Stream: Internet Engineering Task Force (IETF)
RFC: 9899
Category: Standards Track
Published: December 2025
ISSN: 2070-1721
Authors: O. Gonzalez de Dios, *Telefonica*; S. Barguil, *Nokia*; M. Boucadair, *Orange*; Q. Wu, *Huawei*

RFC 9899
Extensions to the YANG Data Model for Access Control Lists (ACLs)

Abstract

RFC 8519 defines a YANG data model for Access Control Lists (ACLs). This document specifies a set of extensions that fix many of the limitations of the ACL model as initially defined in RFC 8519. Specifically, it introduces augmentations to the ACL base model to enhance its functionality and applicability.

This document also creates initial versions of IANA-maintained modules for ICMP types and IPv6 extension headers.

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/rfc9899.

Copyright Notice

Table of Contents

1. Introduction 3
2. Terminology 4
3. Overall Structure of the Enhanced ACL Module 5
   3.1. Tree Structure 5
   3.2. Defined Sets 9
   3.3. IPv6 Extension Headers 10
   3.4. TCP Flags Handling 10
   3.5. Fragments Handling 11
   3.6. Payload-Based Filtering 11
   3.7. Match on MPLS Headers 11
   3.8. VLAN Filtering 12
   3.9. Instance Service Identifier (I-SID) Filtering 12
   3.10. Additional Actions 12
4. Enhanced ACL YANG Module 12
5. Security Considerations 34
6. IANA Considerations 35
   6.1. URI Registrations 35
   6.2. YANG Module Name Registrations 36
   6.3. Considerations for IANA-Maintained Modules 36
      6.3.1. ICMPv4 Types IANA Module 36
      6.3.2. ICMPv6 Types IANA Module 37
      6.3.3. IPv6 Extension Header Types IANA Module 38
7. References 39
   7.1. Normative References 39
   7.2. Informative References 40

Appendix A. Problem Statement and Gap Analysis 42
A.1. Suboptimal Configuration: Lack of Support for Lists of Prefixes 42
A.2. Manageability: Impossibility of Using Aliases or Defined Sets 44
A.3. Bind ACLs to Devices, Not Only Interfaces 45
A.4. Partial or Lack of IPv4/IPv6 Fragment Handling 45
A.5. Suboptimal TCP Flags Handling 45
A.6. Rate-Limit Action 45
A.7. Payload-Based Filtering 45
A.8. Reuse the Content of ACLs Across Several Devices 46
A.9. Match MPLS Headers 46
Appendix B. Examples 46
B.1. TCP Flags Handling 46
B.2. Fragments Handling 47
B.3. Pattern-Based Filtering 50
B.4. VLAN Filtering 50
B.5. I-SID Filtering 51
B.6. Rate-Limit 52
Acknowledgments 53
Authors' Addresses 54

1. Introduction

[RFC8519] defines Access Control Lists (ACLs) as a user-ordered set of filtering rules. The model targets the configuration of the filtering behavior of a device. However, the model structure, as defined in [RFC8519], suffers from a set of limitations. This document identifies these limitations and specifies an enhanced ACL structure, introducing augmentations to the ACL base model (Section 4). The motivation of such an enhanced ACL structure is discussed in detail in Appendix A.

When managing ACLs, it is common for network operators to group match elements in predefined sets. The consolidation into group matches allows for reducing the number of rules, especially in large-scale networks. For example, if finding a match against 100 IP addresses (or

prefixes) is needed, a single rule will suffice rather than creating individual Access Control Entries (ACEs) for each IP address (or prefix). In doing so, implementations would optimize the performance of matching lists versus multiple rules matching.

The enhanced ACL structure (see "ietf-acl-enh" in Section 4) is also meant to facilitate the management of network operators. Instead of entering the IP address or port number literals, using user-named lists decouples the creation of the rule from the management of the sets. Hence, it is possible to remove/add entries to the list without redefining the (parent) ACL rule.

In addition, the notion of ACL and defined sets is generalized so that it is not device specific as per [RFC8519]. ACLs and defined sets may be defined at the network/administrative domain level and associated to devices. This approach facilitates the reusability across multiple network elements. For example, managing the IP prefix sets from a network level makes it easier to maintain by the security groups.

Network operators maintain sets of IP prefixes that are related to each other, e.g., deny-lists or accept-lists that are associated with those provided by a VPN customer. These lists are maintained and manipulated by security expert teams of the network operators.

Note that ACLs are used locally in devices but are triggered by other tools such as DDoS mitigation [RFC9132] or BGP Flow Spec [RFC8955] [RFC8956]. Therefore, it is valuable from a network operation standpoint to support the means to easily map to the filtering rules conveyed in messages triggered by these tools.

The enhanced ACL module (Section 4) conforms to the Network Management Datastore Architecture (NMDA) defined in [RFC8342].

A set of examples to illustrate the use of the enhanced ACL module is provided in Appendix B.

This document also creates initial versions of IANA-maintained modules for ICMP types and IPv6 extension headers. The design of the modules adheres to the recommendations in Section 4.30.2 of [YANG-GUIDELINES]. The latest version of these IANA-maintained modules can be retrieved from the "YANG Parameters" registry group [IANA-YANG-PARAMETERS].

2. Terminology

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.

The terminology for describing YANG modules is defined in [RFC7950]. The meaning of the symbols in the tree diagrams is defined in [RFC8340].

In addition to the terms defined in [RFC8519], this document makes use of the following term:

Defined set:

Elements in a defined set typically share a logical purpose or function, such as IP addresses, IP prefixes, port numbers, or ICMP types.

3. Overall Structure of the Enhanced ACL Module

3.1. Tree Structure

Figure 1 shows the full tree of the enhanced ACL module (Section 4):

```
module: ietf-acl-enh

  augment /acl:acls:
    +--rw defined-sets
       +---u defined-sets
  augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches:
    +--rw (payload)?
    |  +--:(pattern)
    |     +--rw pattern {match-on-payload}?
    |        +---u payload-match
    +--rw (alias)?
    |  +--:(alias-name)
    |     +--rw alias-name*       alias-ref
    +--rw (mpls)?
       +--:(mpls-values)
          +--rw mpls-values {match-on-mpls}?
             +---u mpls-match-parameters-config
  augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches/acl:l2:
    +--rw vlan-filter {match-on-vlan-filter}?
    |  +--rw frame-type?         string
    |  +--rw (vlan-type)?
    |     +--:(range)
    |     |  +--rw lower-vlan    uint16
    |     |  +--rw upper-vlan    uint16
    |     +--:(operator)
    |        +--rw operator?     packet-fields:operator
    |        +--rw vlan*         uint16
    +--rw isid-filter {match-on-isid-filter}?
       +--rw (isid-type)?
          +--:(range)
          |  +--rw lower-isid    uint16
          |  +--rw upper-isid    uint16
          +--:(operator)
             +--rw operator?     packet-fields:operator
             +--rw isid*         uint16
  augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches/acl:l3
            /acl:ipv4/acl:ipv4:
    +--rw ipv4-fragment
    |  +---u fragment-fields
    +--rw source-ipv4-prefix-list?        ipv4-prefix-set-ref
    +--rw destination-ipv4-prefix-list?   ipv4-prefix-set-ref
    +--rw protocol-set?                   protocol-set-ref
  augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches/acl:l3
            /acl:ipv6/acl:ipv6:
    +--rw ipv6-fragment
    |  +---u fragment-fields
    +--rw source-ipv6-prefix-list?        ipv6-prefix-set-ref
    +--rw destination-ipv6-prefix-list?   ipv6-prefix-set-ref
    +--rw protocol-set?                   protocol-set-ref
    +--rw extension-header?
            iana-ipv6-ext-types:ipv6-extension-header-type
  augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches/acl:l4
            /acl:tcp/acl:tcp:
    +--rw flags-bitmask
    |  +---u tcp-flags
    +--rw source-tcp-port-set?        port-set-ref
    +--rw destination-tcp-port-set?   port-set-ref
```

```
augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches/acl:l4
          /acl:udp/acl:udp:
  +--rw source-udp-port-set?        port-set-ref
  +--rw destination-udp-port-set?   port-set-ref
augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches/acl:l4
          /acl:icmp/acl:icmp:
  +--rw icmpv4-set?   icmpv4-type-set-ref
  +--rw icmpv6-set?   icmpv6-type-set-ref
augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:actions:
  +---u acl-complementary-actions
  +--rw rate-limit?                  decimal64
```

Figure 1: Enhanced ACL Tree Structure

Figure 2 shows the reusable groupings that are defined in the enhanced ACL module:

```
  grouping tcp-flags:
    +--rw operator?                 operator
    +-- (mode)?
       +--:(explicit)
       |  +-- explicit-tcp-flag*   identityref
       +--:(builtin)
          +-- bitmask?             uint16
  grouping fragment-fields:
    +-- operator?   operator
    +-- type?       fragment-type
  grouping mpls-match-parameters-config:
    +-- traffic-class?       uint8
    +-- label-position?      identityref
    +-- upper-label-range?   rt-types:mpls-label
    +-- lower-label-range?   rt-types:mpls-label
    +-- label-block-name?    string
    +-- ttl-value?           uint8
  grouping payload-match:
    +-- offset?       identityref
    +-- length?   uint16
    +-- operator?     operator
    +-- pattern?       binary
  grouping alias:
    +-- vlan*         uint16
    +-- prefix*       inet:ip-prefix
    +-- port-range* [lower-port]
    |  +-- lower-port    inet:port-number
    |  +-- upper-port?   inet:port-number
    +-- protocol*     uint8
    +-- fqdn*         inet:domain-name
    +-- uri*          inet:uri
  grouping icmpv4-header-fields:
    +-- type?             iana-icmpv4-types:icmpv4-type
    +-- code?             uint8
    +-- rest-of-header?   binary
  grouping icmpv6-header-fields:
    +-- type?             iana-icmpv6-types:icmpv6-type
    +-- code?             uint8
    +-- rest-of-header?   binary
  grouping acl-complementary-actions:
    +-- log-action
    |  +-- log-type?   identityref
    |  +-- log-id?     string
    +-- counter-action
       +-- counter-type?   identityref
       +-- counter-name*   string
  grouping ipv4-prefix-sets:
    +-- prefix-set* [name]
       +-- name           string
       +-- description?   string
       +-- prefix*        inet:ipv4-prefix
  grouping ipv6-prefix-sets:
    +-- prefix-set* [name]
       +-- name           string
       +-- description?   string
       +-- prefix*        inet:ipv6-prefix
  grouping port-sets:
```

```
     +-- port-set* [name]
        +-- name    string
        +-- port* [id]
           +-- id                                string
           +-- (port)?
              +--:(port-range-or-operator)
                 +-- port-range-or-operator
                    +---u packet-fields:port-range-or-operator
   grouping protocol-sets:
     +-- protocol-set* [name]
        +-- name        string
        +-- protocol*   union
   grouping icmpv4-type-sets:
     +-- set* [name]
        +-- name           string
        +-- icmpv4-type* [type]
           +---u icmpv4-header-fields
   grouping icmpv6-type-sets:
     +-- set* [name]
        +-- name           string
        +-- icmpv6-type* [type]
           +---u icmpv6-header-fields
   grouping aliases:
     +-- alias* [name]
        +-- name     string
        +---u alias
   grouping defined-sets:
     +-- ipv4-prefix-sets
     |  +---u ipv4-prefix-sets
     +-- ipv6-prefix-sets
     |  +---u ipv6-prefix-sets
     +-- port-sets
     |  +---u port-sets
     +-- protocol-sets
     |  +---u protocol-sets
     +-- icmpv4-type-sets
     |  +---u icmpv4-type-sets
     +-- icmpv6-type-sets
     |  +---u icmpv6-type-sets
     +-- aliases
        +---u aliases
```

Figure 2: Enhanced ACL Groupings

3.2. Defined Sets

The augmented ACL structure includes several containers to manage reusable sets of elements that can be matched in an ACL entry. Each set is uniquely identified by a name and can be called from the relevant entry. The following sets (seen in Figure 1) are defined:

IPv4 prefix sets: An IPv4 prefix set contains a list of IPv4 prefixes. A match will be considered if the IP address (source or destination, depending on the ACL entry) is contained in any of the prefixes in the set.

IPv6 prefix sets: An IPv6 prefix contains a list of IPv6 prefixes. A match will be considered if the IP address (source or destination, depending on the ACL entry) is contained in any of the prefixes in the set.

Port sets: A port set contains a list of port numbers to be used in transport protocol entries (e.g., TCP and UDP).

A port number can be a port range or a single port number along with an operator (equal to, greater than or equal to, etc.).

Protocol sets: A protocol set contains a list of protocol values. A protocol can be identified by either a number (e.g., 17) or a name (e.g., UDP).

ICMP sets: An ICMP set contains a list of ICMPv4 [RFC0792] or ICMPv6 [RFC4443] types, and each type is identified by a type value and is optionally identified by the code and the rest of the header.

IANA-maintained modules for ICMP types are created in this document.

Aliases: An alias is defined by a combination of various parameters (e.g., IP prefix, protocol, port number, or VLAN [IEEE802.1Qcp]). When only sets of one parameter (e.g., protocol) are handled, then the relevant parameter sets should be used (e.g., protocol set) rather than an alias.

For example, an alias can be defined to apply ACL policies bound to a set of HTTPS servers. Such an alias will typically include these HTTPS server addresses (e.g., "prefix": ["2001:db8:6401::1/128","2001:db8:6401::2/128"]) and the TCP port number 443 (i.e., "protocol": [6] and "lower-port": 443).

Sets of aliases can be defined and referred to in ACL match criteria.

Payload-based filtering: A network traffic filtering technique that examines the data payload of packets, beyond just the header information, to identify, allow, or block traffic based on specific content or patterns within the payload. An offset type (e.g., Layer 2 or Layer 3) is used to indicate the position of the data in the packet to use for the match.

3.3. IPv6 Extension Headers

The enhanced ACL module can be used to manage ACLs that require matching against IPv6 extension headers [RFC8200]. To that aim, a new IANA-maintained module for IPv6 extension header types, "iana-ipv6-ext-types", is created in this document.

3.4. TCP Flags Handling

The augmented ACL module includes a new container 'flags-bitmask' to better handle TCP flags (Section 3.1 of [RFC9293]). Assigned TCP flags are maintained in the "TCP Header Flags" registry under the "Transmission Control Protocol (TCP) Parameters" registry group [IANA-TCP-FLAGS].

Clients that support both 'flags-bitmask' and 'flags' [RFC8519] matching fields **MUST NOT** set these fields in the same request.

3.5. Fragments Handling

The augmented ACL module includes new leafs 'ipv4-fragment' and 'ipv6-fragment' to better handle fragments.

Clients that support both 'ipv4-fragment' and 'flags' [RFC8519] matching fields **MUST NOT** set these fields in the same request.

3.6. Payload-Based Filtering

Some transport protocols use existing protocols (e.g., TCP or UDP) as substrate. The match criteria for such protocols may rely upon the 'protocol' under 'l3', TCP/UDP match criteria, part of the TCP/UDP payload, or a combination thereof.

A new feature, called 'match-on-payload', is defined in the document. This can be used, for example, for QUIC [RFC9000] or for tunneling protocols. This feature requires configuring a data offset, a length, and a binary pattern to match data against using a specified operator. The data offset indicates the position to look at in a packet (e.g., it starts at the beginning of the IP header or transport header).

3.7. Match on MPLS Headers

The enhanced ACL module (Section 4) can be used to create rules to match against the MPLS fields of a packet. The MPLS header defined in [RFC3032] and [RFC5462] contains the following fields:

- Traffic Class: The 3-bit "Exp" field [RFC3032], which is renamed to "Traffic Class field" ("TC field") [RFC5462].
- Label Value: A 20-bit field that carries the actual value of the MPLS label.
- TTL: An 8-bit field used to encode the Time-to-Live (TTL) value.

The augmented ACL module can be used by an operator to configure ACLs that match based upon the following data nodes:

- 'traffic-class'
- 'label-position' (e.g., top or bottom)
- 'upper-label-range'
- 'lower-label-range'
- 'label-block-name'
- 'ttl-value'

3.8. VLAN Filtering

Being able to filter all packets that are bridged within a VLAN or that are routed into or out of a bridge domain is part of the VPN control requirements for Ethernet VPN (EVPN) [RFC7209].

All packets that are bridged within a VLAN or that are routed into or out of a VLAN can be captured, forwarded, translated, or discarded based on the network policy.

3.9. Instance Service Identifier (I-SID) Filtering

Provider Backbone Bridging (PBB) was originally defined as a Virtual Bridged Local Area Networks standard [IEEE-802-1ah]. However, instead of multiplexing VLANs, PBB duplicates the Media Access Control (MAC) layer of the customer frame and separates it from the provider domain, by encapsulating it in a 24-bit Instance Service Identifier (I-SID). This provides more transparency between the customer network and the provider network.

The I-component forms the customer- or access-facing interface or routing instance. The I-component is responsible for mapping customer Ethernet traffic to the appropriate I-SID. It is mandatory to configure the default service identifier in the network.

Being able to filter by I-component service identifier is a feature of the EVPN-PBB configuration.

3.10. Additional Actions

In order to support rate-limiting (see Appendix A.6), a new action called 'rate-limit' is defined in this document.

Also, the "ietf-acl-enh" module supports new actions to complement existing ones: log ('log-action') and write a counter ('counter-action'). The version of the module defined in this document supports only local actions.

4. Enhanced ACL YANG Module

This YANG module imports types from [RFC6991], [RFC8519], and [RFC8294].

```
<CODE BEGINS> file "ietf-acl-enh@2025-11-07.yang"

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

  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-access-control-list {
    prefix acl;
    reference
      "RFC 8519: YANG Data Model for Network Access
                 Control Lists (ACLs), Section 4.1";
  }
  import ietf-packet-fields {
    prefix packet-fields;
    reference
      "RFC 8519: YANG Data Model for Network Access
                 Control Lists (ACLs), Section 4.2";
  }
  import ietf-routing-types {
    prefix rt-types;
    reference
      "RFC 8294: Common YANG Data Types for the Routing Area";
  }
  import iana-icmpv4-types {
    prefix iana-icmpv4-types;
    reference
      "RFC 9899: Extensions to the YANG Data Model for Access
                 Control Lists (ACLs)";
  }
  import iana-icmpv6-types {
    prefix iana-icmpv6-types;
    reference
      "RFC 9899: Extensions to the YANG Data Model for Access
                 Control Lists (ACLs)";
  }
  import iana-ipv6-ext-types {
    prefix iana-ipv6-ext-types;
    reference
      "RFC 9899: Extensions to the YANG Data Model for Access
                 Control Lists (ACLs)";
  }

  organization
    "IETF NETMOD Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/netmod/>
     WG List:  <mailto:netmod@ietf.org>

     Author:   Mohamed Boucadair
               <mailto:mohamed.boucadair@orange.com>
     Author:   Samier Barguil
               <mailto:samier.barguil_giraldo@nokia.com>
     Author:   Oscar Gonzalez de Dios
               <mailto:oscar.gonzalezdedios@telefonica.com>";
  description

    "This module contains YANG definitions for enhanced ACLs.

     Copyright (c) 2025 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 9899; see the
      RFC itself for full legal notices.";

  revision 2025-11-07 {
    description
      "Initial revision.";
    reference
      "RFC 9899: Extensions to the YANG Data Model for Access
                 Control Lists (ACLs)";
  }

  feature match-on-payload {
    description
      "Match based on a pattern is supported.";
  }

  feature match-on-vlan-filter {
    description
      "Match based on a VLAN range is supported.";
  }

  feature match-on-isid-filter {
    description
      "Match based on an I-SID range is supported.";
  }

  feature match-on-alias {
    description
      "Match based on aliases.";
  }

  feature match-on-mpls {
    description
      "Match based on MPLS headers.";
  }

  identity offset-type {
    description
      "Base identity for payload offset type.";
  }

  identity layer2 {
    base offset-type;
    description
      "The offset starts at the beginning of the Data Link Layer
       header.";
  }

  identity layer3 {
    base offset-type;
    description
```

```
      "The offset starts at the beginning of the IP header.";
  }

  identity layer4 {
    base offset-type;
    description
      "The offset starts right after the IP header (including
       any options or headers pertaining to that IP layer, e.g.,
       IPv6 Extension Headers and the Authentication Header (AH)).

       This can be typically the beginning of transport header
       (e.g., UDP, TCP, the Stream Control Transmission Protocol
       (SCTP), and the Datagram Congestion Control Protocol (DCCP))
       or any encapsulation scheme over IP such as IP-in-IP.";
  }

  identity payload {
    base offset-type;
    description
      "The offset starts right after the end of the transport
       header.  For example, this represents the beginning of the
       TCP data right after any TCP options or the beginning of
       the UDP payload right after the UDP header.

       This type may be used for matches against any data in
       the transport payload and/or any surplus area (if any,
       such as in UDP).";
  }

  identity tcp-flag {
    description
      "Base identity for the TCP Flags.";
    reference
      "RFC 9293: Transmission Control Protocol (TCP), Section 3.1";
  }

  identity ack {
    base tcp-flag;
    description
      "Acknowledgment TCP flag bit.";
    reference
      "RFC 9293: Transmission Control Protocol (TCP), Section 3.1";
  }

  identity syn {
    base tcp-flag;
    description
      "Synchronize sequence numbers.";
    reference
      "RFC 9293: Transmission Control Protocol (TCP), Section 3.1";
  }

  identity fin {
    base tcp-flag;
    description
      "No more data from the sender.";
    reference
      "RFC 9293: Transmission Control Protocol (TCP), Section 3.1";
```

```
  }

  identity urg {
    base tcp-flag;
    description
      "Urgent pointer TCP flag bit.";
    reference
      "RFC 9293: Transmission Control Protocol (TCP), Section 3.1";
  }

  identity psh {
    base tcp-flag;
    description
      "The Push function flag is similar to the URG flag and tells
       the receiver to process these packets as they are received
       instead of buffering them.";
    reference
      "RFC 9293: Transmission Control Protocol (TCP), Section 3.1";
  }

  identity rst {
    base tcp-flag;
    description
      "Reset TCP flag bit.";
    reference
      "RFC 9293: Transmission Control Protocol (TCP), Section 3.1";
  }

  identity ece {
    base tcp-flag;
    description
      "ECN-Echo TCP flag bit.";
    reference
      "RFC 9293: Transmission Control Protocol (TCP), Section 3.1";
  }

  identity cwr {
    base tcp-flag;
    description
      "Congestion Window Reduced flag bit.";
    reference
      "RFC 9293: Transmission Control Protocol (TCP), Section 3.1";
  }

  identity ae {
    base tcp-flag;
    description
      "Accurate Explicit Congestion Notification (ECN).

       Previously used as Nonce Sum (NS), which is now
       historic.";
  }

  identity mpls-acl-type {
    base acl:acl-base;
    description
      "An ACL that matches on fields from the MPLS header.";
  }
```

```
identity label-position {
  description
    "Base identity for deriving MPLS label position.";
}

identity top {
  base label-position;
  description
    "Top of the label stack.";
}

identity bottom {
  base label-position;
  description
    "Bottom of the label stack.";
}

identity log-types {
  description
    "Base identity for deriving the log actions.";
}

identity local-log {
  base log-types;
  description
    "A local log is used to record the ACL results.";
}

identity counter-type {
  description
    "Base identity for deriving the counter actions.";
}

identity counter-name {
  base counter-type;
  description
    "Identity for counter name to be updated based on
      the ACL match actions.";
}

typedef operator {
  type bits {
    bit not {
      position 0;
      description
        "If set, the logical negation of operation.";
    }
    bit match {
      position 1;
      description
        "Match bit.  This is a bitwise match operation defined as
         '(data & value) == value'.";
    }
    bit any {
      position 2;
      description
        "Any bit.  This is a match on any of the bits in bitmask.
```

```
          It evaluates to 'true' if any of the bits in the
          value mask are set in the data, i.e.,
          '(data & value) != 0'.";
      }
    }
    description
      "Specifies how to apply the defined bitmask.  The
       'any' and 'match' bits must not be set simultaneously.";
  }

  typedef fragment-type {
    type bits {
      bit df {
        position 0;
        description
          "Don't fragment bit for IPv4.  Must be set to 0 when it
           appears in an IPv6 filter.";
      }
      bit isf {
        position 1;
        description
          "Is a fragment.";
      }
      bit ff {
        position 2;
        description
          "First fragment.";
      }
      bit lf {
        position 3;
        description
          "Last fragment.";
      }
    }
    description
      "Different fragment types to match against.";
  }

  typedef ipv4-prefix-set-ref {
    type leafref {
      path "/acl:acls/acl-enh:defined-sets/acl-enh:ipv4-prefix-sets"
         + "/acl-enh:prefix-set/acl-enh:name";
    }
    description
      "Defines a reference to an IPv4 prefix set.";
  }

  typedef ipv6-prefix-set-ref {
    type leafref {
      path "/acl:acls/acl-enh:defined-sets/acl-enh:ipv6-prefix-sets"
         + "/acl-enh:prefix-set/acl-enh:name";
    }
    description
      "Defines a reference to an IPv6 prefix set.";
  }

  typedef port-set-ref {
    type leafref {
```

```
      path "/acl:acls/acl-enh:defined-sets/acl-enh:port-sets"
         + "/acl-enh:port-set/acl-enh:name";
    }
    description
      "Defines a reference to a port set.";
  }

  typedef protocol-set-ref {
    type leafref {
      path "/acl:acls/acl-enh:defined-sets/acl-enh:protocol-sets"
         + "/acl-enh:protocol-set/acl-enh:name";
    }
    description
      "Defines a reference to a protocol set.";
  }

  typedef icmpv4-type-set-ref {
    type leafref {
      path "/acl:acls/acl-enh:defined-sets/acl-enh:icmpv4-type-sets"
         + "/acl-enh:set/acl-enh:name";
    }
    description
      "Defines a reference to an ICMPv4 type set.";
  }

  typedef icmpv6-type-set-ref {
    type leafref {
      path "/acl:acls/acl-enh:defined-sets/acl-enh:icmpv6-type-sets"
         + "/acl-enh:set/acl-enh:name";
    }
    description
      "Defines a reference to an ICMPv6 type set.";
  }

  typedef alias-ref {
    type leafref {
      path "/acl:acls/acl-enh:defined-sets/acl-enh:aliases"
         + "/acl-enh:alias/acl-enh:name";
    }
    description
      "Defines a reference to an alias.";
  }

  grouping tcp-flags {
    description
      "Operations on TCP flags.";
    leaf operator {
      type operator;
      description
        "How to interpret the TCP flags.";
    }
    choice mode {
      description
        "Choice of how flags are indicated.";
      case explicit {
        leaf-list explicit-tcp-flag {
          type identityref {
            base acl-enh:tcp-flag;
```

```
          }
          description
            "An explicit list of the TCP flags that are to be
             matched.";
        }
      }
      case builtin {
        leaf bitmask {
          type uint16;
          description
            "The bitmask matches the last 4 bits of byte 13
             and byte 14 of the TCP header.
             For clarity, the 4 bits of byte 12
             corresponding to the TCP data offset field are not
             included in any matching.  Assigned TCP flags
             and their position are maintained in the IANA
             'Transmission Control Protocol (TCP) Parameters'
             registry group.";
          reference
            "RFC 9293: Transmission Control Protocol (TCP),
                       Section 3.1
             IANA: Transmission Control Protocol (TCP) Parameters,
             <https://www.iana.org/assignments/tcp-parameters>";
        }
      }
    }
  }

  grouping fragment-fields {
    description
      "Operations on fragment types.";
    leaf operator {
      type operator;
      default "match";
      description
        "How to interpret the fragment type.";
    }
    leaf type {
      type fragment-type;
      description
        "Specifies what fragment type to look for.";
    }
  }

  grouping mpls-match-parameters-config {
    description
      "Parameters for the configuration of MPLS match rules.";
    leaf traffic-class {
      type uint8 {
        range "0..7";
      }
      description
        "The value of the MPLS Traffic Class (TC) bits,
         formerly known as the EXP bits.";
    }
    leaf label-position {
      type identityref {
        base acl-enh:label-position;
```

```
      }
      description
        "Position of the label.";
    }
    leaf upper-label-range {
      type rt-types:mpls-label;
      description
        "Match MPLS label value on the MPLS header.
         The usage of this field indicates the upper
         range value in the top of the stack.  This
         label value does not include the encodings
         of TC and Time-to-Live (TTL).";
      reference
        "RFC 3032: MPLS Label Stack Encoding";
    }
    leaf lower-label-range {
      type rt-types:mpls-label;
      description
        "Match MPLS label value on the MPLS header.
         The usage of this field indicates the lower
         range value in the top of the stack.
         This label value does not include the
         encodings of TC and TTL.";
      reference
        "RFC 3032: MPLS Label Stack Encoding";
    }
    leaf label-block-name {
      type string;
      description
        "Reference to a label block predefined in the
         implementation.";
    }
    leaf ttl-value {
      type uint8;
      description
        "TTL MPLS packet value match.";
      reference
        "RFC 3032: MPLS Label Stack Encoding";
    }
  }

  grouping payload-match {
    description
      "Operations on payload match.";
    leaf offset {
      type identityref {
        base acl-enh:offset-type;
      }
      description
        "Indicates the payload offset.  This will indicate
         the position of the data in the packet to use for
         the match.";
    }
    leaf length {
      type uint16;
      units "bytes";
      description
        "Indicates the number of bytes to ignore, starting from
```

```
          the offset, to perform the pattern match.";
      }
      leaf operator {
        type operator;
        default "match";
        description
          "How to interpret the pattern match.";
      }
      leaf pattern {
        type binary;
        description
          "The binary pattern to match against starting.
           The match starts from the byte indicated by
           'offset' + length'.";
      }
    }

    grouping alias {
      description
        "Specifies an alias.";
      leaf-list vlan {
        type uint16;
        description
          "VLAN of the alias.";
        reference
          "IEEE Std 802.1Q: Bridges and Bridged Networks";
      }
      leaf-list prefix {
        type inet:ip-prefix;
        description
          "IPv4 or IPv6 prefix of the alias.";
      }
      list port-range {
        key "lower-port";
        description
          "Port range.  When only lower-port is
           present, it represents a single port number.";
        leaf lower-port {
          type inet:port-number;
          mandatory true;
          description
            "Lower port number of the port range.";
        }
        leaf upper-port {
          type inet:port-number;
          must '. >= ../lower-port' {
            error-message
              "The upper-port number must be greater than
               or equal to the lower-port number.";
          }
          description
            "Upper port number of the port range.";
        }
      }
      leaf-list protocol {
        type uint8;
        description
          "Identifies the target protocol number.
```

```
          For example, 6 for TCP or 17 for UDP.";
    }
    leaf-list fqdn {
      type inet:domain-name;
      description
        "Fully Qualified Domain Name (FQDN) identifying the target.";
    }
    leaf-list uri {
      type inet:uri;
      description
        "URI identifying the target.";
    }
  }

  grouping icmpv4-header-fields {
    description
      "Collection of ICMPv4 header fields that can be
       used to set up a match filter.";
    leaf type {
      type iana-icmpv4-types:icmpv4-type;
      description
        "Also known as control messages.";
      reference
        "RFC 792: Internet Control Message Protocol.";
    }
    leaf code {
      type uint8;
      description
        "ICMP subtype.";
      reference
        "RFC 792: Internet Control Message Protocol.";
    }
    leaf rest-of-header {
      type binary;
      description
        "Unbounded in length, the contents vary based on the
         ICMP type and code.";
      reference
        "RFC 792: Internet Control Message Protocol";
    }
  }

  grouping icmpv6-header-fields {
    description
      "Collection of ICMPv6 header fields that can be
       used to set up a match filter.";
    leaf type {
      type iana-icmpv6-types:icmpv6-type;
      description
        "Also known as control messages.";
      reference
        "RFC 4443: Internet Control Message Protocol (ICMPv6)
                   for Internet Protocol Version 6 (IPv6)
                   Specification.";
    }
    leaf code {
      type uint8;
      description
```

```
        "ICMP code.";
      reference
        "RFC 4443: Internet Control Message Protocol (ICMPv6)
                   for Internet Protocol Version 6 (IPv6)
                   Specification.";
    }
    leaf rest-of-header {
      type binary;
      description
        "Unbounded in length, the contents vary based on the
         ICMP type and code.  Also referred to as 'Message Body'
         in ICMPv6.";
      reference
        "RFC 4443: Internet Control Message Protocol (ICMPv6)
                   for Internet Protocol Version 6 (IPv6)
                   Specification.";
    }
  }

  grouping acl-complementary-actions {
    description
      "Collection of complementary ACL actions.";
    container log-action {
      description
        "Container for defining log actions.";
      leaf log-type {
        type identityref {
          base acl-enh:log-types;
        }
        description
          "The type of log action to be performed.";
      }
      leaf log-id {
        when "derived-from-or-self(../log-type, "
           + "'acl-enh:local-log')" {
          description
            "Name of the log file updated when type is 'local-log'.";
        }
        type string;
        description
          "The name of the counter action.";
      }
    }
    container counter-action {
      description
        "Container for defining counter actions.";
      leaf counter-type {
        type identityref {
          base acl-enh:counter-type;
        }
        description
          "The type of counter action to be performed.";
      }
      leaf-list counter-name {
        when "derived-from-or-self(../counter-type, "
           + "'acl-enh:counter-name')" {
          description
            "Name for the counter or variable to update when
```

```
            'counter-type' is 'counter-name'.";
        }
        type string;
        description
          "List of possible variables or counter names to
           update based on match criteria.";
      }
    }
  }

  grouping ipv4-prefix-sets {
    description
      "Data definitions for a list of IPv4 prefixes,
       which are matched as part of a policy.";
    list prefix-set {
      key "name";
      description
        "List of the defined prefix sets.";
      leaf name {
        type string;
        description
          "Name of the prefix set -- this is used as a label to
           reference the set in match conditions.";
      }
      leaf description {
        type string;
        description
          "Defined set description.";
      }
      leaf-list prefix {
        type inet:ipv4-prefix;
        description
          "List of IPv4 prefixes to be used in match
           conditions.";
      }
    }
  }

  grouping ipv6-prefix-sets {
    description
      "Data definitions for a list of IPv6 prefixes, which are
       matched as part of a policy.";
    list prefix-set {
      key "name";
      description
        "List of the defined prefix sets.";
      leaf name {
        type string;
        description
          "Name of the prefix set -- this is used as a label to
           reference the set in match conditions.";
      }
      leaf description {
        type string;
        description
          "A textual description of the prefix list.";
      }
      leaf-list prefix {
```

```
        type inet:ipv6-prefix;
        description
          "List of IPv6 prefixes to be used in match conditions.";
      }
    }
  }

  grouping port-sets {
    description
      "Data definitions for a list of ports, which can
       be matched in policies.";
    list port-set {
      key "name";
      description
        "List of port set definitions.";
      leaf name {
        type string;
        description
          "Name of the port set -- this is used as a label to
           reference the set in match conditions.";
      }
      list port {
        key "id";
        description
          "Port numbers along with the operator on which to
           match.";
        leaf id {
          type string;
          description
            "Identifier of the list of port numbers.";
        }
        choice port {
          description
            "Choice of specifying the port number or referring to a
             group of port numbers.";
          container port-range-or-operator {
            description
              "Indicates a set of ports.";
            uses packet-fields:port-range-or-operator;
          }
        }
      }
    }
  }

  grouping protocol-sets {
    description
      "Data definitions for a list of protocols, which can be
       matched in policies.";
    list protocol-set {
      key "name";
      description
        "List of protocol set definitions.";
      leaf name {
        type string;
        description
          "Name of the protocols set -- this is used as a
           label to reference the set in match conditions.";
```

```
      }
      leaf-list protocol {
        type union {
          type uint8;
          type string;
        }
        description
          "Value of the protocol set.";
      }
    }
  }

  grouping icmpv4-type-sets {
    description
      "Data definitions for a list of ICMPv4 types, which can be
       matched in policies.";
    list set {
      key "name";
      description
        "List of ICMPv4 type set definitions.";
      leaf name {
        type string;
        description
          "Name of the ICMPv4 type set -- this is used as a label
           to reference the set in match conditions.";
      }
      list icmpv4-type {
        key "type";
        description
          "Includes a list of ICMPv4 types.";
        uses icmpv4-header-fields;
      }
    }
  }

  grouping icmpv6-type-sets {
    description
      "Data definitions for a list of ICMPv6 types, which can be
       matched in policies.";
    list set {
      key "name";
      description
        "List of ICMP type set definitions.";
      leaf name {
        type string;
        description
          "Name of the ICMPv6 type set -- this is used as a label
           to reference the set in match conditions.";
      }
      list icmpv6-type {
        key "type";
        description
          "Includes a list of ICMPv6 types.";
        uses icmpv6-header-fields;
      }
    }
  }
```

```
  grouping aliases {
    description
      "Grouping for a set of aliases.";
    list alias {
      key "name";
      description
        "List of aliases.";
      leaf name {
        type string;
        description
          "The name of the alias.";
      }
      uses alias;
    }
  }

  grouping defined-sets {
    description
      "Predefined sets of attributes used in policy match
       statements.";
    container ipv4-prefix-sets {
      description
        "Data definitions for a list of IPv4 or IPv6
         prefixes, which are matched as part of a policy.";
      uses ipv4-prefix-sets;
    }
    container ipv6-prefix-sets {
      description
        "Data definitions for a list of IPv6 prefixes, which are
         matched as part of a policy.";
      uses ipv6-prefix-sets;
    }
    container port-sets {
      description
        "Data definitions for a list of ports, which can
         be matched in policies.";
      uses port-sets;
    }
    container protocol-sets {
      description
        "Data definitions for a list of protocols, which can be
         matched in policies.";
      uses protocol-sets;
    }
    container icmpv4-type-sets {
      description
        "Data definitions for a list of ICMPv4 types, which can be
         matched in policies.";
      uses icmpv4-type-sets;
    }
    container icmpv6-type-sets {
      description
        "Data definitions for a list of ICMPv6 types, which can be
         matched in policies.";
      uses icmpv6-type-sets;
    }
    container aliases {
      description
```

```
        "Top-level container for aliases.";
      uses aliases;
    }
  }

  augment "/acl:acls" {
    description
      "predefined sets.";
    container defined-sets {
      description
        "Predefined sets of attributes used in policy match
         statements.";
      uses defined-sets;
      nacm:default-deny-write;
    }
  }

  augment "/acl:acls/acl:acl/acl:aces/acl:ace"
        + "/acl:matches" {
    description
      "Adds a match type based on the payload.";
    choice payload {
      description
        "Matches based upon a prefix pattern.";
      container pattern {
        if-feature "match-on-payload";
        description
          "Indicates the rule to perform the payload-based match.";
        uses payload-match;
      }
    }
    choice alias {
      description
        "Matches based upon aliases.";
      leaf-list alias-name {
        type alias-ref;
        description
          "Indicates one or more aliases.";
      }
    }
    choice mpls {
      description
        "Matches against MPLS headers, for example, label
         values.";
      container mpls-values {
        if-feature "match-on-mpls";
        description
          "Provides the rule set that matches MPLS headers.";
        uses mpls-match-parameters-config;
      }
    }
  }

  augment "/acl:acls/acl:acl/acl:aces"
        + "/acl:ace/acl:matches/acl:l2" {
    description
      "Adds a match type based on MAC VLAN and I-SID filters.";
    container vlan-filter {
```

```
        if-feature "match-on-vlan-filter";
        description
          "Indicates how to handle MAC VLANs.";
        leaf frame-type {
          type string;
          description
            "Entering the frame type allows the
             filter to match a specific type of frame format.";
        }
        choice vlan-type {
          description
            "VLAN definition from range or operator.";
          case range {
            leaf lower-vlan {
              type uint16;
              must '. <= ../upper-vlan' {
                error-message
                  "The lower-vlan must be less than or equal to
                   the upper-vlan.";
              }
              mandatory true;
              description
                "Lower boundary for a VLAN.";
            }
            leaf upper-vlan {
              type uint16;
              mandatory true;
              description
                "Upper boundary for a VLAN.";
            }
          }
          case operator {
            leaf operator {
              type packet-fields:operator;
              default "eq";
              description
                "Operator to be applied on the VLAN below.";
            }
            leaf-list vlan {
              type uint16;
              description
                "VLAN number along with the operator on which to
                 match.";
              reference
                "IEEE Std 802.1Q: Bridges and Bridged Networks";
            }
          }
        }
      }
      container isid-filter {
        if-feature "match-on-isid-filter";
        description
          "Indicates how to handle I-SID filters.  The
           I-component is responsible for mapping customer
           Ethernet traffic to the appropriate I-SID.";
        choice isid-type {
          description
            "I-SID definition from range or operator.";
```

```
        case range {
          leaf lower-isid {
            type uint16;
            must '. <= ../upper-isid' {
              error-message
                "The lower-isid must be less than or equal to
                 the upper-isid.";
            }
            mandatory true;
            description
              "Lower boundary for an I-SID.";
          }
          leaf upper-isid {
            type uint16;
            mandatory true;
            description
              "Upper boundary for an I-SID.";
          }
        }
        case operator {
          leaf operator {
            type packet-fields:operator;
            default "eq";
            description
              "Operator to be applied on the I-SID below.";
          }
          leaf-list isid {
            type uint16;
            description
              "I-SID number along with the operator on which to
               match.";
            reference
              "IEEE 802.1ah: Provider Backbone Bridges";
          }
        }
      }
    }
  }

  augment "/acl:acls/acl:acl/acl:aces"
        + "/acl:ace/acl:matches/acl:l3/acl:ipv4/acl:ipv4" {
    description
      "Handle non-initial and initial fragments for IPv4 packets.";
    container ipv4-fragment {
      must 'not(../acl:flags)' {
        error-message
          "Either flags or fragment should be provided, but not
           both.";
      }
      description
        "Indicates how to handle IPv4 fragments.";
      uses fragment-fields;
    }
    leaf source-ipv4-prefix-list {
      type ipv4-prefix-set-ref;
      description
        "A reference to an IPv4 prefix list to match the source
         address.";
```

```
    }
    leaf destination-ipv4-prefix-list {
      type ipv4-prefix-set-ref;
      description
        "A reference to a prefix list to match the destination
         address.";
    }
    leaf protocol-set {
      type protocol-set-ref;
      description
        "A reference to a protocol set to match the protocol
         field.";
    }
  }

  augment "/acl:acls/acl:acl/acl:aces"
        + "/acl:ace/acl:matches/acl:l3/acl:ipv6/acl:ipv6" {
    description
      "Handles non-initial and initial fragments for IPv6 packets.";
    container ipv6-fragment {
      description
        "Indicates how to handle IPv6 fragments.";
      uses fragment-fields;
    }
    leaf source-ipv6-prefix-list {
      type ipv6-prefix-set-ref;
      description
        "A reference to a prefix list to match the source address.";
    }
    leaf destination-ipv6-prefix-list {
      type ipv6-prefix-set-ref;
      description
        "A reference to a prefix list to match the destination
         address.";
    }
    leaf protocol-set {
      type protocol-set-ref;
      description
        "A reference to a protocol set to match the next-header
         field.";
    }
    leaf extension-header {
      type iana-ipv6-ext-types:ipv6-extension-header-type;
      description
        "IPv6 extension header value.";
    }
  }

  augment "/acl:acls/acl:acl/acl:aces"
        + "/acl:ace/acl:matches/acl:l4/acl:tcp/acl:tcp" {
    description
      "Handles TCP flags and port sets.";
    container flags-bitmask {
      must 'not(../acl:flags)' {
        error-message
          "Either flags or flags-bitmask should be provided, but not
           both.";
      }
```

```
      description
        "Indicates how to handle TCP flags.";
      uses tcp-flags;
    }
    leaf source-tcp-port-set {
      type port-set-ref;
      description
        "A reference to a port set to match the source port.";
    }
    leaf destination-tcp-port-set {
      type port-set-ref;
      description
        "A reference to a port set to match the destination port.";
    }
  }

  augment "/acl:acls/acl:acl/acl:aces"
        + "/acl:ace/acl:matches/acl:l4/acl:udp/acl:udp" {
    description
      "Handle UDP port sets.";
    leaf source-udp-port-set {
      type port-set-ref;
      description
        "A reference to a port set to match the source port.";
    }
    leaf destination-udp-port-set {
      type port-set-ref;
      description
        "A reference to a port set to match the destination port.";
    }
  }

  augment "/acl:acls/acl:acl/acl:aces"
        + "/acl:ace/acl:matches/acl:l4/acl:icmp/acl:icmp" {
    description
      "Handle ICMP type sets.";
    leaf icmpv4-set {
      type icmpv4-type-set-ref;
      description
        "A reference to an ICMPv4 type set to match the ICMPv4 type
         field.";
    }
    leaf icmpv6-set {
      type icmpv6-type-set-ref;
      description
        "A reference to an ICMPv6 type set to match the ICMPv6 type
         field.";
    }
  }

  augment "/acl:acls/acl:acl/acl:aces"
        + "/acl:ace/acl:actions" {
    description
      "Complementary actions including rate-limit action.";
    uses acl-complementary-actions;
    leaf rate-limit {
      when "../acl:forwarding = 'acl:accept'" {
        description
```

```
            "Rate-limit valid only when the accept action is used.";
        }
        type decimal64 {
          fraction-digits 2;
        }
        units "bytes per second";
        description
          "Indicates a rate-limit for the matched traffic.";
      }
    }
  }

<CODE ENDS>
```

5. Security Considerations

This section is modeled after the template described in Section 3.7.1 of [YANG-GUIDELINES].

The "ietf-acl-enh" YANG module defines a data model that is designed to be accessed via YANG-based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. These YANG-based management protocols (1) have to use a secure transport layer (e.g., SSH [RFC4252], TLS [RFC8446], and QUIC [RFC9000]) and (2) have to use mutual authentication.

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). All writable data nodes are likely to be reasonably sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) and delete operations to these data nodes without proper protection or authentication can have a negative effect on network operations. The following subtrees and data nodes have particular sensitivities/vulnerabilities:

'defined-sets': These lists specify a set of IP addresses, port numbers, protocols, ICMP types, and aliases. Similar to [RFC8519], unauthorized write access to these lists can allow intruders to modify the entries to permit traffic that should not be permitted or deny traffic that should be permitted. The former may result in a DoS attack or compromise a device. The latter may result in a DoS attack.

These sets are defined with "nacm:default-deny-write" tagging.

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. Specifically, the following subtrees and data nodes have particular sensitivities/vulnerabilities:

'defined-sets':

Unauthorized read access of these lists will allow an attacker to identify the actual resources that are bound to ACLs. Likewise, access to this information will help an attacker to better scope its attacks to target resources that are specific to a given network instead of performing random scans. Also, disclosing some of this information (e.g., IP addresses of core routers) may nullify the effect of topology-hiding strategies in some networks.

There are no particularly sensitive RPC or action operations.

The document defines a match policy based on a pattern that can be observed in a packet. For example, such a policy can be combined with header-based matches in the context of DDoS mitigation. Filtering based on a pattern match is deterministic for packets with unencrypted data. However, the efficiency for encrypted packets depends on the presence of an unvarying pattern. Readers may also refer to Section 11 of [RFC8329] for security considerations related to Network Security Functions (NSFs) that apply packet content matching.

The YANG modules "iana-icmpv4-types", "iana-icmpv6-types", and "iana- ipv6-ext-types" define a set of types. These nodes are intended to be reused by other YANG modules. Each module by itself does not expose any data nodes that are writable, data nodes that contain read-only state, or RPCs. As such, there are no additional security issues related to these YANG modules that need to be considered.

6. IANA Considerations

6.1. URI Registrations

IANA has registered the following URIs in the "ns" registry within the "IETF XML Registry" [RFC3688]:

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

URI: urn:ietf:params:xml:ns:yang:iana-icmpv4-types
Registrant Contact: The IESG
XML: N/A; the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:iana-icmpv6-types
Registrant Contact: The IESG
XML: N/A; the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:iana-ipv6-ext-types
Registrant Contact: The IESG
XML: N/A; the requested URI is an XML namespace.

6.2. YANG Module Name Registrations

IANA has registered the following YANG modules in the "YANG Module Names" registry [RFC6020] [RFC9890] within the "YANG Parameters" registry group.

Name: ietf-acl-enh
Maintained by IANA: N
Namespace: urn:ietf:params:xml:ns:yang:ietf-acl-enh
Prefix: acl-enh
Reference: RFC 9899

Name: iana-icmpv4-types
Maintained by IANA: Y
Namespace: urn:ietf:params:xml:ns:yang:iana-icmpv4-types
Prefix: iana-icmpv4-types
Reference: RFC 9899

Name: iana-icmpv6-types
Maintained by IANA: Y
Namespace: urn:ietf:params:xml:ns:yang:iana-icmpv6-types
Prefix: iana-icmpv6-types
Reference: RFC 9899

Name: iana-ipv6-ext-types
Maintained by IANA: Y
Namespace: urn:ietf:params:xml:ns:yang:iana-ipv6-ext-types
Prefix: iana-ipv6-ext-types
Reference: RFC 9899

6.3. Considerations for IANA-Maintained Modules

6.3.1. ICMPv4 Types IANA Module

This document creates the initial version of the IANA-maintained "iana-icmpv4-types" YANG module. The most recent version of the YANG module is available in the "YANG Parameters" registry group [IANA-YANG-PARAMETERS].

IANA has added this note to the registry:

> New values must not be directly added to the "iana-icmpv4-types" YANG module. They must instead be added to the "ICMP Type Numbers" registry [IANA-ICMPv4].

When a value is added to the "ICMP Type Numbers" registry, a new "enum" statement must be added to the "iana-icmpv4-types" YANG module. The "enum" statement, and substatements thereof, should be defined as follows:

"enum": Replicates the name from the registry with all illegal characters (e.g., spaces) stripped.

"value": Contains the decimal value of the IANA-assigned value.

"status": Included only if a registration has been deprecated or obsoleted. IANA "deprecated" maps to YANG status "deprecated", and IANA "obsolete" maps to YANG status "obsolete".

"description": Replicates the name from the registry.

"reference": Replicates the reference(s) from the registry with the title of the document(s) added.

Unassigned, reserved, or values styled like those in [RFC3692] are not present in the module.

When the "iana-icmpv4-types" YANG module is updated, a new "revision" statement with a unique revision date must be added in front of the existing revision statements.

IANA has added this note to "ICMP Type Numbers" registry [IANA-ICMPv4] and listed this document as an additional reference for the registry:

> When this registry is modified, the YANG module "iana-icmpv4-types" [IANA-YANG-PARAMETERS] must be updated as defined in RFC 9899.

6.3.2. ICMPv6 Types IANA Module

This document creates the initial version of the IANA-maintained "iana-icmpv6-types" YANG module. The most recent version of the YANG module is available in the "YANG Parameters" registry group [IANA-YANG-PARAMETERS].

IANA has added this note to the registry:

> New values must not be directly added to the "iana-icmpv6-types" YANG module. They must instead be added to the "ICMPv6 "type" Numbers" registry [IANA-ICMPv6].

When a value is added to the "ICMPv6 "type" Numbers" registry, a new "enum" statement must be added to the "iana-icmpv6-types" YANG module. The "enum" statement, and substatements thereof, should be defined as follows:

"enum": Replicates the name from the registry with all illegal characters (e.g., spaces) stripped.

"value": Contains the decimal value of the IANA-assigned value.

"status": Included only if a registration has been deprecated or obsoleted. IANA "deprecated" maps to YANG status "deprecated", and IANA "obsolete" maps to YANG status "obsolete".

"description": Replicates the name from the registry.

"reference": Replicates the reference(s) from the registry with the title of the document(s) added.

Unassigned, reserved, or private experimentation values are not present in the module.

When the "iana-icmpv6-types" YANG module is updated, a new "revision" statement with a unique revision date must be added in front of the existing revision statements.

IANA has added this note to the "ICMPv6 "type" Numbers" registry [IANA-ICMPv6] and listed this document as an additional reference for the registry:

> When this registry is modified, the YANG module "iana-icmpv6-types" [IANA-YANG-PARAMETERS] must be updated as defined in RFC 9899.

6.3.3. IPv6 Extension Header Types IANA Module

This document creates the initial version of the IANA-maintained "iana-ipv6-ext-types" YANG module. The most recent version of the YANG module is available in the "YANG Parameters" registry group [IANA-YANG-PARAMETERS].

IANA has added this note to the registry:

> New values must not be directly added to the "iana-ipv6-ext-types" YANG module. They must instead be added to the "IPv6 Extension Header Types" registry [IANA-IPv6].

When a value is added to the "IPv6 Extension Header Types" registry, a new "enum" statement must be added to the "iana-ipv6-ext-types" YANG module. The "enum" statement, and substatements thereof, should be defined as follows

"enum": Replicates the description from the registry with all spaces stripped.

"value": Contains the decimal value of the IANA-assigned value.

"status": Included only if a registration has been deprecated or obsoleted. IANA "deprecated" maps to YANG status "deprecated", and IANA "obsolete" maps to YANG status "obsolete".

"description": Replicates the description from the registry.

"reference": Replicates the reference(s) from the registry with the title of the document(s) added.

Unassigned or reserved values are not present in the module.

When the "iana-ipv6-ext-types" YANG module is updated, a new "revision" statement with a unique revision date must be added in front of the existing revision statements.

IANA has added this note to the "IPv6 Extension Header Types" registry [IANA-IPv6] and listed this document as an additional reference for the registry:

> When this registry is modified, the YANG module "iana-ipv6-ext-types" [IANA-YANG-PARAMETERS] must be updated as defined in RFC 9899.

7. References

7.1. Normative References

[IANA-ICMPv4] IANA, "ICMP Type Numbers", <https://www.iana.org/assignments/icmp-parameters>.

[IANA-ICMPv6] IANA, "ICMPv6 "type" Numbers", <https://www.iana.org/assignments/icmpv6-parameters>.

[IANA-IPv6] IANA, "IPv6 Extension Header Types", <https://www.iana.org/assignments/ipv6-parameters>.

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <https://www.rfc-editor.org/info/rfc792>.

[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>.

[RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, DOI 10.17487/RFC3032, January 2001, <https://www.rfc-editor.org/info/rfc3032>.

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

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <https://www.rfc-editor.org/info/rfc4443>.

[RFC5462] Andersson, L. and R. Asati, "Multiprotocol Label Switching (MPLS) Label Stack Entry: "EXP" Field Renamed to "Traffic Class" Field", RFC 5462, DOI 10.17487/RFC5462, February 2009, <https://www.rfc-editor.org/info/rfc5462>.

[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>.

[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>.

[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>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8294] Liu, X., Qu, Y., Lindem, A., Hopps, C., and L. Berger, "Common YANG Data Types for the Routing Area", RFC 8294, DOI 10.17487/RFC8294, December 2017, <https://www.rfc-editor.org/info/rfc8294>.

[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>.

[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>.

[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>.

7.2. Informative References

[IANA-TCP-FLAGS] IANA, "Transmission Control Protocol (TCP) Parameters", <https://www.iana.org/assignments/tcp-parameters>.

[IANA-YANG-PARAMETERS] IANA, "YANG Parameters", <https://www.iana.org/assignments/yang-parameters>.

[IEEE-802-1ah] IEEE, "IEEE Standard for Local and metropolitan area networks -- Virtual Bridged Local Area Networks Amendment 7: Provider Backbone Bridges", IEEE Std 802.1ah-2008, DOI 10.1109/IEEESTD.2008.4602826, August 2008, <https://doi.org/10.1109/IEEESTD.2008.4602826>.

[IEEE802.1Qcp] IEEE, "IEEE Standard for Local and metropolitan area networks--Bridges and Bridged Networks--Amendment 30: YANG Data Model", IEEE Std 802.1Qcp-2018, DOI 10.1109/IEEESTD.2018.8467507, September 2018, <https://doi.org/10.1109/IEEESTD.2018.8467507>.

[RFC3692] Narten, T., "Assigning Experimental and Testing Numbers Considered Useful", BCP 82, RFC 3692, DOI 10.17487/RFC3692, January 2004, <https://www.rfc-editor.org/info/rfc3692>.

[RFC4252] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Authentication Protocol", RFC 4252, DOI 10.17487/RFC4252, January 2006, <https://www.rfc-editor.org/info/rfc4252>.

[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>.

[RFC7209] Sajassi, A., Aggarwal, R., Uttaro, J., Bitar, N., Henderickx, W., and A. Isaac, "Requirements for Ethernet VPN (EVPN)", RFC 7209, DOI 10.17487/RFC7209, May 2014, <https://www.rfc-editor.org/info/rfc7209>.

[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>.

[RFC8329] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", RFC 8329, DOI 10.17487/RFC8329, February 2018, <https://www.rfc-editor.org/info/rfc8329>.

[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>.

[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>.

[RFC8955] Loibl, C., Hares, S., Raszuk, R., McPherson, D., and M. Bacher, "Dissemination of Flow Specification Rules", RFC 8955, DOI 10.17487/RFC8955, December 2020, <https://www.rfc-editor.org/info/rfc8955>.

[RFC8956] Loibl, C., Ed., Raszuk, R., Ed., and S. Hares, Ed., "Dissemination of Flow Specification Rules for IPv6", RFC 8956, DOI 10.17487/RFC8956, December 2020, <https://www.rfc-editor.org/info/rfc8956>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

[RFC9132] Boucadair, M., Ed., Shallow, J., and T. Reddy.K, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification", RFC 9132, DOI 10.17487/RFC9132, September 2021, <https://www.rfc-editor.org/info/rfc9132>.

[RFC9890] Bierman, A., Boucadair, M., Ed., and Q. Wu, "An Update to YANG Module Names Registration", RFC 9890, DOI 10.17487/RFC9890, October 2025, <https://www.rfc-editor.org/info/rfc9890>.

[YANG-GUIDELINES] Bierman, A., Boucadair, M., and Q. Wu, "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", Work in Progress, Internet-Draft, draft-ietf-netmod-rfc8407bis-28, 5 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-rfc8407bis-28>.

[YANG-XSLT] "iana-yang", commit 3a6cb69, December 2021, <https://github.com/llhotka/iana-yang>.

Appendix A. Problem Statement and Gap Analysis

A.1. Suboptimal Configuration: Lack of Support for Lists of Prefixes

IP prefix-related data nodes (e.g., "destination-ipv4-network" or "destination-ipv6-network") do not support handling a list of IP prefixes, which may then lead to having to support large numbers of ACL entries in a configuration file.

The same issue is encountered when ACLs have to be in place to mitigate DDoS attacks that involve a set of sources (e.g., [RFC9132]). The situation is even worse when both a list of sources and destination prefixes are involved in the filtering.

Figure 3 shows an example of the required ACL configuration for filtering traffic from two prefixes.

```
{
  "ietf-access-control-list:acls": {
    "acl": [
      {
        "name": "first-prefix",
        "type": "ipv6-acl-type",
        "aces": {
          "ace": [
            {
              "name": "my-test-ace",
              "matches": {
                "ipv6": {
                  "destination-ipv6-network":
                    "2001:db8:6401:1::/64",
                  "source-ipv6-network":
                    "2001:db8:1234::/96",
                  "protocol": 17,
                  "flow-label": 10000
                },
                "udp": {
                  "source-port": {
                    "operator": "lte",
                    "port": 80
                  },
                  "destination-port": {
                    "operator": "neq",
                    "port": 1010
                  }
                }
              },
              "actions": {
                "forwarding": "accept"
              }
            }
          ]
        }
      },
      {
        "name": "second-prefix",
        "type": "ipv6-acl-type",
        "aces": {
          "ace": [
            {
              "name": "my-test-ace",
              "matches": {
                "ipv6": {
                  "destination-ipv6-network":
                    "2001:db8:6401:c::/64",
                  "source-ipv6-network":
                    "2001:db8:1234::/96",
                  "protocol": 17,
                  "flow-label": 10000
                },
                "udp": {
                  "source-port": {
                    "operator": "lte",
                    "port": 80
```

```
                  },
                  "destination-port": {
                    "operator": "neq",
                    "port": 1010
                  }
                }
              },
              "actions": {
                "forwarding": "accept"
              }
            }
          ]
        }
      }
    ]
  }
}
```

Figure 3: Example Illustrating Suboptimal Use of the ACL Model with a Prefix List (Message Body)

Such a configuration is suboptimal for both:

- network controllers that need to manipulate large files, as all or a subset for this configuration will need to be passed to the underlying network devices, and
- devices that may receive such a configuration and thus will need to maintain it locally.

A.2. Manageability: Impossibility of Using Aliases or Defined Sets

The same approach as the one discussed for IP prefixes can be generalized by introducing the concept of "aliases" or "defined sets".

The defined sets are reusable definitions across several ACLs. Each category is modeled in YANG as a list of parameters related to the class it represents. The following sets can be considered:

Prefix sets: Used to create lists of IPv4 or IPv6 prefixes.

Protocol sets: Used to create a list of protocols.

Port number sets: Used to create lists of TCP or UDP port values (or any other transport protocol that makes uses of port numbers). The identity of the protocols is identified by the protocol set, if present. Otherwise, a set applies to any protocol.

ICMP sets: Used to create lists of ICMP-based filters. This applies only when the protocol is set to ICMP or ICMPv6.

Aliases may also be considered to manage resources that are identified by a combination of various parameters (e.g., prefix, protocol, port number, FQDN, or VLAN IDs). Note that some aliases can be provided by decomposing them into separate sets.

A.3. Bind ACLs to Devices, Not Only Interfaces

In the context of network management, an ACL may be enforced in many network locations. As such, the ACL module should allow for binding an ACL to multiple devices, not only (abstract) interfaces.

Thus, the ACL name must be unique at the scale of the network, but the same name may be used in many devices when enforcing node-specific ACLs.

A.4. Partial or Lack of IPv4/IPv6 Fragment Handling

[RFC8519] does not support fragment handling for IPv6 but offers a partial support for IPv4 through the use of 'flags'. Nevertheless, the use of 'flags' is problematic since it does not allow a bitmask to be defined. For example, setting other bits not covered by the 'flags' filtering clause in a packet will allow that packet to get through (because it won't match the ACE).

Defining a new IPv4/IPv6 matching field called 'fragment' is thus required to efficiently handle fragment-related filtering rules.

A.5. Suboptimal TCP Flags Handling

[RFC8519] supports including flags in the TCP match fields; however, that structure does not support matching operations as those supported in BGP Flow Spec. Defining this field as a flag bitmask together with a set of operations is meant to efficiently handle TCP flags filtering rules.

A.6. Rate-Limit Action

[RFC8519] specifies that forwarding actions can be 'accept' (i.e., accept matching traffic), 'drop' (i.e., drop matching traffic without sending any ICMP error message), or 'reject' (i.e., drop matching traffic and send an ICMP error message to the source). However, there are situations where the matching traffic can be accepted, but with a rate-limit policy. This capability is not supported by [RFC8519].

A.7. Payload-Based Filtering

Some transport protocols use existing protocols (e.g., TCP or UDP) as substrate. The match criteria for such protocols may rely upon the 'protocol' under 'l3', TCP/UDP match criteria, part of the TCP/UDP payload, or a combination thereof. [RFC8519] does not support matching based on the payload.

Likewise, the ACL model defined in [RFC8519] does not support filtering of encapsulated traffic.

A.8. Reuse the Content of ACLs Across Several Devices

Having a global network view of the ACLs is highly valuable for service providers. An ACL could be defined and applied based on the network topology hierarchy. Therefore, an ACL can be defined at the network level, and then that same ACL can be used in (or referenced to) several devices (including termination points) within the same network.

This network/device differentiation of ACLs introduces several new requirements, for example:

- An ACL name can be used at both network and device levels.
- An ACL content updated at the network level should imply a transaction that updates the relevant content in all the nodes using this ACL.
- ACLs defined at the device level have a local meaning for the specific node.
- A device can be associated with a router, a Virtual Routing and Forwarding (VRF), a logical system, or a virtual node. ACLs can be applied in physical and logical infrastructure.

A.9. Match MPLS Headers

The ACLs can be used to create rules to match MPLS fields on a packet. [RFC8519] does not support such function.

Appendix B. Examples

This section provides a few examples to illustrate the use of the enhanced ACL module ("ietf-acl-enh").

B.1. TCP Flags Handling

Figure 4 shows an example of the message body of a request to install a filter to discard incoming TCP messages having all flags unset.

```
{
  "ietf-access-control-list:acls": {
    "acl": [
      {
        "name": "tcp-flags-example",
        "aces": {
          "ace": [
            {
              "name": "null-attack",
              "matches": {
                "tcp": {
                  "ietf-acl-enh:flags-bitmask": {
                    "operator": "not any",
                    "bitmask": 4095
                  }
                }
              },
              "actions": {
                "forwarding": "drop"
              }
            }
          ]
        }
      }
    ]
  }
}
```

Figure 4: Example of an ACL to Deny TCP Null Attack Messages (Request Body)

B.2. Fragments Handling

Figure 5 shows the content of a POST request to allow the traffic destined to 198.51.100.0/24 and UDP port number 53, but to drop all fragmented packets. The following ACEs are defined (in this order):

"drop-all-fragments" ACE: discards all fragments.

"allow-dns-packets" ACE: accepts DNS packets destined to 198.51.100.0/24.

```
{
  "ietf-access-control-list:acls": {
    "acl": [
      {
        "name": "dns-fragments",
        "type": "ipv4-acl-type",
        "aces": {
          "ace": [
            {
              "name": "drop-all-fragments",
              "matches": {
                "ipv4": {
                  "ietf-acl-enh:ipv4-fragment": {
                    "operator": "match",
                    "type": "isf"
                  }
                }
              },
              "actions": {
                "forwarding": "drop"
              }
            },
            {
              "name": "allow-dns-packets",
              "matches": {
                "ipv4": {
                  "destination-ipv4-network": "198.51.100.0/24"
                },
                "udp": {
                  "destination-port": {
                    "operator": "eq",
                    "port": 53
                  }
                }
              },
              "actions": {
                "forwarding": "accept"
              }
            }
          ]
        }
      }
    ]
  }
}
```

Figure 5: Example Illustrating Candidate Filtering of IPv4 Fragmented Packets (Message Body)

Figure 6 shows an example of the body of a POST request to allow the traffic destined to 2001:db8::/32 and UDP port number 53, but to drop all fragmented packets. The following ACEs are defined (in this order):

"drop-all-fragments" ACE: discards all fragments (including atomic fragments). That is, IPv6 packets that include a Fragment header (44) are dropped.

"allow-dns-packets" ACE: accepts DNS packets destined to 2001:db8::/32.

```
{
  "ietf-access-control-list:acls": {
    "acl": [
      {
        "name": "dns-fragments",
        "type": "ipv6-acl-type",
        "aces": {
          "ace": [
            {
              "name": "drop-all-fragments",
              "matches": {
                "ipv6": {
                  "ietf-acl-enh:ipv6-fragment": {
                    "operator": "match",
                    "type": "isf"
                  }
                }
              },
              "actions": {
                "forwarding": "drop"
              }
            },
            {
              "name": "allow-dns-packets",
              "matches": {
                "ipv6": {
                  "destination-ipv6-network": "2001:db8::/32"
                },
                "udp": {
                  "destination-port": {
                    "operator": "eq",
                    "port": 53
                  }
                }
              },
              "actions": {
                "forwarding": "accept"
              }
            }
          ]
        }
      }
    ]
  }
}
```

Figure 6: An Example Illustrating Filtering of IPv6 Fragmented Packets (Message Body)

B.3. Pattern-Based Filtering

Pattern-based filtering is useful to detect specific patterns, signatures, or encapsulated packets. Figure 7 shows an example of the message body of a request to install a filter to discard IP-in-IP encapsulated messages with an inner destination IP address equal to "2001:db8::1". By using the offset at the end of Layer 3, the rule targets a specific portion of the payload that starts 20 bytes after the beginning of the data (that is, skipping the first 20 bytes of the inner IPv6 header).

For the reader's convenience, the textual representation of the pattern is used in the example instead of the binary form.

```
{
  "ietf-access-control-list:acls": {
    "acl": [
      {
        "name": "pattern-example",
        "aces": {
          "ace": [
            {
              "name": "pattern-1",
              "matches": {
                "ipv6": {
                  "protocol": 41
                },
                "ietf-acl-enh:pattern": {
                  "offset": "ietf-acl-enh:layer4",
                  "length": 20,
                  "operator": "match",
                  "pattern": "2001:db8::1"
                }
              },
              "actions": {
                "forwarding": "drop"
              }
            }
          ]
        }
      }
    ]
  }
}
```

Figure 7: Example of an ACL to Deny Encapsulated Messages with a Specific Inner Destination Address (Request Body)

B.4. VLAN Filtering

Figure 8 shows an ACL example to illustrate how to apply a VLAN range filter.

```
{
  "ietf-access-control-list:acls": {
    "acl": [
      {
        "name": "VLAN_FILTER",
        "aces": {
          "ace": [
            {
              "name": "1",
              "matches": {
                "ietf-acl-enh:vlan-filter": {
                  "lower-vlan": 10,
                  "upper-vlan": 20
                }
              },
              "actions": {
                "forwarding": "ietf-access-control-list:accept"
              }
            }
          ]
        }
      }
    ]
  }
}
```

Figure 8: Example of VLAN Filter (Message Body)

B.5. I-SID Filtering

Figure 9 shows an ACL example to illustrate the I-SID range filtering.

```
{
  "ietf-access-control-list:acls": {
    "acl": [
      {
        "name": "test",
        "aces": {
          "ace": [
            {
              "name": "1",
              "matches": {
                "ietf-acl-enh:isid-filter": {
                  "lower-isid": 100,
                  "upper-isid": 200
                }
              },
              "actions": {
                "forwarding": "ietf-access-control-list:accept"
              }
            }
          ]
        }
      }
    ]
  }
}
```

Figure 9: Example I-SID Filter (Message Body)

B.6. Rate-Limit

Figure 10 shows an ACL example to rate-limit incoming SYNs during a SYN flood attack.

```
{
  "ietf-access-control-list:acls": {
    "acl": [
      {
        "name": "tcp-flags-example-with-rate-limit",
        "aces": {
          "ace": [
            {
              "name": "rate-limit-syn",
              "matches": {
                "tcp": {
                  "ietf-acl-enh:flags-bitmask": {
                    "operator": "match",
                    "bitmask": 2
                  }
                }
              },
              "actions": {
                "forwarding": "accept",
                "ietf-acl-enh:rate-limit": "20.00"
              }
            }
          ]
        }
      }
    ]
  }
}
```

Figure 10: An Example of Rate-Limiting Incoming TCP SYNs (Message Body)

Acknowledgments

Many thanks to Jon Shallow and Miguel Cros for their review and comments on this document.

Thanks to Qiufang Ma, Victor Lopez, Joe Clarke, and Mahesh Jethanandani for their comments and suggestions.

Thanks to Lou Berger for shepherding this document.

Thanks to David Black for the tsvart review, Tim Wicinski for the intdir review, Per Andersson for the yangdoctors review, Russ Housley for the genart review, and Linda Dunbar and Sean Turner for the secdir reviews.

Thanks to Erik Kline, Mike Bishop, Éric Vyncke, Roman Danyliw, and Deb Cooley for the IESG review.

The IANA-maintained modules were generated using an XSLT stylesheet from the 'iana-yang' project [YANG-XSLT].

This work is partially supported by the European Commission under Horizon 2020 Secured autonomic traffic management for a Tera of SDN flows (Teraflow) project (grant agreement number 101015857).

Authors' Addresses

Oscar Gonzalez de Dios
Telefonica
Email: oscar.gonzalezdedios@telefonica.com

Samier Barguil
Nokia
Email: samier.barguil_giraldo@nokia.com

Mohamed Boucadair
Orange
Email: mohamed.boucadair@orange.com

Qin Wu
Huawei
Email: bill.wu@huawei.com