 9648
     (https://www.rfc-editor.org/info/rfc9648); see the RFC itself
     for full legal notices.";

  revision 2022-09-11 {
    description
      "Initial version.";
    reference
      "RFC 9648: A YANG Model for Transmission Control Protocol (TCP)
                 Configuration and State.";
  }

  // Typedefs
  typedef mss {
```

```
    type uint16;
    description
      "Type definition for the Maximum Segment Size.";
  }

  // Features
  feature statistics {
    description
      "This implementation supports statistics reporting.";
  }

  feature authentication {
    description
      "This implementation supports authentication.";
  }

  // Identities
  identity aes-128 {
    base key-chain:crypto-algorithm;
    description
      "AES128 authentication algorithm used by TCP-AO.";
    reference
      "RFC 5926: Cryptographic Algorithms for the TCP
                 Authentication Option (TCP-AO).";
  }

  // TCP-AO Groupings

  grouping ao {
    leaf send-id {
      type uint8 {
        range "0..max";
      }
      description
        "The SendID is inserted as the KeyID of the TCP-AO option
         of outgoing segments.  In a consistent configuration, the
         SendID matches the RecvID at the other endpoint.";
      reference
        "RFC 5925: The TCP Authentication Option, Section 3.1.";
    }

    leaf recv-id {
      type uint8 {
        range "0..max";
      }
      description
        "The RecvID is matched against the TCP-AO KeyID of incoming
         segments.  In a consistent configuration, the RecvID matches
         the SendID at the other endpoint.";
      reference
        "RFC 5925: The TCP Authentication Option, Section 3.1.";
    }

    leaf include-tcp-options {
      type boolean;
      default "true";
      description
        "When set to true, TCP options are included in the message
```

```
          authentication code (MAC) calculation.";
      reference
        "RFC 5925: The TCP Authentication Option, Section 3.1.";
    }

    leaf accept-key-mismatch {
      type boolean;
      description
        "Accept, when set to true, TCP segments with a Master Key
         Tuple (MKT) that is not configured.";
      reference
        "RFC 5925: The TCP Authentication Option, Section 7.3.";
    }

    leaf r-next-key-id {
      type uint8;
      config false;
      description
        "A field indicating the Master Key Tuple (MKT) that is ready
         at the sender to be used to authenticate received segments,
         i.e., the desired 'receive next' key ID.";
      reference
        "RFC 5925: The TCP Authentication Option.";
    }

    description
      "Authentication Option (AO) for TCP.";
    reference
      "RFC 5925: The TCP Authentication Option.";
  }

  // TCP configuration

  container tcp {
    presence "The container for TCP configuration.";

    description
      "TCP container.";

    container connections {
      list connection {
        key "local-address remote-address local-port remote-port";

        leaf local-address {
          type inet:ip-address;
          description
            "Identifies the address that is used by the local
             endpoint for the connection and is one of the four
             elements that form the connection identifier.";
        }

        leaf remote-address {
          type inet:ip-address;
          description
            "Identifies the address that is used by the remote
             endpoint for the connection and is one of the four
             elements that form the connection identifier.";
        }
```

```
        leaf local-port {
          type inet:port-number;
          description
            "Identifies the local TCP port used for the connection
             and is one of the four elements that form the
             connection identifier.";
        }

        leaf remote-port {
          type inet:port-number;
          description
            "Identifies the remote TCP port used for the connection
             and is one of the four elements that form the
             connection identifier.";
        }

        leaf mss {
          type mss;
          description
            "Maximum Segment Size (MSS) desired on this connection.
             Note that the 'effective send MSS' can be smaller than
             what is configured here.";
          reference
            "RFC 9293: Transmission Control Protocol (TCP).";
        }

        leaf pmtud {
          type boolean;
          default "false";
          description
            "Turns Path Maximum Transmission Unit Discovery (PMTUD)
             on (true) or off (false).";
          reference
            "RFC 9293: Transmission Control Protocol (TCP).";
        }

        uses tcpcmn:tcp-common-grouping;

        leaf state {
          type enumeration {
            enum closed {
              value 1;
              description
                "Connection is closed. Connections in this state
                 may not appear in this list.";
            }
            enum listen {
              value 2;
              description
                "Represents waiting for a connection request from any
                 remote TCP peer and port.";
            }
            enum syn-sent {
              value 3;
              description
                "Represents waiting for a matching connection request
                 after having sent a connection request.";
```

```
            }
            enum syn-received {
              value 4;
              description
                "Represents waiting for a confirming connection
                 request acknowledgement after having both received
                 and sent a connection request.";
            }
            enum established {
              value 5;
              description
                "Represents an open connection; data received can be
                 delivered to the user.  The normal state for the
                 data transfer phase of the connection.";
            }
            enum fin-wait-1 {
              value 6;
              description
                "Represents waiting for a connection termination
                 request from the remote TCP peer or an
                 acknowledgement of the connection termination
                 request previously sent.";
            }
            enum fin-wait-2 {
              value 7;
              description
                "Represents waiting for a connection termination
                 request from the remote TCP peer.";
            }
            enum close-wait {
              value 8;
              description
                "Represents waiting for a connection termination
                 request from the local user.";
            }
            enum last-ack {
              value 9;
              description
                "Represents waiting for an acknowledgement of the
                 connection termination request previously sent to
                 the remote TCP peer (this termination request sent
                 to the remote TCP peer already included an
                 acknowledgement of the termination request sent from
                 the remote TCP peer).";
            }
            enum closing {
              value 10;
              description
                "Represents waiting for a connection termination
                 request acknowledgement from the remote TCP peer.";
            }
            enum time-wait {
              value 11;
              description
                "Represents waiting for enough time to pass to be
                 sure the remote TCP peer received the
                 acknowledgement of its connection termination
                 request and to avoid new connections being impacted
```

```
                by delayed segments from previous connections.";
            }
          }
          config false;
          description
            "The state of this TCP connection.";
        }
        description
          "List of TCP connections with their parameters.

           The list is modeled as writeable even though only some of
           the nodes are writeable, e.g., keepalive.  Connections
           that are created and match this list SHOULD apply the
           writeable parameters.  At the same time, implementations
           may not allow creation of new TCP connections simply by
           adding entries to the list.  Furthermore, the behavior
           upon removal is implementation-specific.  Implementations
           may not support closing or resetting a TCP connection
           upon an operation that removes the entry from the list.

           The operational state of this list SHOULD reflect
           connections that have configured but not created and
           connections that have been created.  Connections in the
           CLOSED state are not reflected on this list.";
      }
      description
        "A container of all TCP connections.";
    }

    list tcp-listeners {
      key "type address port";
      config false;

      description
        "A table containing information about a particular
         TCP listener.";

      leaf type {
        type inet:ip-version;
        description
          "The address type of address.  The value
           should be unspecified (0) if connection initiations
           to all local IP addresses are accepted.";
      }

      leaf address {
        type union {
          type inet:ip-address;
          type string {
            length "0";
          }
        }
        description
          "The local IP address for this TCP connection.

           The value of this node can be represented in three
           possible ways, depending on the characteristics of the
           listening application:
```

```
            1. For an application willing to accept both IPv4 and
               IPv6 datagrams, the value of this node must be
               ''h (a zero-length octet string), with the value
               of the corresponding 'type' object being
               unspecified (0).

            2. For an application willing to accept only IPv4 or
               IPv6 datagrams, the value of this node must be
               '0.0.0.0' or '::' respectively, with
               'type' representing the appropriate address type.

            3. For an application that is listening for data
               destined only to a specific IP address, the value
               of this node is the specific local address, with
               'type' representing the appropriate address type.";
        }

        leaf port {
          type inet:port-number;
          description
            "The local port number for this TCP connection.";
        }
      }

      container statistics {
        if-feature "statistics";
        config false;

        leaf active-opens {
          type yang:counter64;
          description
            "The number of times that TCP connections have made a
             direct transition to the SYN-SENT state from the CLOSED
             state.";
          reference
            "RFC 9293: Transmission Control Protocol (TCP).";
        }

        leaf passive-opens {
          type yang:counter64;
          description
            "The number of times TCP connections have made a direct
             transition to the SYN-RCVD state from the LISTEN state.";
          reference
            "RFC 9293: Transmission Control Protocol (TCP).";
        }

        leaf attempt-fails {
          type yang:counter64;
          description
            "The number of times that TCP connections have made a
             direct transition to the CLOSED state from either the
             SYN-SENT state or the SYN-RCVD state, plus the number of
             times that TCP connections have made a direct transition
             to the LISTEN state from the SYN-RCVD state.";
          reference
            "RFC 9293: Transmission Control Protocol (TCP).";
```

```
      }

      leaf establish-resets {
        type yang:counter64;
        description
          "The number of times that TCP connections have made a
           direct transition to the CLOSED state from either the
           ESTABLISHED state or the CLOSE-WAIT state.";
        reference
          "RFC 9293: Transmission Control Protocol (TCP).";
      }

      leaf currently-established {
        type yang:gauge32;
        description
          "The number of TCP connections for which the current state
           is either ESTABLISHED or CLOSE-WAIT.";
        reference
          "RFC 9293: Transmission Control Protocol (TCP).";
      }

      leaf in-segments {
        type yang:counter64;
        description
          "The total number of TCP segments received, including those
           received in error.  This count includes TCP segments
           received on currently established connections.";
        reference
          "RFC 9293: Transmission Control Protocol (TCP).";
      }

      leaf out-segments {
        type yang:counter64;
        description
          "The total number of TCP segments sent, including those on
           current connections but excluding those containing only
           retransmitted octets.";
        reference
          "RFC 9293: Transmission Control Protocol (TCP).";
      }

      leaf retransmitted-segments {
        type yang:counter64;
        description
          "The total number of TCP segments retransmitted; that is,
           the number of TCP segments transmitted containing one or
           more previously transmitted octets.";
        reference
          "RFC 9293: Transmission Control Protocol (TCP).";
      }

      leaf in-errors {
        type yang:counter64;
        description
          "The total number of TCP segments received in error
           (e.g., bad TCP checksums).";
        reference
          "RFC 9293: Transmission Control Protocol (TCP).";
```

```
      }

      leaf out-resets {
        type yang:counter64;
        description
          "The number of TCP segments sent containing the RST flag.";
        reference
          "RFC 9293: Transmission Control Protocol (TCP).";
      }

      leaf auth-failures {
        if-feature "authentication";
        type yang:counter64;
        description
          "The number of times that authentication has failed either
           with TCP-AO or MD5.";
      }

      action reset {
        nacm:default-deny-all;
        description
          "Reset statistics action command.";
        input {
          leaf reset-at {
            type yang:date-and-time;
            description
              "Time when the reset action needs to be
               executed.";
          }
        }
        output {
          leaf reset-finished-at {
            type yang:date-and-time;
            description
              "Time when the reset action command completed.";
          }
        }
      }
      description
        "Statistics across all connections.";
    }
  }

  augment "/key-chain:key-chains/key-chain:key-chain/key-chain:key" {
    description
      "Augmentation of the key-chain model to add TCP-AO and TCP-MD5
       authentication.";

    container authentication {
      if-feature "authentication";
      leaf keychain {
        type key-chain:key-chain-ref;
        description
          "Reference to the key chain that will be used by
           this model.  Applicable for TCP-AO and TCP-MD5
           only.";
        reference
          "RFC 8177: YANG Data Model for Key Chains.";
```

```
      }

      choice authentication {
        container ao {
          presence "Presence container for all TCP-AO related"
                 + " configuration";
          uses ao;
          description
            "Use TCP-AO to secure the connection.";
        }

        container md5 {
          presence "Presence container for all MD5 related"
                 + " configuration";
          description
            "Use TCP-MD5 to secure the connection.  As the TCP MD5
             signature option is obsoleted by TCP-AO, it is
             RECOMMENDED to use TCP-AO instead.";
          reference
            "RFC 2385: Protection of BGP Sessions via the TCP MD5
                       Signature Option.";
        }
        description
          "Choice of TCP authentication.";
      }
      description
        "Authentication definitions for TCP configuration.
         This includes parameters such as how to secure the
         connection, which can be part of either the client
         or server.";
    }
  }
}

<CODE ENDS>
```

5. IANA Considerations

5.1. The IETF XML Registry

IANA has registered the following URI in the "ns" registry defined in the "IETF XML Registry" [RFC3688].

URI: urn:ietf:params:xml:ns:yang:ietf-tcp
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

5.2. The YANG Module Names Registry

IANA has registered the following in the "YANG Module Names" registry created by "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)" [RFC6020].

Name: ietf-tcp
Namespace: urn:ietf:params:xml:ns:yang:ietf-tcp
Prefix: tcp
Reference: RFC 9648

The registration is not maintained by IANA.

6. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., "config true", which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

- Common configuration included from NETCONF client and server models [RFC9643]. Unrestricted access to all the nodes, e.g., keepalive idle timer, can cause connections to fail or to timeout prematurely.
- Authentication configuration. Unrestricted access to the nodes under authentication configuration can prevent the use of authenticated communication and cause connection setups to fail. This can result in massive security vulnerabilities and service disruption for the traffic requiring authentication.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

- Unrestricted access to connection information of the client or server can be used by a malicious user to launch an attack.
- Similarly, unrestricted access to statistics of the client or server can be used by a malicious user to exploit any vulnerabilities of the system.

Some of the RPC operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. These are the operations and their sensitivity/vulnerability:

- The YANG module allows for the statistics to be cleared by executing the reset action. This action should be restricted to users with the right permission.

The module specified in this document supports MD5 to basically accommodate the installed BGP base. MD5 suffers from the security weaknesses discussed in Section 2 of [RFC6151] or Section 2.1 of [RFC6952].

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2385] Heffernan, A., "Protection of BGP Sessions via the TCP MD5 Signature Option", RFC 2385, DOI 10.17487/RFC2385, August 1998, <https://www.rfc-editor.org/info/rfc2385>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, DOI 10.17487/RFC5925, June 2010, <https://www.rfc-editor.org/info/rfc5925>.

[RFC5926] Lebovitz, G. and E. Rescorla, "Cryptographic Algorithms for the TCP Authentication Option (TCP-AO)", RFC 5926, DOI 10.17487/RFC5926, June 2010, <https://www.rfc-editor.org/info/rfc5926>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8177] Lindem, A., Ed., Qu, Y., Yeung, D., Chen, I., and J. Zhang, "YANG Data Model for Key Chains", RFC 8177, DOI 10.17487/RFC8177, June 2017, <https://www.rfc-editor.org/info/rfc8177>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC9293] Eddy, W., Ed., "Transmission Control Protocol (TCP)", STD 7, RFC 9293, DOI 10.17487/RFC9293, August 2022, <https://www.rfc-editor.org/info/rfc9293>.

[RFC9643] Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", RFC 9643, DOI 10.17487/RFC9643, May 2024, <https://www.rfc-editor.org/info/rfc9643>.

7.2. Informative References

[BGP-MODEL] Jethanandani, M., Patel, K., Hares, S., and J. Haas, "YANG Model for Border Gateway Protocol (BGP-4)", Work in Progress, Internet-Draft, draft-ietf-idr-bgp-model-17, 5 July 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-idr-bgp-model-17>.

[NSF-CAP-YANG] Hares, S., Ed., Jeong, J., Ed., Kim, J., Moskowitz, R., and Q. Lin, "I2NSF Capability YANG Data Model", Work in Progress, Internet-Draft, draft-ietf-i2nsf-capability-data-model-32, 23 May 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-i2nsf-capability-data-model-32>.

[NSF-FACING-YANG] Kim, J., Ed., Jeong, J., Ed., Park, J., Hares, S., and Q. Lin, "I2NSF Network Security Function-Facing Interface YANG Data Model", Work in Progress, Internet-Draft, draft-ietf-i2nsf-nsf-facing-interface-dm-29, 1 June 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-i2nsf-nsf-facing-interface-dm-29>.

[RFC4022] Raghunarayan, R., Ed., "Management Information Base for the Transmission Control Protocol (TCP)", RFC 4022, DOI 10.17487/RFC4022, March 2005, <https://www.rfc-editor.org/info/rfc4022>.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <https://www.rfc-editor.org/info/rfc4364>.

[RFC4898] Mathis, M., Heffner, J., and R. Raghunarayan, "TCP Extended Statistics MIB", RFC 4898, DOI 10.17487/RFC4898, May 2007, <https://www.rfc-editor.org/info/rfc4898>.

[RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, DOI 10.17487/RFC6151, March 2011, <https://www.rfc-editor.org/info/rfc6151>.

[RFC6643] Schoenwaelder, J., "Translation of Structure of Management Information Version 2 (SMIv2) MIB Modules to YANG Modules", RFC 6643, DOI 10.17487/RFC6643, July 2012, <https://www.rfc-editor.org/info/rfc6643>.

[RFC6952] Jethanandani, M., Patel, K., and L. Zheng, "Analysis of BGP, LDP, PCEP, and MSDP Issues According to the Keying and Authentication for Routing Protocols (KARP) Design Guide", RFC 6952, DOI 10.17487/RFC6952, May 2013, <https://www.rfc-editor.org/info/rfc6952>.

[RFC8512] Boucadair, M., Ed., Sivakumar, S., Jacquenet, C., Vinapamula, S., and Q. Wu, "A YANG Module for Network Address Translation (NAT) and Network Prefix Translation (NPT)", RFC 8512, DOI 10.17487/RFC8512, January 2019, <https://www.rfc-editor.org/info/rfc8512>.

[RFC8513] Boucadair, M., Jacquenet, C., and S. Sivakumar, "A YANG Data Model for Dual-Stack Lite (DS-Lite)", RFC 8513, DOI 10.17487/RFC8513, January 2019, <https://www.rfc-editor.org/info/rfc8513>.

[RFC8519] Jethanandani, M., Agarwal, S., Huang, L., and D. Blair, "YANG Data Model for Network Access Control Lists (ACLs)", RFC 8519, DOI 10.17487/RFC8519, March 2019, <https://www.rfc-editor.org/info/rfc8519>.

[RFC8684] Ford, A., Raiciu, C., Handley, M., Bonaventure, O., and C. Paasch, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 8684, DOI 10.17487/RFC8684, March 2020, <https://www.rfc-editor.org/info/rfc8684>.

[RFC8783] Boucadair, M., Ed. and T. Reddy.K, Ed., "Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel Specification", RFC 8783, DOI 10.17487/RFC8783, May 2020, <https://www.rfc-editor.org/info/rfc8783>.

[RFC9182] Barguil, S., Gonzalez de Dios, O., Ed., Boucadair, M., Ed., Munoz, L., and A. Aguado, "A YANG Network Data Model for Layer 3 VPNs", RFC 9182, DOI 10.17487/RFC9182, February 2022, <https://www.rfc-editor.org/info/rfc9182>.

[RFC9235] Touch, J. and J. Kuusisaari, "TCP Authentication Option (TCP-AO) Test Vectors", RFC 9235, DOI 10.17487/RFC9235, May 2022, <https://www.rfc-editor.org/info/rfc9235>.

[TAPS-INTERFACE] Trammell, B., Ed., Welzl, M., Ed., Enghardt, R., Fairhurst, G., Kühlewind, M., Perkins, C., Tiesel, P., and T. Pauly, "An Abstract Application Layer Interface to Transport Services", Work in Progress, Internet-Draft, draft-ietf-taps-interface-26, 16 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-taps-interface-26>.

Appendix A. Examples

A.1. Keepalive Configuration

This particular example demonstrates how a particular connection can be configured for keepalives.

```
NOTE: '\' line wrapping per RFC 8792

<?xml version="1.0" encoding="UTF-8"?>
<!--
This example shows how TCP keepalive, MSS, and PMTU can be configure\
d for a given connection. An idle connection is dropped after
idle-time + (max-probes * probe-interval).
-->
<tcp
    xmlns="urn:ietf:params:xml:ns:yang:ietf-tcp">
  <connections>
    <connection>
      <local-address>192.0.2.1</local-address>
      <remote-address>192.0.2.2</remote-address>
      <local-port>1025</local-port>
      <remote-port>22</remote-port>
      <mss>1400</mss>
      <pmtud>true</pmtud>
      <keepalives>
        <idle-time>5</idle-time>
        <max-probes>5</max-probes>
        <probe-interval>10</probe-interval>
      </keepalives>
    </connection>
  </connections>
</tcp>
```

A.2. TCP-AO Configuration

The following example demonstrates how to model a TCP-AO [RFC5925] configuration for the example in "TCP Authentication Option (TCP-AO) Test Vectors" [RFC9235]. The IP addresses and other parameters are taken from the test vectors.

```
NOTE: '\' line wrapping per RFC 8792

<?xml version="1.0" encoding="UTF-8"?>
<!--
This example sets TCP-AO configuration parameters similarly to
the examples in RFC 9235.
-->

<key-chains
    xmlns="urn:ietf:params:xml:ns:yang:ietf-key-chain">
  <key-chain>
    <name>ao-config</name>
    <description>"An example for TCP-AO configuration."</description>
    <key>
      <key-id>55</key-id>
      <lifetime>
        <send-lifetime>
          <start-date-time>2017-01-01T00:00:00Z</start-date-time>
          <end-date-time>2017-02-01T00:00:00Z</end-date-time>
        </send-lifetime>
        <accept-lifetime>
          <start-date-time>2016-12-31T23:59:55Z</start-date-time>
          <end-date-time>2017-02-01T00:00:05Z</end-date-time>
        </accept-lifetime>
      </lifetime>
      <crypto-algorithm
          xmlns:tcp=
          "urn:ietf:params:xml:ns:yang:ietf-tcp">tcp:aes-128</crypto\
-algorithm>
      <key-string>
        <keystring>testvector</keystring>
      </key-string>
      <authentication
          xmlns="urn:ietf:params:xml:ns:yang:ietf-tcp">
        <keychain>ao-config</keychain>
        <ao>
          <send-id>61</send-id>
          <recv-id>84</recv-id>
        </ao>
      </authentication>
    </key>
  </key-chain>
</key-chains>
```

Appendix B. Complete Tree Diagram

Here is the complete tree diagram for the TCP YANG data model.

```
module: ietf-tcp
  +--rw tcp!
     +--rw connections
     |  +--rw connection*
     |          [local-address remote-address local-port remote-port]
     |     +--rw local-address     inet:ip-address
     |     +--rw remote-address    inet:ip-address
     |     +--rw local-port        inet:port-number
     |     +--rw remote-port       inet:port-number
     |     +--rw mss?              mss
     |     +--rw pmtud?            boolean
     |     +--rw keepalives! {keepalives-supported}?
     |     |  +--rw idle-time         uint16
     |     |  +--rw max-probes        uint16
     |     |  +--rw probe-interval    uint16
     |     +--ro state?            enumeration
     +--ro tcp-listeners* [type address port]
     |  +--ro type       inet:ip-version
     |  +--ro address    union
     |  +--ro port       inet:port-number
     +--ro statistics {statistics}?
        +--ro active-opens?             yang:counter64
        +--ro passive-opens?            yang:counter64
        +--ro attempt-fails?            yang:counter64
        +--ro establish-resets?         yang:counter64
        +--ro currently-established?    yang:gauge32
        +--ro in-segments?              yang:counter64
        +--ro out-segments?             yang:counter64
        +--ro retransmitted-segments?   yang:counter64
        +--ro in-errors?                yang:counter64
        +--ro out-resets?               yang:counter64
        +--ro auth-failures?            yang:counter64
        |       {authentication}?
        +---x reset
           +---w input
           |  +---w reset-at?   yang:date-and-time
           +--ro output
              +--ro reset-finished-at?   yang:date-and-time

  augment /key-chain:key-chains/key-chain:key-chain/key-chain:key:
    +--rw authentication {authentication}?
       +--rw keychain?    key-chain:key-chain-ref
       +--rw (authentication)?
          +--:(ao)
          |  +--rw ao!
          |     +--rw send-id?               uint8
          |     +--rw recv-id?               uint8
          |     +--rw include-tcp-options?   boolean
          |     +--rw accept-key-mismatch?   boolean
          |     +--ro r-next-key-id?         uint8
```

```
            +--:(md5)
               +--rw md5!
```

Acknowledgements

Michael Scharf was supported by the StandICT.eu project, which is funded by the European Commission under the Horizon 2020 Programme.

The following persons have contributed to this document by reviews (in alphabetical order): Mohamed Boucadair, Gorry Fairhurst, Jeffrey Haas, and Tom Petch.

Authors' Addresses

Michael Scharf
Hochschule Esslingen - University of Applied Sciences
Kanalstr. 33
73728 Esslingen
Germany
Email: michael.scharf@hs-esslingen.de

Mahesh Jethanandani
Kloud Services
Email: mjethanandani@gmail.com

Vishal Murgai
F5, Inc.
Email: vmurgai@gmail.com