

NETMOD WG
Internet-Draft
Intended status: Standards Track
Expires: August 9, 2015

D. Bogdanovic
Juniper Networks
K. Sreenivasa
Brocade Communications System
L. Huang
D. Blair
Cisco Systems
February 05, 2015

Network Access Control List (ACL) YANG Data Model
draft-ietf-netmod-acl-model-01

Abstract

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

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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 August 9, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in [Section 4](#).e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	2
1.1.	Definitions and Acronyms	3
2.	Problem Statement	3
3.	Design of the ACL Model	3
3.1.	ACL Modules	4
4.	ACL YANG Models	6
4.1.	IETF-ACL module	6
4.2.	Packet Header module	11
4.3.	A company proprietary module example	15
4.4.	An ACL Example	17
4.5.	Port Range Usage Example	18
5.	Example of extending existing model for route filtering . . .	19
6.	Linux nftables	21
7.	Security Considerations	21
8.	IANA Considerations	22
9.	Acknowledgements	22
10.	Change log [RFC Editor: Please remove]	23
11.	References	23
11.1.	Normative References	23
11.2.	Informative References	23
	Authors' Addresses	23

[1.](#) Introduction

Access Control List (ACL) is one of the basic elements to configure device forwarding behavior. It is used in many networking concepts 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 metadata match criteria.

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

- o 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 innovations of the networked devices.

1.1. Definitions and Acronyms

ACE: Access Control Entry

ACL: Access Control List

AFI: Address Field Identifier

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 reused between vendors and between applications.

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

3. Design of the ACL Model

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 and metadata matching. 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 overall operational state for the ACL and operational state for each ACE, targets where the ACL applied. 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).

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 very simple and with this design we hope to achieve needed flexibility for each vendor to extend the base model.

3.1. ACL Modules

There are three YANG modules in the model. The first module, "ietf-acl", 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, "packet-headers". The match container in "ietf-acl" uses groupings in "packet-headers". The "packet-headers" modules can easily be extended to reuse definitions from other modules such as IPFIX [[RFC5101](#)] or migrate proprietary augmented module definitions into the standard module.

```
module: ietf-acl
  +-rw access-lists
    +-rw access-list* [acl-name]
      +-rw acl-name          string
      +-rw acl-type?         acl-type
      +-ro acl-oper-data
      | +-ro match-counter?  ietf:counter64
      | +-ro targets*        string
      +-rw access-list-entries
        +-rw access-list-entry* [rule-name]
          +-rw rule-name      string
```



```

+--rw matches
| +--rw (ace-type)?
| | +--:(ace-ip)
| | | +--rw source-port-range
| | | | +--rw lower-port    inet:port-number
| | | | +--rw upper-port?  inet:port-number
| | | +--rw destination-port-range
| | | | +--rw lower-port    inet:port-number
| | | | +--rw upper-port?  inet:port-number
| | | +--rw dscp?          inet:dscp
| | | +--rw protocol?      uint8
| | | +--rw (ace-ip-version)?
| | |   +--:(ace-ipv4)
| | |   | +--rw destination-ipv4-address?
| | |   |                               inet:ipv4-prefix
| | |   | +--rw source-ipv4-address?
| | |   |                               inet:ipv4-prefix
| | |   +--:(ace-ipv6)
| | |   | +--rw destination-ipv6-address?
| | |   |                               inet:ipv6-prefix
| | |   | +--rw source-ipv6-address?
| | |   |                               inet:ipv6-prefix
| | |   | +--rw flow-label?    inet:ipv6-flow-label
| | +--:(ace-eth)
| | | +--rw destination-mac-address?
| | |                                     yang:mac-address
| | | +--rw destination-mac-address-mask?
| | |                                     yang:mac-address
| | | +--rw source-mac-address?
| | |                                     yang:mac-address
| | | +--rw source-mac-address-mask?
| | |                                     yang:mac-address
| +--rw input-interface?
| | +--rw absolute
| | | +--rw start?    yang:date-and-time
| | | +--rw end?      yang:date-and-time
| | | +--rw active?   boolean
+--rw actions
| +--rw (packet-handling)?
| | +--:(deny)
| | | +--rw deny?    empty
| | +--:(permit)
| | | +--rw permit?  empty
+--ro ace-oper-data
  +--ro match-counter?  ietf:counter64

```


Module "newco-acl" is an example of company proprietary model, that augments "ietf-acl" module. It shows how 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. "newco-acl" is just an example and it is expected from vendors to create their own proprietary models.

```

module: newco-acl
augment /ietf-acl:access-list/ietf-acl:access-list-entries/ietf-acl:matches:
  +--rw (protocol_payload_choice)?
    +--:(protocol_payload)
      +--rw protocol_payload* [value_keyword]
        +--rw value_keyword      enumeration
augment /ietf-acl:access-list/ietf-acl:access-list-entries/ietf-acl:actions:
  +--rw (action)?
    +--:(count)
      | +--rw count?          string
    +--:(policer)
      | +--rw policer?        string
    +--:(hierarchical-policer)
      +--rw hierarchitac1-policer?  string
augment /ietf-acl:access-lists/ietf-acl:access-list:
  +--rw default-actions
  +--rw deny?      empty

```

[4. ACL YANG Models](#)

[4.1. IETF-ACL module](#)

"ietf-acl" is the standard top level module for Access lists. It has a container for "access-list" to store access list information. This container 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" have containers defining "matches" and "actions". The "matches" define criteria used to identify patterns in "packet-fields". The "actions" define behavior to undertake once a "match" has been identified.

```

module ietf-acl {
  yang-version 1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-acl";

  prefix acl;

  import ietf-yang-types {
    prefix "ietf";
  }

```



```
}

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

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

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

    WG Chair: Juergen Schoenwaelder
    j.schoenwaelder@jacobs-university.de

    WG Chair: Tom Nadeau
    tnadeau@lucidvision.com

    Editor: Dean Bogdanovic
    deanb@juniper.net

    Editor: Kiran Agrahara Sreenivasa
    kkoushik@brocade.com

    Editor: Lisa Huang
    yihuan@cisco.com

    Editor: Dana Blair
    dblair@cisco.com";

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

revision 2014-10-10 {
    description "Creating base model for netmod.";
    reference
        "RFC 6020: YANG - A Data Modeling Language for the
        Network Configuration Protocol (NETCONF)";
}

identity acl-base {
    description "Base acl type for all ACL type identifiers.";
}

identity ip-acl {
    base "acl:acl-base";
```



```
        description "layer 3 ACL type";
    }
    identity eth-acl {
        base "acl:acl-base";
        description "layer 2 ACL type";
    }

    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 "/acl:access-lists/acl:access-list/acl:acl-name";
        }
        description "This type is used by data models that
            need to referenced an acl";
    }

    container access-lists {
        description
            "Access control lists.";

        list access-list {
            key acl-name;
            description "
                An access list (acl) is an ordered list of
                access list entries (ace). Each ace has a
                sequence number to define the order, list
                of match criteria, and a list of actions.
                Since there are several kinds of acls
                implemented with different attributes for
                each and different for each vendor, this
                model accomodates customizing acls for
                each kind and for each vendor.
            ";

            leaf acl-name {
                type string;
                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 ACL";
}

container acl-oper-data {
    config false;

    description "Overall ACL operational data";
    leaf match-counter {
        type ietf:counter64;
        description "Total match count for ACL";
    }

    leaf-list targets {
        type string;
        description "List of targets where ACL is applied";
    }
}

container access-list-entries {
    description "The access-list-entries container contains
        a list of access-list-entry(ACE).";

    list access-list-entry {
        key rule-name;
        ordered-by user;

        description "List of access list entries(ACE)";
        leaf rule-name {
            type string;
            description "Entry name.";
        }

        container matches {
            description "Define match criteria";
            choice ace-type {
                description "Type of ace.";
                case ace-ip {
                    uses packet-fields:acl-ip-header-fields;
                    choice ace-ip-version {
                        description "Choice of IP version.";
                        case ace-ipv4 {
                            uses packet-fields:acl-ipv4-header-fields;
                        }
                        case ace-ipv6 {
```



```
        uses packet-fields:acl-ipv6-header-fields;
    }
}
}
case ace-eth {
    uses packet-fields:acl-eth-header-fields;
}
}
uses packet-fields:metadata;
}

container actions {
    description "Define action criteria";
    choice packet-handling {
        default deny;

        description "Packet handling action.";
        case deny {
            leaf deny {
                type empty;
                description "Deny action.";
            }
        }
        case permit {
            leaf permit {
                type empty;
                description "Permit action.";
            }
        }
    }
}

container ace-oper-data {
    config false;

    description "Per ace operational data";
    leaf match-counter {
        type ietf:counter64;
        description "Number of matches for an ace";
    }
}
}
}
}
}
}
```


4.2. Packet Header module

The packet fields module defines the necessary groups for matching on fields in the packet including ethernet, ipv4, ipv6, transport layer fields and metadata. These groupings can be augmented to include other proprietary matching criteria. 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.

```
module packet-fields {  
  yang-version 1;  
  
  namespace "urn:ietf:params:xml:ns:yang:packet-fields";  
  
  prefix packet-fields;  
  
  import ietf-inet-types {  
    prefix "inet";  
  }  
  
  import ietf-yang-types {  
    prefix "yang";  
  }  
  
  organization  
    "IETF NETMOD (NETCONF Data Modeling Language) Working Group";  
  
  contact  
    "WG Web: http://tools.ietf.org/wg/netmod/  
    WG List: netmod@ietf.org  
  
    WG Chair: Juergen Schoenwaelder  
    j.schoenwaelder@jacobs-university.de  
  
    WG Chair: Tom Nadeau  
    tnadeau@lucidvision.com  
  
    Editor: Dean Bogdanovic  
    deanb@juniper.net  
  
    Editor: Kiran Agrahara Sreenivasa  
    kkoushik@brocade.com  
  
    Editor: Lisa Huang  
    yihuan@cisco.com  
  
    Editor: Dana Blair
```



```
    dblair@cisco.com";

description
  "This YANG module defines groupings that used by ietf-acl
  but not limited to acl.";

revision 2014-11-06 {
  description "Initial version of packet fields used by
    access-lists";
  reference
    "RFC 6020: YANG - A Data Modeling Language for the
    Network Configuration Protocol (NETCONF)";
}

grouping acl-transport-header-fields {
  description "Transport header fields";

  container source-port-range {
    description "inclusive range of source ports";
    leaf lower-port {
      type inet:port-number;
      mandatory true;
      description "Lower boundary.";
    }
    leaf upper-port {
      type inet:port-number;
      description "Upper boundary.";
    }
  }

  container destination-port-range {
    description "inclusive range of destination ports";
    leaf lower-port {
      type inet:port-number;
      mandatory true;
      description "Lower boundary.";
    }
    leaf upper-port {
      type inet:port-number;
      description "Upper boundary.";
    }
  }
}

grouping acl-ip-header-fields {
  description "Header fields common to ipv4 and ipv6";

  uses acl-transport-header-fields;
```



```
    leaf dscp {
      type inet:dscp;
      description "Value of dscp.";
    }

    leaf protocol {
      type uint8;
      description "Internet Protocol number.";
    }
  }

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

    leaf destination-ipv4-network {
      type inet:ipv4-prefix;
      description "One or more ip addresses.";
    }

    leaf source-ipv4-network {
      type inet:ipv4-prefix;
      description "One or more ip addresses.";
    }
  }

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

    leaf destination-ipv6-network {
      type inet:ipv6-prefix;
      description "One or more ip addresses.";
    }

    leaf source-ipv6-network {
      type inet:ipv6-prefix;
      description "One or more ip addresses.";
    }

    leaf flow-label {
      type inet:ipv6-flow-label;
      description "Flow label.";
    }
  }

  grouping acl-eth-header-fields {
```



```
    description "fields in ethernet header";

    leaf destination-mac-address {
        type yang:mac-address;
        description "Mac addresses.";
    }

    leaf destination-mac-address-mask {
        type yang:mac-address;
        description "Mac addresses mask.";
    }

    leaf source-mac-address {
        type yang:mac-address;
        description "Mac addresses.";
    }

    leaf source-mac-address-mask {
        type yang:mac-address;
        description "Mac addresses mask.";
    }
}

grouping timerange {
    description "Define time range entries to restrict
        the access. The time range is identified by a name
        and then referenced by a function, so that those
        time restrictions are imposed on the function itself.";

    container absolute {
        description
            "Absolute time and date that
            the associated function starts
            going into effect.";

        leaf start {
            type yang:date-and-time;
            description
                "Start time and date";
        }
        leaf end {
            type yang:date-and-time;
            description "Absolute end time and date";
        }
        leaf active {
            type boolean;
            default "true";
            description
```



```
        "Specify the associated function
        active or inactive state when
        starts going into effect";
    }
} // container absolute
} //grouping timerange

grouping metadata {
    description "Fields associated with a packet but not in
    the header";

    leaf input-interface {
        type string;
        description "Packet was received on this interface";
    }
    uses timerange;
}
}
```

4.3. A company proprietary module example

In the figure below is an example how proprietary models can be created on top of base ACL module. It is a simple example of how to use 'augment' with an XPath expression which extends instances of a particular type. In this example, all /ietf-acl:access-list/ietf-acl:access-list-entries/ietf-acl:matches are augmented with a new choice, protocol-payload-choice. The protocol-payload-choice uses a grouping with an enumeration of all supported protocol values. In other example, /ietf-acl:access-list/ietf-acl:access-list-entries/ietf-acl:actions are augmented with new choice of actions. Here is an inclusive list of cases listed within a choice statement.

```
module newco-acl {
    yang-version 1;

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

    prefix newco-acl;

    import ietf-acl {
        prefix "ietf-acl";
    }

    revision 2014-05-21{
        description "creating newo proprietary extensions to ietf-acl model";
    }
}
```



```
    augment "/ietf-acl:access-lists/ietf-acl:access-list
    /ietf-acl:access-list-entries/ietf-acl:access-list-entry/ietf-
acl:matches" {
    description "Newco proprietry simple filter matches";
    choice protocol-payload-choice {
        list protocol-payload {
            key value-keyword;
            ordered-by user;
            description "Match protocol payload";
            uses match-simple-payload-protocol-value;
        }
    }
}

    augment "/ietf-acl:access-lists/ietf-acl:access-list
    /ietf-acl:access-list-entries/ietf-acl:access-list-entry/ietf-
acl:actions" {
    description "Newco proprietary simple filter actions";
    choice action {
        case count {
            description "Count the packet in the named counter";
            leaf count {
                type string;
            }
        }
        case policer {
            description "Name of policer to use to rate-limit traffic";
            leaf policer {
                type string;
            }
        }
        case hiearchical-policer {
            description "Name of hierarchical policer to use to rate-limit
traffic";
            leaf hierarchitac1-policer{
                type string;
            }
        }
    }
}

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

```
}
```

```
grouping match-simple-payload-protocol-value {
```

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

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

[4.4.](#) An ACL Example

Requirement: Deny All traffic from 1.1.1.1 bound for host 2.2.2.2 from leaving.

In order to achieve the requirement, an name access control list is needed. The acl and aces can be described in CLI as the following:

```
access-list ip iacl  
deny tcp host 1.1.1.1 host 2.2.2.2
```

Figure 1

Here is the example acl configuration xml:


```
<rpc message-id="101" xmlns:nc="urn:cisco:params:xml:ns:yang:ietf-acl:1.0">
// replace with IANA namespace when assigned
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <top xmlns="http://example.com/schema/1.2/config">
        <access-lists>
          <access-list>
            <acl-name>sample-ip-acl</acl-name>
            <access-list-entries>
              <access-list-entry>
                <rule-name>telnet-block-rule</rule-name>
                <matches>
                  <destination-ipv4-address>2.2.2.2/32</destination-ipv4-address>
                  <source-ipv4-address>1.1.1.1/32</source-ipv4-address>
                </matches>
                <actions>
                  <deny/>
                </actions>
              </access-list-entry>
            </access-list-entries>
          </access-list>
        </access-lists>
      </top>
    </config>
  </edit-config>
</rpc>
```

Figure 2

[4.5.](#) 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.

With the follow XML snippet:

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

This represents port 21.

5. 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 ietf-route-filter {
  yang-version 1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-route-filter";

  prefix ietf-route-filter;

  import ietf-inet-types {
    prefix "ietf-types";
  }

  import ietf-acl {
    prefix "ietf-acl";
  }
```


organization

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

contact

"WG Web: <http://tools.ietf.org/wg/netmod/>

[WG](mailto:WG@ietf.org) List: netmod@ietf.org

WG Chair: Juergen Schoenwaelder

j.schoenwaelder@jacobs-university.de

WG Chair: Tom Nadeau

tnadeau@lucidvision.com

Editor: Dean Bogdanovic

deanb@juniper.net

Editor: Kiran Agrahara Sreenivasa

kkoushik@brocade.com

Editor: Lisa Huang

yihuan@cisco.com

Editor: Dana Blair

dblair@cisco.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 2014-08-15 {

description "creating Route-Filter extensions to ietf-acl model";

reference " ";

}

augment "/ietf-acl:access-list/ietf-acl:access-list-entries/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 upper
prefix/prefix-length.
";
  choice ipv4-range {
    description "Defines the lower IPv4 prefix/prefix range";
    leaf v4-lower-bound {
      type ietf-types:ipv4-prefix;
      description "Defines the lower IPv4 prefix/prefix length";
    }
    leaf v4-upper-bound {
      type ietf-types: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 ietf-types:ipv6-prefix;
      description "Defines the lower IPv6 prefix/prefix length";
    }
    leaf v6-upper-bound {
      type ietf-types:ipv6-prefix;
      description "Defines the upper IPv6 prefix/prefix length";
    }
  }
}
}
```

6. 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 for it (iptables, ip6tables, arptables, ebtables). Recently, this has changed and a single utility, nftables, has been provided. This utility follows very similarly the same base model as proposed in this draft. The nftables support input and output ACEs and each ACE can be defined with match and action.

7. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [[RFC6241](#)] [[RFC6241](#)]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [[RFC6242](#)] [[RFC6242](#)]. The NETCONF access control model [[RFC6536](#)] [[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.

TBD: List specific Subtrees and data nodes and their sensitivity/vulnerability.

8. IANA Considerations

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

URI: urn:ietf:params:xml:ns:yang:ietf-acl

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-acl namespace: urn:ietf:params:xml:ns:yang:ietf-acl
prefix: ietf-acl reference: RFC XXXX

9. 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 draft separately and then work together, to created a new ACL draft that can be supported by different vendors. The new draft removes vendor specific features, and gives examples to allow vendors to extend in their own proprietary ACL. The earlier draft was superseded with the new one that received more participation from many vendors.

10. Change log [RFC Editor: Please remove]

11. References

11.1. Normative References

- [RFC3688] Mealling, M., "The IETF XML Registry", [BCP 81](#), [RFC 3688](#), January 2004.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", [RFC 6020](#), October 2010.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", [RFC 6241](#), June 2011.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", [RFC 6242](#), June 2011.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", [RFC 6536](#), March 2012.

11.2. Informative References

- [RFC5101] Claise, B., "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information", [RFC 5101](#), January 2008.

Authors' Addresses

Dean Bogdanovic
Juniper Networks

Email: deanb@juniper.net

Kiran Agrahara Sreenivasa
Brocade Communications System

Email: kkoushik@brocade.com

Lisa Huang
Cisco Systems

Email: yihuan@cisco.com

Dana Blair
Cisco Systems

Email: dblair@cisco.com