NETMOD WG   M. Jethanandani
Internet-Draft   Cisco Systems, Inc
Intended status: Standards Track   L. Huang
Expires: April 6, 2018   General Electric
                                                               S. Agarwal
                                                        Cisco Systems, Inc.
                                                                 D. Blair
                                                        Cisco Systems, INc
                                                        October 03, 2017

Network Access Control List (ACL) YANG Data Model
draft-ietf-netmod-acl-model-14

Abstract

   This document describes a data model of Access Control List (ACL)
   basic building blocks.

   Editorial Note (To be removed by RFC Editor)

   This draft contains many placeholder values that need to be replaced
   with finalized values at the time of publication. This note
   summarizes all of the substitutions that are needed. Please note
   that no other RFC Editor instructions are specified anywhere else in
   this document.

   Artwork in this document contains shorthand references to drafts in
   progress. Please apply the following replacements

   o "XXXX" --> the assigned RFC value for this draft both in this
      draft and in the YANG models under the revision statement.

   o Revision date in model needs to get updated with the date the
      draft gets approved. The date also needs to get reflected on the
      line with <CODE BEGINS>.

Status of This Memo

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."

This Internet-Draft will expire on April 6, 2018.

Copyright Notice

Table of Contents

1. Introduction

Access Control List (ACL) is one of the basic elements used to configure device forwarding behavior. It is used in many networking technologies such as Policy Based Routing, Firewalls etc.

An ACL is an ordered set of rules that is used to filter traffic on a networking device. Each rule is represented by an Access Control Entry (ACE).

Each ACE has a group of match criteria and a group of action criteria.

The match criteria consist of a tuple of packet header match criteria and can have metadata match criteria as well.

o Packet header matches apply to fields visible in the packet such as address or class of service or port numbers.

o In case vendor supports it, metadata matches apply to fields associated with the packet but not in the packet header such as input interface or overall packet length

The actions specify what to do with the packet when the matching criteria is met. These actions are any operations that would apply to the packet, such as counting, policing, or simply forwarding.The list of potential actions is endless depending on the capabilities of the networked devices.

Access Control List is also widely knowns as ACL (pronounce as [ak-uh l]) or Access List. In this document, Access Control List, ACL and Access List are used interchangeably.

The matching of filters and actions in an ACE/ACL are triggered only after application/attachment of the ACL to an interface, VRF, vty/tty session, QoS policy, routing protocols amongst various other config attachment points. Once attached, it is used for filtering traffic using the match criteria in the ACE's and taking appropriate action(s) that have been configured against that ACE. In order to apply an ACL to any attachment point, vendors would have to augment the ACL YANG model.

1.1. Definitions and Acronyms

ACE: Access Control Entry

ACL: Access Control List

   DSCP: Differentiated Services Code Point

   ICMP: Internet Control Message Protocol

   IP: Internet Protocol

   IPv4: Internet Protocol version 4

   IPv6: Internet Protocol version 6

   MAC: Media Access Control

   TCP: Transmission Control Protocol

2. Problem Statement

   This document defines a YANG [RFC6020] data model for the
   configuration of ACLs. It is very important that model can be easily
   used by applications/attachments.

   ACL implementations in every device may vary greatly in terms of the
   filter constructs and actions that they support. Therefore this
   draft proposes a model that can be augmented by standard extensions
   and vendor proprietary models.

3. Understanding ACL's Filters and Actions

   Although different vendors have different ACL data models, there is a
   common understanding of what access control list (ACL) is. A network
   system usually have a list of ACLs, and each ACL contains an ordered
   list of rules, also known as access list entries - ACEs. Each ACE
   has a group of match criteria and a group of action criteria. The
   match criteria consist of packet header matching. It as also
   possible for ACE to match on metadata, if supported by the vendor.
   Packet header matching applies to fields visible in the packet such
   as address or class of service or port numbers. Metadata matching
   applies to fields associated with the packet, but not in the packet
   header such as input interface, packet length, or source or
   destination prefix length. The actions can be any sort of operation
   from logging to rate limiting or dropping to simply forwarding.
   Actions on the first matching ACE are applied with no processing of
   subsequent ACEs.

   The model also includes a container to hold overall operational state
   for each ACL and operational state for each ACE. One ACL can be
   applied to multiple targets within the device, such as interfaces of
   a networked device, applications or features running in the device,
   etc. When applied to interfaces of a networked device, the ACL is

   applied in a direction which indicates if it should be applied to
   packet entering (input) or leaving the device (output). An example
   in the appendix shows how to express it in YANG model.

   This draft tries to address the commonalities between all vendors and
   create a common model, which can be augmented with proprietary
   models. The base model is simple and with this design we hope to
   achieve enough flexibility for each vendor to extend the base model.
   The use of feature statements in the document allows vendors to
   advertise match rules they support.

3.1. ACL Modules

   There are two YANG modules in the model. The first module, "ietf-
   access-control-list", defines generic ACL aspects which are common to
   all ACLs regardless of their type or vendor. In effect, the module
   can be viewed as providing a generic ACL "superclass". It imports
   the second module, "ietf-packet-fields". The match container in
   "ietf-access-control-list" uses groupings in "ietf-packet-fields".
   The combination of if-feature checks and must statements allow for
   the selection of relevant match fields that a user can define rules
   for.

   If there is a need to define new "matches" choice, such as IPFIX
   [RFC5101], the container "matches" can be augmented.

   For a reference to the annotations used in the diagram below, see
   YANG Tree Diagrams [I-D.ietf-netmod-yang-tree-diagrams].

```
 module: ietf-access-control-list
    +--rw access-lists
       +--rw acl* [acl-type acl-name]
       |  +--rw acl-name    string
       |  +--rw acl-type    acl-type
       |  +--rw aces
       |     +--rw ace* [rule-name]
       |        +--rw rule-name           string
       |        +--rw matches
       |        |  +--rw l2-acl {l2-acl}?
       |        |  |  +--rw destination-mac-address?       yang:mac-ad
 dress
       |        |  |  +--rw destination-mac-address-mask?   yang:mac-ad
 dress
       |        |  |  +--rw source-mac-address?            yang:mac-ad
 dress
       |        |  |  +--rw source-mac-address-mask?       yang:mac-ad
 dress
       |        |  |  +--rw ethertype?                     eth:etherty
```

```
pe
        |           |  +--rw ipv4-acl {ipv4-acl}?
        |           |  |  +--rw dscp?                       inet:dscp
        |           |  |  +--rw ecn?                        uint8
        |           |  |  +--rw length?                     uint16
        |           |  |  +--rw ttl?                        uint8
        |           |  |  +--rw protocol?                   uint8
        |           |  |  +--rw source-port-range!
        |           |  |  |  +--rw lower-port    inet:port-number
        |           |  |  |  +--rw upper-port?   inet:port-number
        |           |  |  |  +--rw operation?    operator
        |           |  |  +--rw destination-port-range!
        |           |  |  |  +--rw lower-port    inet:port-number
        |           |  |  |  +--rw upper-port?   inet:port-number
        |           |  |  |  +--rw operations?   operator
        |           |  |  +--rw ihl?                        uint8
        |           |  |  +--rw flags?                      bits
        |           |  |  +--rw offset?                     uint16
        |           |  |  +--rw identification?             uint16
        |           |  |  +--rw destination-ipv4-network?   inet:ipv4-prefi
  x
        |           |  |  +--rw source-ipv4-network?        inet:ipv4-prefi
  x
        |           |  +--rw ipv6-acl {ipv6-acl}?
        |           |  |  +--rw dscp?                       inet:dscp
        |           |  |  +--rw ecn?                        uint8
        |           |  |  +--rw length?                     uint16
        |           |  |  +--rw ttl?                        uint8
        |           |  |  +--rw protocol?                   uint8
        |           |  |  +--rw source-port-range!
        |           |  |  |  +--rw lower-port    inet:port-number
        |           |  |  |  +--rw upper-port?   inet:port-number
        |           |  |  |  +--rw operation?    operator
        |           |  |  +--rw destination-port-range!
        |           |  |  |  +--rw lower-port    inet:port-number
        |           |  |  |  +--rw upper-port?   inet:port-number
        |           |  |  |  +--rw operations?   operator
        |           |  |  +--rw next-header?                uint8
        |           |  |  +--rw destination-ipv6-network?   inet:ipv6-prefi
  x
        |           |  |  +--rw source-ipv6-network?        inet:ipv6-prefi
  x
        |           |  |  +--rw flow-label?                 inet:ipv6-flow-
  label
        |           |  +--rw l2-l3-ipv4-acl {mixed-ipv4-acl}?
        |           |  |  +--rw destination-mac-address?        yang:mac-ad
  dress
        |           |  |  +--rw destination-mac-address-mask?   yang:mac-ad
```

```
 dress
        |            |  |  +--rw source-mac-address?             yang:mac-ad
 dress
        |            |  |  +--rw source-mac-address-mask?        yang:mac-ad
 dress
        |            |  |  +--rw ethertype?                      eth:etherty
 pe
        |            |  |  +--rw dscp?                           inet:dscp
        |            |  |  +--rw ecn?                            uint8
        |            |  |  +--rw length?                         uint16
        |            |  |  +--rw ttl?                            uint8
        |            |  |  +--rw protocol?                       uint8
        |            |  |  +--rw source-port-range!
        |            |  |  |  +--rw lower-port    inet:port-number
        |            |  |  |  +--rw upper-port?   inet:port-number
        |            |  |  |  +--rw operation?    operator
        |            |  |  +--rw destination-port-range!
        |            |  |  |  +--rw lower-port    inet:port-number
        |            |  |  |  +--rw upper-port?   inet:port-number
        |            |  |  |  +--rw operations?   operator
        |            |  |  +--rw ihl?                            uint8
        |            |  |  +--rw flags?                          bits
        |            |  |  +--rw offset?                         uint16
        |            |  |  +--rw identification?                 uint16
        |            |  |  +--rw destination-ipv4-network?       inet:ipv4-p
   refix
        |            |  |  +--rw source-ipv4-network?            inet:ipv4-p
   refix
        |            |  +--rw l2-l3-ipv6-acl {mixed-ipv6-acl}?
        |            |  |  +--rw destination-mac-address?        yang:mac-ad
 dress
        |            |  |  +--rw destination-mac-address-mask?   yang:mac-ad
 dress
        |            |  |  +--rw source-mac-address?             yang:mac-ad
 dress
        |            |  |  +--rw source-mac-address-mask?        yang:mac-ad
 dress
        |            |  |  +--rw ethertype?                      eth:etherty
 pe
        |            |  |  +--rw dscp?                           inet:dscp
        |            |  |  +--rw ecn?                            uint8
        |            |  |  +--rw length?                         uint16
        |            |  |  +--rw ttl?                            uint8
        |            |  |  +--rw protocol?                       uint8
        |            |  |  +--rw source-port-range!
        |            |  |  |  +--rw lower-port    inet:port-number
        |            |  |  |  +--rw upper-port?   inet:port-number
        |            |  |  |  +--rw operation?    operator
```

```
|            |  |   +--rw destination-port-range!
|            |  |   |  +--rw lower-port    inet:port-number
|            |  |   |  +--rw upper-port?   inet:port-number
|            |  |   |  +--rw operations?   operator
|            |  |   +--rw next-header?                  uint8
|            |  |   +--rw destination-ipv6-network?     inet:ipv6-p
   refix
|            |  |   +--rw source-ipv6-network?          inet:ipv6-p
   refix
|            |  |   +--rw flow-label?
|            |  |          inet:ipv6-flow-label
|            |  +--rw l2-l3-ipv4-ipv6-acl {l2-l3-ipv4-ipv6-acl}?
|            |  |   +--rw destination-mac-address?      yang:mac-ad
   dress
|            |  |   +--rw destination-mac-address-mask?  yang:mac-ad
   dress
|            |  |   +--rw source-mac-address?           yang:mac-ad
   dress
|            |  |   +--rw source-mac-address-mask?      yang:mac-ad
   dress
|            |  |   +--rw ethertype?                    eth:etherty
   pe
|            |  |   +--rw dscp?                         inet:dscp
|            |  |   +--rw ecn?                          uint8
|            |  |   +--rw length?                       uint16
|            |  |   +--rw ttl?                          uint8
|            |  |   +--rw protocol?                     uint8
|            |  |   +--rw source-port-range!
|            |  |   |  +--rw lower-port    inet:port-number
|            |  |   |  +--rw upper-port?   inet:port-number
|            |  |   |  +--rw operation?    operator
|            |  |   +--rw destination-port-range!
|            |  |   |  +--rw lower-port    inet:port-number
|            |  |   |  +--rw upper-port?   inet:port-number
|            |  |   |  +--rw operations?   operator
|            |  |   +--rw ihl?                          uint8
|            |  |   +--rw flags?                        bits
|            |  |   +--rw offset?                       uint16
|            |  |   +--rw identification?               uint16
|            |  |   +--rw destination-ipv4-network?     inet:ipv4-p
   refix
|            |  |   +--rw source-ipv4-network?          inet:ipv4-p
   refix
|            |  |   +--rw next-header?                  uint8
|            |  |   +--rw destination-ipv6-network?     inet:ipv6-p
   refix
|            |  |   +--rw source-ipv6-network?          inet:ipv6-p
   refix
```

```
        |          |  |  +--rw flow-label?
        |          |  |          inet:ipv6-flow-label
        |          |  +--rw tcp-acl {tcp-acl}?
        |          |  |  +--rw sequence-number?          uint32
        |          |  |  +--rw acknowledgement-number?   uint32
        |          |  |  +--rw data-offset?              uint8
        |          |  |  +--rw reserved?                 uint8
        |          |  |  +--rw flags?                    bits
        |          |  |  +--rw window-size?              uint16
        |          |  |  +--rw urgent-pointer?           uint16
        |          |  |  +--rw options?                  uint32
        |          |  +--rw udp-acl {udp-acl}?
        |          |  |  +--rw length?   uint16
        |          |  +--rw icmp-acl {icmp-acl}?
        |          |  |  +--rw type?           uint8
        |          |  |  +--rw code?           uint8
        |          |  |  +--rw rest-of-header?   uint32
        |          |  +--rw any-acl! {any-acl}?
        |          |  +--rw interface?           if:interface-ref
        |          +--rw actions
        |          |      {acl-aggregate-stats or interface-acl-aggregate
}?
        |          |  +--rw forwarding    identityref
        |          |  +--rw logging?      identityref
        |          |  +--rw icmp-off?     boolean
        |          +--ro matched-packets?   yang:counter64
        |          +--ro matched-octets?    yang:counter64
        +--rw interfaces
           +--rw interface* [interface-id]
              +--rw interface-id    if:interface-ref
              +--rw ingress
              |  +--rw acl-sets
              |     +--rw acl-set* [set-name type]
              |        +--rw set-name    -> ../../../../../../acl/acl-na
me
              |        +--rw type        -> ../../../../../../acl/acl-ty
pe
              |        +--rw ace* [rule-name]
              |                {interface-stats or interface-acl-aggrega
te}?
              |           +--rw rule-name          leafref
              |           +--ro matched-packets?   yang:counter64
              |           +--ro matched-octets?    yang:counter64
              +--rw egress
                 +--rw acl-sets
                    +--rw acl-set* [set-name type]
                       +--rw set-name    -> ../../../../../../acl/acl-na
me
```

```
                          +--rw type          -> ../../../../../../acl/acl-ty
 pe
                          +--rw ace* [rule-name]
                                  {interface-stats or interface-acl-aggrega
 te}?
                             +--rw rule-name         leafref
                             +--ro matched-packets?   yang:counter64
                             +--ro matched-octets?    yang:counter64
```

4. ACL YANG Models

4.1. IETF Access Control List module

"ietf-access-control-list" is the standard top level module for
access lists. The "access-lists" container stores a list of "acl".
Each "acl" has information identifying the access list by a
name("acl-name") and a list("access-list-entries") of rules
associated with the "acl-name". Each of the entries in the
list("access-list-entries"), indexed by the string "rule-name", has
containers defining "matches" and "actions".

The model uses defines several ACL types in the form of identities
and features. Features are used by implementors to select the ACL
types the system can support. These types are implicitly inherited
by the "ace", thus safeguarding against misconfiguration of "ace"
types in an "acl".

The "matches" define criteria used to identify patterns in "ietf-
packet-fields". The "actions" define behavior to undertake once a
"match" has been identified. In addition to permit and deny for
actions, a logging option allows for a match to be logged that can be
used to determine which rule was matched upon. The model also
defines the ability for ACL's to be attached to a particular
interface.

Statistics in the ACL can be collected for an "ace" or for an
"interface". The feature statements defined for statistics can be
used to determine whether statistics are being collected per "ace",
per "interface" or both.

<CODE BEGINS> file "ietf-access-control-list@2017-10-03.yang"

```
module ietf-access-control-list {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-access-control-list";
  prefix acl;

  import ietf-yang-types {
```

```
  prefix yang;
}

import ietf-packet-fields {
  prefix packet-fields;
}

import ietf-interfaces {
  prefix if;
}

organization
  "IETF NETMOD (NETCONF Data Modeling Language)
   Working Group";

contact
  "WG Web: http://tools.ietf.org/wg/netmod/
   WG List: netmod@ietf.org

   Editor: Mahesh Jethanandani
           mjethanandani@gmail.com
   Editor: Lisa Huang
           lyihuang16@gmail.com
   Editor: Sonal Agarwal
           sagarwal12@cisco.com
   Editor: Dana Blair
           dblair@cisco.com";

description
  "This YANG module defines a component that describe the
   configuration of Access Control Lists (ACLs).

   Copyright (c) 2017 IETF Trust and the persons identified as
   the document authors.  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 Simplified BSD
   License set forth in Section 4.c of the IETF Trust's Legal
   Provisions Relating to IETF Documents
   (http://trustee.ietf.org/license-info).

   This version of this YANG module is part of RFC XXXX; see
   the RFC itself for full legal notices.";

revision 2017-10-03 {
  description
    "Added feature and identity statements for different types
     of rule matches. Split the matching rules based on the
```

```
      feature statement and added a must statement within
      each container.";
   reference
     "RFC XXX: Network Access Control List (ACL) YANG Data Model.";
 }

 /*
  * Identities
  */

 /*
  * Forwarding actions for a packet
  */
 identity forwarding-action {
   description
     "Base identity for actions in the forwarding category";
 }

 identity accept {
   base forwarding-action;
   description
     "Accept the packet";
 }

 identity drop {
   base forwarding-action;
   description
     "Drop packet without sending any ICMP error message";
 }

 identity reject {
   base forwarding-action;
   description
     "Drop the packet and send an ICMP error message to the source";
 }

 /*
  * Logging actions for a packet
  */
 identity log-action {
   description
     "Base identity for defining the destination for logging actions";
 }

 identity log-syslog {
   base log-action;
   description
     "System log (syslog) the information for the packet";
```

```
  }

  identity log-none {
    base log-action;
    description
      "No logging for the packet";
  }

  identity acl-base {
    description
      "Base Access Control List type for all Access Control List type
       identifiers.";
  }

  identity ipv4-acl {
    base acl:acl-base;
    description
      "ACL that primarily matches on fields from the IPv4 header
       (e.g. IPv4 destination address) and layer 4 headers (e.g. TCP
       destination port).  An acl of type ipv4-acl does not contain
       matches on fields in the ethernet header or the IPv6 header.";
  }

  identity ipv6-acl {
    base acl:acl-base;
    description
      "ACL that primarily matches on fields from the IPv6 header
       (e.g. IPv6 destination address) and layer 4 headers (e.g. TCP
       destination port). An acl of type ipv6-acl does not contain
       matches on fields in the ethernet header or the IPv4 header.";
  }

  identity eth-acl {
    base acl:acl-base;
    description
      "ACL that primarily matches on fields in the ethernet header,
       like 10/100/1000baseT or WiFi Access Control List. An acl of
       type eth-acl does not contain matches on fields in the IPv4
       header, IPv6 header or layer 4 headers.";
  }

  identity mixed-l2-l3-ipv4-acl {
    base "acl:acl-base";

    description
      "ACL that contains a mix of entries that
       primarily match on fields in ethernet headers,
       entries that primarily match on IPv4 headers.
```

```
         Matching on layer 4 header fields may also exist in the
         list.";
   }

   identity mixed-l2-l3-ipv6-acl {
     base "acl:acl-base";

     description
       "ACL that contains a mix of entries that
        primarily match on fields in ethernet headers, entries
        that primarily match on fields in IPv6 headers. Matching on
        layer 4 header fields may also exist in the list.";
   }

   identity mixed-l2-l3-ipv4-ipv6-acl {
     base "acl:acl-base";

     description
       "ACL that contains a mix of entries that
        primarily match on fields in ethernet headers, entries
        that primarily match on fields in IPv4 headers, and entries
        that primarily match on fields in IPv6 headers. Matching on
        layer 4 header fields may also exist in the list.";
   }

   identity any-acl {
     base "acl:acl-base";

     description
       "ACL that can contain any pattern to match upon";
   }

   /*
    * Features
    */
   feature l2-acl {
     description
       "Layer 2 ACL supported";
   }

   feature ipv4-acl {
     description
       "Layer 3 IPv4 ACL supported";
   }

   feature ipv6-acl {
     description
       "Layer 3 IPv6 ACL supported";
```

```
  }

  feature mixed-ipv4-acl {
    description
      "Layer 2 and Layer 3 IPv4 ACL supported";
  }

  feature mixed-ipv6-acl {
    description
      "Layer 2 and Layer 3 IPv6 ACL supported";
  }

  feature l2-l3-ipv4-ipv6-acl {
    description
      "Layer 2 and any Layer 3 ACL supported.";
  }

  feature tcp-acl {
    description
      "TCP header ACL supported.";
  }

  feature udp-acl {
    description
      "UDP header ACL supported.";
  }

  feature icmp-acl {
    description
      "ICMP header ACL supported.";
  }

  feature any-acl {
    description
     "ACL for any pattern.";
  }

  /*
   * Stats Features
   */
  feature interface-stats {
    description
      "ACL counters are available and reported only per interface";
  }

  feature acl-aggregate-stats {
    description
      "ACL counters are aggregated over all interfaces, and reported
```

```
          only per ACL entry";
    }

    feature interface-acl-aggregate {
      description
        "ACL counters are reported per interface, and also aggregated
         and reported per ACL entry";
    }

    /*
     * Typedefs
     */
    typedef acl-type {
      type identityref {
        base acl-base;
      }
      description
        "This type is used to refer to an Access Control List
         (ACL) type";
    }

    typedef acl-ref {
      type leafref {
        path "/access-lists/acl/acl-name";
      }
      description
        "This type is used by data models that need to reference an
         Access Control List";
    }

    grouping interface-acl {
      description
        "Grouping for per-interface ingress ACL data";

      container acl-sets {
        description
          "Enclosing container the list of ingress ACLs on the
           interface";

        list acl-set {
          key "set-name type";
          ordered-by user;
          description
            "List of ingress ACLs on the interface";

          leaf set-name {
            type leafref {
              path "../../../../../../acl/acl-name";
```

```
        }
        description
          "Reference to the ACL set name applied on ingress";
      }

      leaf type {
        type leafref {
          path "../../../../../../acl/acl-type";
        }
        description
          "Reference to the ACL set type applied on ingress";
      }

      list ace {
        if-feature "interface-stats or interface-acl-aggregate";
        key "rule-name";
        description
          "List of access list entries(ACE)";
        leaf rule-name {
          type leafref {
            path "../../../../../../../acl/aces/ace/rule-name";
          }
          description
            "The ace rule-name";
        }
        uses acl-counters;
      }
    }
  }
}

grouping acl-counters {
  description
    "Common grouping for ACL counters";

  leaf matched-packets {
    type yang:counter64;
    config false;
    description
      "Count of the number of packets matching the current ACL
       entry.

       An implementation should provide this counter on a
       per-interface per-ACL-entry if possible.

       If an implementation only supports ACL counters per entry
       (i.e., not broken out per interface), then the value
       should be equal to the aggregate count across all interfaces.
```

```
         An implementation that provides counters per entry per
         interface is not required to also provide an aggregate count,
         e.g., per entry -- the user is expected to be able implement
         the required aggregation if such a count is needed.";
    }

    leaf matched-octets {
      type yang:counter64;
      config false;
      description
        "Count of the number of octets (bytes) matching the current
         ACL entry.

         An implementation should provide this counter on a
         per-interface per-ACL-entry if possible.

         If an implementation only supports ACL counters per entry
         (i.e., not broken out per interface), then the value
         should be equal to the aggregate count across all interfaces.

         An implementation that provides counters per entry per
         interface is not required to also provide an aggregate count,
         e.g., per entry -- the user is expected to be able implement
         the required aggregation if such a count is needed.";
    }
  }

  /*
   * Configuration data nodes
   */
  container access-lists {
    description
      "This is a top level container for Access Control Lists.
       It can have one or more Access Control Lists.";
    list acl {
      key "acl-type acl-name";
      description
        "An Access Control List(ACL) is an ordered list of
         Access List Entries (ACE). Each Access Control Entry has a
         list of match criteria and a list of actions.
         Since there are several kinds of Access Control Lists
         implemented with different attributes for
         different vendors, this
         model accommodates customizing Access Control Lists for
         each kind and for each vendor.";
      leaf acl-name {
        type string {
          length "1..64";
```

```
          }
          description
            "The name of access-list. A device MAY restrict the length
             and value of this name, possibly space and special
             characters are not allowed.";
        }
        leaf acl-type {
          type acl-type;
          description
            "Type of access control list. Indicates the primary intended
             type of match criteria (e.g. ethernet, IPv4, IPv6, mixed,
             etc) used in the list instance.";
        }
        container aces {
          description
            "The access-list-entries container contains
             a list of access-list-entries(ACE).";
          list ace {
            key "rule-name";
            ordered-by user;
            description
              "List of access list entries(ACE)";
            leaf rule-name {
              type string {
                length "1..64";
              }
              description
                "A unique name identifying this Access List
                 Entry(ACE).";
            }

            container matches {
              description
                "The rules in this set determine what fields will be
                 matched upon before any action is taken on them.
                 The rules are selected based on the feature set
                 defined by the server and the acl-type defined.";

              container l2-acl {
                if-feature l2-acl;
                must "derived-from(../../../../acl-type, 'acl:eth-acl')";
                uses packet-fields:acl-eth-header-fields;
                description
                  "Rule set for L2 ACL.";
              }

              container ipv4-acl {
                if-feature ipv4-acl;
```

```
                 must "derived-from(../../../../acl-type, " +
                    "'acl:ipv4-acl')";
              uses packet-fields:acl-ip-header-fields;
                   uses packet-fields:acl-ipv4-header-fields;
              description
                "Rule set that supports IPv4 headers.";
            }

            container ipv6-acl {
              if-feature ipv6-acl;
              must "derived-from(../../../../acl-type, " +
                   "'acl:ipv6-acl')";
              uses packet-fields:acl-ip-header-fields;
              uses packet-fields:acl-ipv6-header-fields;
              description
                "Rule set that supports IPv6 headers.";
            }

            container l2-l3-ipv4-acl {
              if-feature mixed-ipv4-acl;
              must "derived-from(../../../../acl-type, " +
                   "'acl:mixed-l2-l3-ipv4-acl')";
              uses packet-fields:acl-eth-header-fields;
              uses packet-fields:acl-ip-header-fields;
              uses packet-fields:acl-ipv4-header-fields;
              description
                "Rule set that is a logical AND (&&) of l2
                 and ipv4 headers.";
            }

            container l2-l3-ipv6-acl {
              if-feature mixed-ipv6-acl;
              must "derived-from(../../../../acl-type, " +
                   "'acl:mixed-l2-l3-ipv6-acl')";
              uses packet-fields:acl-eth-header-fields;
              uses packet-fields:acl-ip-header-fields;
              uses packet-fields:acl-ipv6-header-fields;
              description
                "Rule set that is a logical AND (&&) of L2
                 && IPv6 headers.";
            }

            container l2-l3-ipv4-ipv6-acl {
              if-feature l2-l3-ipv4-ipv6-acl;
              must "derived-from(../../../../acl-type, " +
                   "'acl:mixed-l2-l3-ipv4-ipv6-acl')";
              uses packet-fields:acl-eth-header-fields;
              uses packet-fields:acl-ip-header-fields;
```

```
                 uses packet-fields:acl-ipv4-header-fields;
                 uses packet-fields:acl-ipv6-header-fields;
                 description
                   "Rule set that is a logical AND (&&) of L2
                    && IPv4 && IPv6 headers.";
               }

               container tcp-acl {
                 if-feature tcp-acl;
                 uses packet-fields:acl-tcp-header-fields;
                 description
                   "Rule set that defines TCP headers.";
               }

               container udp-acl {
                 if-feature udp-acl;
                 uses packet-fields:acl-udp-header-fields;
                 description
                   "Rule set that defines UDP headers.";
               }

               container icmp-acl {
                 if-feature icmp-acl;
                 uses packet-fields:acl-icmp-header-fields;
                 description
                   "Rule set that defines ICMP headers.";
               }

               container any-acl {
                 if-feature any-acl;
                 must "derived-from(../../../../acl-type, 'acl:any-acl')";
                 presence "Matches any";
                 description
                   "Rule set that allows for a any ACL.";
               }

               leaf interface {
                 type if:interface-ref;
                 description
                   "Interface name that is specified to
                    match upon.";
               }
             }

             container actions {
               if-feature "acl-aggregate-stats or interface-acl-aggregate";
               description
                 "Definitions of action criteria for this ace entry";
```

```
                leaf forwarding {
                  type identityref {
                    base forwarding-action;
                  }
                  mandatory true;
                  description
                    "Specifies the forwarding action per ace entry";
                }

                leaf logging {
                  type identityref {
                    base log-action;
                  }
                  default log-none;
                  description
                  "Specifies the log action and destination for
                   matched packets. Default value is not to log the
                   packet.";
                }

                leaf icmp-off {
                  type boolean;
                  default "false";
                  description
                  "true indicates ICMP errors will never be generated
                   in response to an ICMP error message. false indicates
                   ICMP error will be generated.";
                }
            }
            uses acl-counters;
          }
        }
      }
      container interfaces {
        description
          "Enclosing container for the list of interfaces on which
           ACLs are set";

        list interface {
          key "interface-id";
            description
              "List of interfaces on which ACLs are set";

          leaf interface-id {
            type if:interface-ref;
            description
              "Reference to the interface id list key";
          }
```

```
        container ingress {
          uses interface-acl;
          description
            "The ACL's applied to ingress interface";
        }
        container egress {
          uses interface-acl;
          description
            "The ACL's applied to egress interface";
        }
      }
    }
  }
}
```

<CODE ENDS>

4.2. IETF Packet Fields module

   The packet fields module defines the necessary groups for matching on
   fields in the packet including ethernet, ipv4, ipv6, and transport
   layer fields. The 'acl-type' node determines which of these fields
   get included for any given ACL with the exception of TCP, UDP and
   ICMP header fields. Those fields can be used in conjunction with any
   of the above layer 2 or layer 3 fields.

   Since the number of match criteria is very large, the base draft does
   not include these directly but references them by "uses" to keep the
   base module simple. In case more match conditions are needed, those
   can be added by augmenting choices within container "matches" in
   ietf-access-control-list.yang model.

<CODE BEGINS> file "ietf-packet-fields@2017-10-03.yang"

```
module ietf-packet-fields {
  namespace "urn:ietf:params:xml:ns:yang:ietf-packet-fields";
  prefix packet-fields;

  import ietf-inet-types {
    prefix inet;
  }

  import ietf-yang-types {
    prefix yang;
  }

  import ietf-ethertypes {
    prefix eth;
```

```
  }

  organization
    "IETF NETMOD (NETCONF Data Modeling Language) Working
     Group";

  contact
    "WG Web: http://tools.ietf.org/wg/netmod/
     WG List: netmod@ietf.org

     Editor: Mahesh Jethanandani
             mjethanandani@gmail.com
     Editor: Lisa Huang
             lyihuang16@gmail.com
     Editor: Sonal Agarwal
             agarwaso@cisco.com
     Editor: Dana Blair
             dblair@cisco.com";

  description
    "This YANG module defines groupings that are used by
     ietf-access-control-list YANG module. Their usage is not
     limited to ietf-access-control-list and can be
     used anywhere as applicable.
     Copyright (c) 2017 IETF Trust and the persons identified as
     the document authors.  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 Simplified BSD
     License set forth in Section 4.c of the IETF Trust's Legal
     Provisions Relating to IETF Documents
     (http://trustee.ietf.org/license-info).
     This version of this YANG module is part of RFC XXXX; see
     the RFC itself for full legal notices.";

  revision 2017-10-03 {
    description
      "Added header fields for TCP, UDP, and ICMP.";
    reference
      "RFC XXX: Network Access Control List (ACL) YANG Data Model.";
  }

  /*
   * Typedefs
   */
  typedef operator {
    type enumeration {
```

```
      enum lt {
        description
          "Less than.";
      }
      enum gt {
        description
          "Greater than.";
      }
      enum eq {
        description
          "Equal to.";
      }
      enum neq {
        description
          "Not equal to.";
      }
    }
    description
      "The source and destination port range definitions
       can be further qualified using an operator. An
       operator is needed only if lower-port is specified
       and upper-port is not specified. The operator
       therefore further qualifies lower-port only.";
  }

  grouping acl-transport-header-fields {
    description
      "Transport header fields";
    container source-port-range {
      presence "Enables setting source port range";
      description
        "Inclusive range representing source ports to be used.
         When only lower-port is present, it represents a single
         port and eq operator is assumed to be default.

         When both lower-port and upper-port are specified,
         it implies a range inclusive of both values.

         If no port is specified, 'any' (wildcard) is assumed.";

      leaf lower-port {
        type inet:port-number;
        mandatory true;
        description
          "Lower boundary for port.";
      }
      leaf upper-port {
        type inet:port-number;
```

```
        must ". >= ../lower-port" {
          error-message
          "The upper-port must be greater than or equal
           to lower-port";
        }
        description
          "Upper boundary for port. If it exists, the upper port
           must be greater or equal to lower-port.";
      }
      leaf operation {
        type operator;
        must "(../lower-port and not(../upper-port))" {
          error-message
            "If lower-port is specified, and an operator is also
             specified, then upper-port should not be specified.";
          description
            "If lower-port is specified, and an operator is also
             specified, then upper-port should not be specified.";
        }
        default eq;
        description
          "Operator to be applied on the lower-port.";
      }
    }

    container destination-port-range {
      presence "Enables setting destination port range";
      description
        "Inclusive range representing destination ports to be used.
         When only lower-port is present, it represents a single
         port and eq operator is assumed to be default.

         When both lower-port and upper-port are specified,
         it implies a range inclusive of both values.

         If no port is specified, 'any' (wildcard) is assumed. ";

      leaf lower-port {
        type inet:port-number;
        mandatory true;
        description
          "Lower boundary for port.";
      }
      leaf upper-port {
        type inet:port-number;
        must ". >= ../lower-port" {
          error-message
            "The upper-port must be greater than or equal
```

```
              to lower-port";
        }
        description
          "Upper boundary for port. If existing, the upper port must
           be greater or equal to lower-port";
      }
      leaf operations {
        type operator;
        must "(../lower-port and not(../upper-port))" {
          error-message
            "If lower-port is specified, and an operator is also
             specified, then upper-port should not be specified.";
          description
            "If lower-port is specified, and an operator is also
             specified, then upper-port should not be specified.";
        }
        default eq;
        description
          "Operator to be applied on the lower-port.";
      }
    }
  }
}

grouping acl-ip-header-fields {
  description
    "IP header fields common to ipv4 and ipv6";
  reference
    "RFC 791.";

  leaf dscp {
    type inet:dscp;
    description
      "Differentiated Services Code Point.";
    reference
      "RFC 2474: Definition of Differentiated services field
       (DS field) in the IPv4 and IPv6 headers.";
  }

  leaf ecn {
    type uint8 {
      range 0..3;
    }
    description
      "Explicit Congestion Notification.";
    reference
      "RFC 3168.";
  }
```

```
    leaf length {
      type uint16;
      description
        "In IPv4 header field, this field is known as the Total Length.
         Total Length is the length of the datagram, measured in octets,
         including internet header and data.

         In IPv6 header field, this field is known as the Payload
         Length, the length of the IPv6 payload, i.e. the rest of
         the packet following the IPv6 header, in octets.";
      reference
        "RFC 719, RFC 2460";
    }

    leaf ttl {
      type uint8;
      description
        "This field indicates the maximum time the datagram is allowed
         to remain in the internet system.  If this field contains the
         value zero, then the datagram must be destroyed.

         In IPv6, this field is known as the Hop Limit.";
      reference "RFC 719, RFC 2460";
    }

    leaf protocol {
      type uint8;
      description
        "Internet Protocol number.";
    }
    uses acl-transport-header-fields;
  }

  grouping acl-ipv4-header-fields {
    description
      "Fields in IPv4 header.";

    leaf ihl {
      type uint8 {
        range "5..60";
      }
      description
        "An IPv4 header field, the Internet Header Length (IHL) is
         the length of the internet header in 32 bit words, and
         thus points to the beginning of the data. Note that the
         minimum value for a correct header is 5.";
    }
```

```
    leaf flags {
      type bits {
        bit reserved {
          position 0;
          description
            "Reserved. Must be zero.";
        }
        bit fragment {
          position 1;
          description
            "Setting value to 0 indicates may fragment, while setting
             the value to 1 indicates do not fragment.";
        }
        bit more {
          position 2;
          description
            "Setting the value to 0 indicates this is the last fragment,
             and setting the value to 1 indicates more fragments are
             coming.";
        }
      }
      description
        "Bit definitions for the flags field in IPv4 header.";
    }

    leaf offset {
      type uint16 {
        range "20..65535";
      }
      description
        "The fragment offset is measured in units of 8 octets (64 bits).
         The first fragment has offset zero. The length is 13 bits";
    }

    leaf identification {
      type uint16;
      description
        "An identifying value assigned by the sender to aid in
         assembling the fragments of a datagram.";
    }

    leaf destination-ipv4-network {
      type inet:ipv4-prefix;
      description
        "Destination IPv4 address prefix.";
    }
    leaf source-ipv4-network {
      type inet:ipv4-prefix;
```

```
      description
        "Source IPv4 address prefix.";
    }
  }

  grouping acl-ipv6-header-fields {
    description
      "Fields in IPv6 header";

    leaf next-header {
      type uint8;
      description
        "Identifies the type of header immediately following the
         IPv6 header. Uses the same values as the IPv4 Protocol
         field.";
      reference
        "RFC 2460";
    }

    leaf destination-ipv6-network {
      type inet:ipv6-prefix;
      description
        "Destination IPv6 address prefix.";
    }

    leaf source-ipv6-network {
      type inet:ipv6-prefix;
      description
        "Source IPv6 address prefix.";
    }

    leaf flow-label {
      type inet:ipv6-flow-label;
      description
        "IPv6 Flow label.";
    }
    reference
      "RFC 4291: IP Version 6 Addressing Architecture
       RFC 4007: IPv6 Scoped Address Architecture
       RFC 5952: A Recommendation for IPv6 Address Text
                 Representation";
  }

  grouping acl-eth-header-fields {
    description
      "Fields in Ethernet header.";

    leaf destination-mac-address {
```

```
          type yang:mac-address;
          description
            "Destination IEEE 802 MAC address.";
        }
        leaf destination-mac-address-mask {
          type yang:mac-address;
          description
            "Destination IEEE 802 MAC address mask.";
        }
        leaf source-mac-address {
          type yang:mac-address;
          description
            "Source IEEE 802 MAC address.";
        }
        leaf source-mac-address-mask {
          type yang:mac-address;
          description
            "Source IEEE 802 MAC address mask.";
        }
        leaf ethertype {
          type eth:ethertype;
          description
            "The Ethernet Type (or Length) value represented
             in the canonical order defined by IEEE 802.
             The canonical representation uses lowercase
             characters.";
          reference
            "IEEE 802-2014 Clause 9.2";
        }
        reference
          "IEEE 802: IEEE Standard for Local and Metropolitan
           Area Networks: Overview and Architecture.";
      }

      grouping acl-tcp-header-fields {
        description
          "Collection of TCP header fields that can be used to
           setup a match filter.";

        leaf sequence-number {
          type uint32;
          description
            "Sequence number that appears in the packet.";
        }

        leaf acknowledgement-number {
          type uint32;
          description
```

```
        "The acknowledgement number that appears in the
         packet.";
    }

    leaf data-offset {
      type uint8 {
        range "5..15";
      }
      description
        "Specifies the size of the TCP header in 32-bit
         words. The minimum size header is 5 words and
         the maximum is 15 words thus giving the minimum
         size of 20 bytes and maximum of 60 bytes,
         allowing for up to 40 bytes of options in the
         header.";
    }

    leaf reserved {
      type uint8;
      description
        "Reserved for future use.";
    }

    leaf flags {
      type bits {
        bit ns {
          position 0;
          description
            "ECN-nonce concealment protection";
          reference "RFC 3540).";
        }
        bit cwr {
          position 1;
          description
            "Congestion Window Reduced (CWR) flag is set by
             the sending host to indicate that it received
             a TCP segment with the ECE flag set and had
             responded in congestion control mechanism.";
          reference "RFC 3168";
        }
        bit ece {
          position 2;
          description
            "ECN-Echo has a dual role, depending on the value
             of the SYN flag. It indicates:
             If the SYN flag is set (1), that the TCP peer is ECN
             capable. If the SYN flag is clear (0), that a packet
             with Congestion Experienced flag set (ECN=11) in IP
```

```
            header was received during normal transmission
            (added to header by RFC 3168). This serves as an
            indication of network congestion (or impending
            congestion) to the TCP sender.";
      }
      bit urg {
        position 3;
        description
          "Indicates that the Urgent pointer field is significant.";
      }
      bit ack {
        position 4;
        description
          "Indicates that the Acknowledgment field is significant.
           All packets after the initial SYN packet sent by the
           client should have this flag set.";
      }
      bit psh {
        position 5;
        description
          "Push function. Asks to push the buffered data to the
           receiving application.";
      }
      bit rst {
        position 6;
        description
          "Reset the connection.";
      }
      bit syn {
        position 7;
        description
          "Synchronize sequence numbers. Only the first packet
           sent from each end should have this flag set. Some
           other flags and fields change meaning based on this
           flag, and some are only valid for when it is set,
           and others when it is clear.";
      }
      bit fin {
        position 8;
        description
          "Last package from sender.";
      }
    }
    description
      "Also known as Control Bits. Contains 9 1-bit flags.";
  }

  leaf window-size {
```

```
      type uint16;
      description
        "The size of the receive window, which specifies
         the number of window size units (by default,
         bytes) (beyond the segment identified by the
         sequence number in the acknowledgment field)
         that the sender of this segment is currently
         willing to receive.";
    }

    leaf urgent-pointer {
      type uint16;
      description
        "This field is an offset from the sequence number
         indicating the last urgent data byte.";
    }

    leaf options {
      type uint32;
      description
        "The length of this field is determined by the
         data offset field. Options have up to three
         fields: Option-Kind (1 byte), Option-Length
         (1 byte), Option-Data (variable). The Option-Kind
         field indicates the type of option, and is the
         only field that is not optional. Depending on
         what kind of option we are dealing with,
         the next two fields may be set: the Option-Length
         field indicates the total length of the option,
         and the Option-Data field contains the value of
         the option, if applicable.";
    }
  }

  grouping acl-udp-header-fields {
    description
      "Collection of UDP header fields that can be used
       to setup a match filter.";

    leaf length {
      type uint16;
      description
        "A field that specifies the length in bytes of
         the UDP header and UDP data. The minimum
         length is 8 bytes because that is the length of
         the header. The field size sets a theoretical
         limit of 65,535 bytes (8 byte header + 65,527
         bytes of data) for a UDP datagram. However the
```

```
            actual limit for the data length, which is
            imposed by the underlying IPv4 protocol, is
            65,507 bytes (65,535 minus 8 byte UDP header
            minus 20 byte IP header).

            In IPv6 jumbograms it is possible to have
            UDP packets of size greater than 65,535 bytes.
            RFC 2675 specifies that the length field is set
            to zero if the length of the UDP header plus
            UDP data is greater than 65,535.";
      }
    }

    grouping acl-icmp-header-fields {
      description
        "Collection of ICMP header fields that can be
         used to setup a match filter.";

      leaf type {
        type uint8;
        description
          "Also known as Control messages.";
        reference "RFC 792";
      }

      leaf code {
        type uint8;
        description
          "ICMP subtype. Also known as Control messages.";
      }

      leaf rest-of-header {
        type uint32;
        description
          "Four-bytes field, contents vary based on the
           ICMP type and code.";
      }
    }
}

<CODE ENDS>
```

4.3. An ACL Example

   Requirement: Deny tcp traffic from 10.10.10.1/24, destined to
   11.11.11.1/24.

   Here is the acl configuration xml for this Access Control List:

```
<?xml version='1.0' encoding='UTF-8'?>
  <data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <access-lists xmlns="urn:ietf:params:xml:ns:yang:
     ietf-access-control-list">
      <acl>
        <acl-name>sample-ipv4-acl</acl-name>
        <acl-type>ipv4-acl</acl-type>
        <aces>
          <ace>
            <rule-name>rule1</rule-name>
            <matches>
              <ipv4-acl>
                <protocol>tcp</protocol>
                <destination-ipv4-network>
                  11.11.11.1/24
                </destination-ipv4-network>
                <source-ipv4-network>
                  10.10.10.1/24
                </source-ipv4-network>
              </ipv4-acl>
            </matches>
            <actions>
              <packet-handling>deny</packet-handling>
            </actions>
          </ace>
        </aces>
      </acl>
    </access-lists>
  </data>
```

The acl and aces can be described in CLI as the following:

```
access-list ipv4 sample-ipv4-acl
deny tcp 10.10.10.1/24 11.11.11.1/24
```

4.4. Port Range Usage Example

When a lower-port and an upper-port are both present, it represents a
range between lower-port and upper-port with both the lower-port and
upper-port are included. When only a lower-port presents, it
represents a single port.

With the follow XML snippet:

```
        <source-port-range>
          <lower-port>16384</lower-port>
          <upper-port>16387</upper-port>
        </source-port-range>
```

This represents source ports 16384,16385, 16386, and 16387.

With the follow XML snippet:

```
        <source-port-range>
          <lower-port>16384</lower-port>
          <upper-port>65535</upper-port>
        </source-port-range>
```

This represents source ports greater than/equal to 16384 and less than equal to 65535.

With the follow XML snippet:

```
        <source-port-range>
          <lower-port>21</lower-port>
        </source-port-range>
```

This represents port 21.

With the following XML snippet, the configuration is specifying all ports that are not equal to 21.

```
        <source-port-range>
          <lower-port>21</lower-port>
          <operations>neq</operations>
        </source-port-range>
```


5. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242]. The NETCONF Access Control Model (NACM [RFC6536]) provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

There are a number of data nodes defined in the YANG module which 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:

/access-lists/acl/access-list-entries: This list specifies all the
configured access list entries on the device. Unauthorized write
access to this list can allow intruders to access and control the
system. Unauthorized read access to this list can allow intruders to
spoof packets with authorized addresses thereby compromising the
system.

6. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688].
Following the format in RFC 3688, the following registration is
requested to be made:

URI: urn:ietf:params:xml:ns:yang:ietf-access-control-list

URI: urn:ietf:params:xml:ns:yang:ietf-packet-fields

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

This document registers a YANG module in the YANG Module Names
registry [RFC6020].

name: ietf-access-control-list namespace:
urn:ietf:params:xml:ns:yang:ietf-access-control-list prefix: ietf-acl
reference: RFC XXXX

name: ietf-packet-fields namespace: urn:ietf:params:xml:ns:yang:ietf-
packet-fields prefix: ietf-packet-fields reference: RFC XXXX

7. Acknowledgements

Alex Clemm, Andy Bierman and Lisa Huang started it by sketching out
an initial IETF draft in several past IETF meetings. That draft
included an ACL YANG model structure and a rich set of match filters,
and acknowledged contributions by Louis Fourie, Dana Blair, Tula
Kraiser, Patrick Gili, George Serpa, Martin Bjorklund, Kent Watsen,
and Phil Shafer. Many people have reviewed the various earlier
drafts that made the draft went into IETF charter.

Dean Bogdanovic, Kiran Agrahara Sreenivasa, Lisa Huang, and Dana
Blair each evaluated the YANG model in previous drafts separately,
and then worked together to created a ACL draft that was supported by
different vendors. That draft removed vendor specific features, and
gave examples to allow vendors to extend in their own proprietary
ACL. The earlier draft was superseded with this updated draft and
received more participation from many vendors.

Authors would like to thank Jason Sterne, Lada Lhotka, Juergen
Schoenwalder, David Bannister, and Jeff Haas for their review of and
suggestions to the draft.

8. Open Issues

o The current model does not support the concept of "containers"
   used to contain multiple addresses per rule entry.

9. References

9.1. Normative References

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

[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>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration
           Protocol (NETCONF) Access Control Model", RFC 6536,
           DOI 10.17487/RFC6536, March 2012,
           <https://www.rfc-editor.org/info/rfc6536>.

9.2. Informative References

[I-D.ietf-netmod-yang-tree-diagrams]
          Bjorklund, M. and L. Berger, "YANG Tree Diagrams", draft-
          ietf-netmod-yang-tree-diagrams-01 (work in progress), June
          2017.

[RFC5101] Claise, B., Ed., "Specification of the IP Flow Information
          Export (IPFIX) Protocol for the Exchange of IP Traffic
          Flow Information", RFC 5101, DOI 10.17487/RFC5101, January
          2008, <https://www.rfc-editor.org/info/rfc5101>.

Appendix A. Extending ACL model examples

A.1. Example of extending existing model for route filtering

With proposed modular design, it is easy to extend the model with
other features. Those features can be standard features, like route
filters. Route filters match on specific IP addresses or ranges of
prefixes. Much like ACLs, they include some match criteria and
corresponding match action(s). For that reason, it is very simple to
extend existing ACL model with route filtering. The combination of a
route prefix and prefix length along with the type of match
determines how route filters are evaluated against incoming routes.
Different vendors have different match types and in this model we are
using only ones that are common across all vendors participating in
this draft. As in this example, the base ACL model can be extended
with company proprietary extensions, described in the next section.

```
module: example-ext-route-filter
  augment /ietf-acl:access-lists/ietf-acl:acl/ietf-acl:aces/ietf-acl:ac
e/ietf-acl:matches:
    +--rw (route-prefix)?
       +--:(range)
          +--rw (ipv4-range)?
          |  +--:(v4-lower-bound)
          |  |  +--rw v4-lower-bound?   inet:ipv4-prefix
          |  +--:(v4-upper-bound)
          |     +--rw v4-upper-bound?   inet:ipv4-prefix
          +--rw (ipv6-range)?
             +--:(v6-lower-bound)
             |  +--rw v6-lower-bound?   inet:ipv6-prefix
             +--:(v6-upper-bound)
                +--rw v6-upper-bound?   inet:ipv6-prefix

  file "example-ext-route-filter@2017-10-03.yang"
  module example-ext-route-filter {
    namespace "urn:ietf:params:xml:ns:yang:example-ext-route-filter";
```

```
prefix example-ext-route-filter;

import ietf-inet-types {
  prefix "inet";
}
import ietf-access-control-list {
  prefix "ietf-acl";
}

organization
  "Route model group.";

contact
  "abc@abc.com";

description "
  This module describes route filter as a collection of
  match prefixes. When specifying a match prefix, you
  can specify an exact match with a particular route or
  a less precise match. You can configure either a
  common action that applies to the entire list or an
  action associated with each prefix.
  ";
revision 2017-10-03 {
  description
    "Creating Route-Filter extension model based on
    ietf-access-control-list model";
  reference "Example route filter";
}

augment "/ietf-acl:access-lists/ietf-acl:acl/" +
        "ietf-acl:aces/ietf-acl:ace/ietf-acl:matches" {
  description "
    This module augments the matches container in the ietf-acl
    module with route filter specific actions";

  choice route-prefix{
    description "Define route filter match criteria";
    case range {
      description
        "Route falls between the lower prefix/prefix-length
         and the upperprefix/prefix-length.";
      choice ipv4-range {
        description "Defines the IPv4 prefix range";
        leaf v4-lower-bound {
          type inet:ipv4-prefix;
          description
            "Defines the lower IPv4 prefix/prefix length";
```

```
            }
            leaf v4-upper-bound {
              type inet:ipv4-prefix;
              description
                "Defines the upper IPv4 prefix/prefix length";
            }
          }
          choice ipv6-range {
            description "Defines the IPv6 prefix/prefix range";
            leaf v6-lower-bound {
              type inet:ipv6-prefix;
              description
                "Defines the lower IPv6 prefix/prefix length";
            }
            leaf v6-upper-bound {
              type inet:ipv6-prefix;
              description
                "Defines the upper IPv6 prefix/prefix length";
            }
          }
        }
      }
    }
  }
```

A.2. A company proprietary module example

   Module "example-newco-acl" is an example of company proprietary model
   that augments "ietf-acl" module. It shows how to use 'augment' with
   an XPath expression to add additional match criteria, action
   criteria, and default actions when no ACE matches found. All these
   are company proprietary extensions or system feature extensions.
   "example-newco-acl" is just an example and it is expected from
   vendors to create their own proprietary models.

   The following figure is the tree structure of example-newco-acl. In
   this example, /ietf-acl:access-lists/ietf-acl:acl/ietf-acl:aces/ietf-
   acl:ace/ietf-acl:matches are augmented with two new choices,
   protocol-payload-choice and metadata. The protocol-payload-choice
   uses a grouping with an enumeration of all supported protocol values.
   Metadata matches apply to fields associated with the packet but not
   in the packet header such as overall packet length. In other
   example, /ietf-acl:access-lists/ietf-acl:acl/ietf-acl:aces/ietf-
   acl:ace/ietf-acl:actions are augmented with new choice of actions.

```
module: example-newco-acl
  augment /ietf-acl:access-lists/ietf-acl:acl/ietf-acl:aces/ietf-acl:ac
e/ietf-acl:matches:
    +--rw (protocol-payload-choice)?
    |  +--:(protocol-payload)
    |     +--rw protocol-payload* [value-keyword]
    |        +--rw value-keyword    enumeration
    +--rw (metadata)?
       +--:(packet-length)
          +--rw packet-length?      uint16
  augment /ietf-acl:access-lists/ietf-acl:acl/ietf-acl:aces/ietf-acl:ac
e/ietf-acl:actions:
    +--rw (action)?
       +--:(count)
       |  +--rw count?                   string
       +--:(policer)
       |  +--rw policer?                 string
       +--:(hiearchical-policer)
          +--rw hierarchitacl-policer?   string
  augment /ietf-acl:access-lists/ietf-acl:acl:
    +--rw default-actions
       +--rw deny?   empty

  module example-newco-acl {

    yang-version 1.1;

    namespace "urn:newco:params:xml:ns:yang:example-newco-acl";

    prefix example-newco-acl;

    import ietf-access-control-list {
      prefix "ietf-acl";
    }

    organization
      "Newco model group.";

    contact
      "abc@newco.com";
    description
      "This YANG module augments IETF ACL Yang.";

    revision 2017-10-03 {
      description
        "Creating NewCo proprietary extensions to ietf-acl model";

      reference
```

```
        "RFC XXXX: Network Access Control List (ACL)
         YANG Data  Model";
    }

    augment "/ietf-acl:access-lists/ietf-acl:acl/" +
            "ietf-acl:aces/ietf-acl:ace/" +
            "ietf-acl:matches" {
      description "Newco proprietary simple filter matches";
      choice protocol-payload-choice {
        description "Newo proprietary payload match condition";
        list protocol-payload {
          key value-keyword;
          ordered-by user;
          description "Match protocol payload";
          uses match-simple-payload-protocol-value;
        }
      }

      choice metadata {
        description "Newco proprietary interface match condition";
        leaf packet-length {
          type uint16;
          description "Match on packet length";
        }
      }
    }

    augment "/ietf-acl:access-lists/ietf-acl:acl/" +
            "ietf-acl:aces/ietf-acl:ace/" +
            "ietf-acl:actions" {
      description "Newco proprietary simple filter actions";
      choice action {
        description "";
        case count {
          description "Count the packet in the named counter";
          leaf count {
            type string;
            description "";
          }
        }
        case policer {
          description "Name of policer to use to rate-limit traffic";
          leaf policer {
            type string;
            description "";
          }
        }
        case hiearchical-policer {
```

```
          description "Name of hierarchical policer to use to
                       rate-limit traffic";
          leaf hierarchitacl-policer {
            type string;
            description "";
          }
        }
      }
    }

    augment "/ietf-acl:access-lists/ietf-acl:acl" {
      description "Newco proprietary default action";
      container default-actions {
        description
          "Actions that occur if no access-list entry is matched.";
        leaf deny {
          type empty;
          description "";
        }
      }
    }

    grouping match-simple-payload-protocol-value {
      description "Newco proprietary payload";
      leaf value-keyword {
        type enumeration {
          enum icmp {
            description "Internet Control Message Protocol";
          }
          enum icmp6 {
            description "Internet Control Message Protocol Version 6";
          }
          enum range {
            description "Range of values";
          }
        }
        description "(null)";
      }
    }
  }
```

Draft authors expect that different vendors will provide their own
yang models as in the example above, which is the augmentation of the
base model

A.3. Linux nftables

   As Linux platform is becoming more popular as networking platform,
   the Linux data model is changing. Previously ACLs in Linux were
   highly protocol specific and different utilities were used (iptables,
   ip6tables, arptables, ebtables), so each one had separate data model.
   Recently, this has changed and a single utility, nftables, has been
   developed. With a single application, it has a single data model for
   filewall filters and it follows very similarly to the ietf-access-
   control list module proposed in this draft. The nftables support
   input and output ACEs and each ACE can be defined with match and
   action.

   The example in Section 4.3 can be configured using nftable tool as
   below.

```
      nft add table ip filter
      nft add chain filter input
      nft add rule ip filter input ip protocol tcp ip saddr \
          10.10.10.1/24 drop
```

   The configuration entries added in nftable would be.

```
      table ip filter {
        chain input {
          ip protocol tcp ip saddr 10.10.10.1/24 drop
        }
      }
```

   We can see that there are many similarities between Linux nftables
   and IETF ACL YANG data models and its extension models. It should be
   fairly easy to do translation between ACL YANG model described in
   this draft and Linux nftables.

A.4. Ethertypes

   The ACL module is dependent on the definition of ethertypes. IEEE
   owns the allocation of those ethertypes. This model is being
   included here to enable definition of those types till such time that
   IEEE takes up the task of publication of the model that defines those
   ethertypes. At that time, this model can be deprecated.

   <CODE BEGINS> file "ietf-ethertypes@2017-10-03.yang"

```
   module ietf-ethertypes {
     namespace "urn:ietf:params:xml:ns:yang:ietf-ethertypes";
     prefix ie;
```

```
      organization
        "IETF NETMOD (NETCONF Data Modeling Language)";

      contact
        "WG Web:    <http://tools.ietf.org/wg/netmod/>
         WG List:  <mailto:netmod@ietf.org>

         Editor:    Mahesh Jethanandani
                    <mjethanandani@gmail.com>";

      description
        "This module contains the common definitions for the
         Ethertype used by different modules. It is a
         placeholder module, till such time that IEEE
         starts a project to define these Ethertypes
         and publishes a standard.

         At that time this module can be deprecated.";

      revision 2017-10-03 {
        description
          "Initial revision.";
        reference
          "RFC XXXX: IETF Ethertype YANG Data Module.";
      }

      typedef ethertype {
        type union {
          type uint16;
          type enumeration {
            enum ipv4 {
              value 2048;
              description
                "Internet Protocol version 4 (IPv4) with a
                 hex value of 0x0800.";
              reference
                "RFC 791, Internet Protocol.";
            }
            enum arp {
              value 2054;
              description
                "Address Resolution Protocol (ARP) with a
                 hex value of 0x0806.";
              reference
                "RFC 826 An Ethernet Address Resolution Protocol.";
            }
            enum wlan {
              value 2114;
```

```
              description
                "Wake-on-LAN. Hex value of 0x0842.";
            }
            enum trill {
              value 8947;
              description
                "Transparent Interconnection of Lots of Links.
                 Hex value of 0x22F3.";
              reference
                "RFC 6325 Routing Bridges (RBridges): Base Protocol
                 Specification.";
            }
            enum srp {
              value 8938;
              description
                "Stream Reservation Protocol. Hex value of
                 0x22EA.";
              reference
                "IEEE 801.1Q-2011.";
            }
            enum decnet {
              value 24579;
              description
                "DECnet Phase IV. Hex value of 0x6003.";
            }
            enum rarp {
              value 32821;
              description
                "Reverse Address Resolution Protocol.
                 Hex value 0x8035.";
              reference
                    "RFC 903. A Reverse Address Resolution Protocol.";
            }
            enum appletalk {
              value 32923;
              description
                "Appletalk (Ethertalk). Hex value 0x809B.";
            }
            enum aarp {
              value 33011;
              description
                "Appletalk Address Resolution Protocol. Hex value
                 of 0x80F3.";
            }
            enum vlan {
              value 33024;
              description
                "VLAN-tagged frame (802.1Q) and Shortest Path
```

```
                 Bridging IEEE 802.1aq with NNI compatibility.
                 Hex value 0x8100.";
               reference
                    "802.1Q.";

             }
             enum ipx {
               value 33079;
               description
                 "Internetwork Packet Exchange (IPX). Hex value
                  of 0x8137.";
             }
             enum qnx {
               value 33284;
               description
                 "QNX Qnet. Hex value of 0x8204.";
             }
             enum ipv6 {
               value 34525;
               description
                 "Internet Protocol Version 6 (IPv6). Hex value
                  of 0x86DD.";
               reference
                 "RFC 8200, 8201.";
             }
             enum efc {
               value 34824;
               description
                 "Ethernet flow control using pause frames.
                  Hex value of 0x8808";
               reference
                 "IEEE Std. 802.1Qbb.";
             }
             enum esp {
               value 34825;
               description
                 "Ethernet Slow Protocol. Hex value of 0x8809.";
               reference
                 "IEEE Std. 802.3-2015";
             }
             enum cobranet {
               value 34841;
               description
                 "CobraNet. Hex value of 0x";
             }
             enum mpls-unicast {
               value 34887;
               description
```

```
            "MultiProtocol Label Switch (MPLS) unicast traffic.
             Hex value of 0x8847.";
          reference
            "RFC 3031.";
        }
        enum mpls-multicast {
          value 34888;
          description
            "MultiProtocol Label Switch (MPLS) multicast traffic.
             Hex value of 0x8848.";
          reference
            "RFC 3031.";
        }
        enum pppoe-discovery {
          value 34915;
          description
            "Point-to-Point Protocol over Ethernet. Used during
             the discovery process. Hex value of 0x8863.";
          reference
            "RFC 2516.";
        }
        enum pppoe-session {
          value 34916;
          description
            "Point-to-Point Protocol over Ethernet. Used during
             session stage. Hex value of 0x8864.";
          reference
            "RFC 2516.";
        }
        enum intel-ans {
          value 34925;
          description
            "Intel Advanced Networking Services. Hex value of
             0x886D.";
        }
        enum jumbo-frames {
          value 34928;
          description
            "Jumbo frames or Ethernet frames with more than
             1500 bytes of payload, upto 9000 bytes.";
        }
        enum homeplug {
          value 34939;
          description
            "Family name for the various power line
             communications. Hex value of 0x887B.";
        }
        enum eap {
```

```
            value 34958;
            description
              "Ethernet Access Protocol (EAP) over LAN. Hex value
               of 0x888E.";
            reference
              "IEEE 802.1X";
          }
          enum profinet {
            value 34962;
            description
              "PROcess FIeld Net (PROFINET). Hex value of 0x8892.";
          }
          enum hyperscsi {
            value 34970;
            description
              "SCSI over Ethernet. Hex value of 0x889A";
          }
          enum aoe {
            value 34978;
            description
              "Advanced Technology Advancement (ATA) over Ethernet.
               Hex value of 0x88A2.";
          }
          enum ethercat {
            value 34980;
            description
              "Ethernet for Control Automation Technology (EtherCAT).
               Hex value of 0x88A4.";
          }
          enum provider-bridging {
            value 34984;
            description
              "Provider Bridging (802.1ad) and Shortest Path Bridging
               (801.1aq). Hex value of 0x88A8.";
            reference
              "IEEE 802.1ad, IEEE 802.1aq).";
          }
          enum ethernet-powerlink {
            value 34987;
            description
              "Ethernet Powerlink. Hex value of 0x88AB.";
          }
          enum goose {
            value 35000;
            description
              "Generic Object Oriented Substation Event (GOOSE).
               Hex value of 0x88B8.";
            reference
```

```
            "IEC/ISO 8802-2 and 8802-3.";
        }
        enum gse {
          value 35001;
          description
            "Generic Substation Events. Hex value of 88B9.";
          reference
            "IEC 61850.";
        }
        enum sv {
          value 35002;
          description
            "Sampled Value Transmission. Hex value of 0x88BA.";
          reference
            "IEC 61850.";
        }
        enum lldp {
          value 35020;
          description
            "Link Layer Discovery Protocol (LLDP). Hex value of
             0x88CC.";
          reference
            "IEEE 802.1AB.";
        }
        enum sercos {
          value 35021;
          description
            "Sercos Interface. Hex value of 0x88CD.";
        }
        enum wsmp {
          value 35036;
          description
            "WAVE Short Message Protocl (WSMP). Hex value of
             0x88DC.";
        }
        enum homeplug-av-mme {
          value 35041;
          description
            "HomePlug AV MME. Hex value of 88E1.";
        }
        enum mrp {
          value 35043;
          description
            "Media Redundancy Protocol (MRP). Hex value of
             0x88E3.";
          reference
            "IEC62439-2.";
        }
```

```
            enum macsec {
              value 35045;
              description
                "MAC Security. Hex value of 0x88E5.";
              reference
                "IEEE 802.1AE.";
            }
            enum pbb {
              value 35047;
              description
                "Provider Backbone Bridges (PBB). Hex value of
                 0x88E7.";
              reference
                "IEEE 802.1ah.";
            }
            enum cfm {
              value 35074;
              description
                "Connectivity Fault Management (CFM). Hex value of
                 0x8902.";
              reference
                "IEEE 802.1ag.";
            }
            enum fcoe {
              value 35078;
              description
                "Fiber Channel over Ethernet (FCoE). Hex value of
                 0x8906.";
              reference
                "T11 FC-BB-5.";
            }
            enum fcoe-ip {
              value 35092;
              description
                "FCoE Initialization Protocol. Hex value of 0x8914.";
            }
            enum roce {
              value 35093;
              description
                "RDMA over Converged Ethernet (RoCE). Hex value of
                 0x8915.";
            }
            enum tte {
              value 35101;
              description
                "TTEthernet Protocol Control Frame (TTE). Hex value
                 of 0x891D.";
              reference
```

```
              "SAE AS6802.";
          }
          enum hsr {
            value 35119;
            description
              "High-availability Seamless Redundancy (HSR). Hex
               value of 0x892F.";
            reference
              "IEC 62439-3:2016.";
          }
          enum ctp {
            value 36864;
            description
              "Ethernet Configuration Test Protocol (CTP). Hex
               value of 0x9000.";
          }
          enum vlan-double-tagged {
            value 37120;
            description
              "VLAN-tagged frame with double tagging. Hex value
               of 0x9100.";
          }
        }
      }
      description
        "The uint16 type placeholder type is defined to enable
         users to manage their own ethertypes not
         covered by the module. Otherwise the module contains
         enum definitions for the more commonly used ethertypes.";
    }
  }

  <CODE ENDS>
```

Authors' Addresses

Mahesh Jethanandani
Cisco Systems, Inc

Email: mjethanandani@gmail.com

Lisa Huang
General Electric

Email: lyihuang16@gmail.com

   Sonal Agarwal
   Cisco Systems, Inc.

   Email: agarwaso@cisco.com

   Dana Blair
   Cisco Systems, INc

   Email: dblair@cisco.com