NETMOD WG
Internet-Draft
Intended status: Informational
Expires: December 18, 2014

D. Bogdanovic
Juniper Networks
K. Sreenivasa
Brocade Communications System
D. Blair
Cisco Systems

June 16, 2014

ACL YANG model
draft-bogdanovic-netmod-acl-model-00

Abstract

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Access Control List (ACL) are 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 tuple of match criteria and metadata matching.

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

o Metadata matching applies 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 can be any sort of thing from counting to 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

ARP: Address Resolution Protocol

CoS: Class of Service

DSCP: Differentiated Services Code Point

ICMP: Internet Control Message Protocol

IGMP: Internet Group Management Protocol

IP: Internet Protocol

IPv4: Internet Protocol version 4

IPv6: Internet Protocol version 6

MAC: Media Access Control

QoS: Quality of Service

TCP: Transmission Control Protocol

ToS: Type of Service

TTL: Time To Live

UDP: User Datagram Protocol

VLAN: Virtual Local Area Network

VRF: Virtual Routing and Forwarding

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 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 an access list is. An access list contains an ordered list of rules called access list entries - ACEs. Actions on the first matching ACE are applied with no processing of subsequent 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 matches apply 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 thing from logging to rate limiting or dropping to simply forwarding. There is a default ACE which applies if a packet does not match any of the other ACEs. As some devices fail closed (reject), some fail open (accept) with no explicit configuration, this model defines daufault action, ACE which applies if a packet does not match any of the other ACEs. This will help with cross platform conformitiy and for that reason this draft specifies deny as default action. There is overall operational state for the ACL and operational state for each ACE, and depending on the action, operational state for each action. Access-lists can also have notifications such as logging, configuration changing, activation state changes. The ACL can be applied to targets within the device which may be interfaces of a networked device, applications or features running in the device, or other objects. When applied to interfaces of the 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 module "packet-headers" into the match container. The packet headers can be extended with different types and fragments.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-list
      +--rw acl-name?              string
      +--rw acl-oper-data
      |  +--rw match-counter?    ietf:counter64
      |  +--rw permit-counter?   ietf:counter64
      |  +--rw deny-counter?     ietf:counter64
      |  +--rw targets*          string
      +--rw access-list-entries* [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 ip-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-address
      |  |  |  |        +--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?                string
      |  +--rw actions
      |  |  +--rw (packet-handling)?
      |  |     +--:(deny)
      |  |     |  +--rw deny?     empty
      |  |     +--:(permit)
      |  |        +--rw permit?   empty
      |  +--rw ace-oper-data
      |     +--rw match-counter?   ietf:counter64
      +--rw default-actions
         +--rw deny?   empty
```

Two other module "newco-acl" is example of company proprietary model, that augments the ietf-acl module with definitions that are specific to match criteria and company proprietary extensions to match and action criteria. The model below is shown just an example and it is expected from vendors to create their own propietary models, based on the example below.

```
    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
          +--:(hiearchical-policer)
             +--rw hierarchitacl-policer?   string
```

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 ietf-acl;

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

    revision 2014-05-20{
        description "creating base model for netmod";
    }

    typedef acl-ref {
        description "This type is used by data models that need to referenced
an acl";
        type leafref {
            path "/ietf-acl:access-list/ietf-acl:acl-name";
        }
    }

    container access-list {
        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 implementeded
                     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 {
            description "name of access-list";
            type string;
        }

        container acl-oper-data {
            description "Overall ACL operational data";
            leaf match-counter {
                description "Total match count for ACL";
                type ietf:counter64;
            }
            leaf permit-counter {
                description "Total permit count for ACL";
                type ietf:counter64;
            }
            leaf deny-counter {
                description "Total deny count";
                type ietf:counter64;
            }
            leaf-list targets {
                description "List of targets where ACL is applied";
                type string;
            }
        }
        list access-list-entries {
            key rule-name;
            ordered-by user;
            leaf rule-name {
                description "Entry name";
                type string;
            }

            container matches {
                description "Define match criteria";
                choice ace-type {
                    case ace-ip {
                        uses packet-fields:acl-ip-header-fields;
                        choice ace-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;
                    case deny {
                        leaf deny {
                            type empty;
                        }
                    }
                    case permit {
                        leaf permit {
                            type empty;
                        }
                    }
                }
            }

            container ace-oper-data {
                description "Per ace operational data";
```

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

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

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";
    }

    revision 2014-04-25 {
        description "Initial version of packet fields used by access-lists";
    }

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

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

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

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

        uses acl-transport-header-fields;

        leaf dscp {
            type inet:dscp;
        }

        leaf ip-protocol {
            type uint8;
        }

    }

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

        leaf destination-ipv4-address {
            type inet:ipv4-prefix;
        }

        leaf source-ipv4-address {
            type inet:ipv4-prefix;
        }
```

```
    }

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

        leaf destination-ipv6-address {
            type inet:ipv6-prefix;
        }

        leaf source-ipv6-address {
            type inet:ipv6-address;
        }

        leaf flow-label {
            type inet:ipv6-flow-label;
        }

    }

    grouping acl-eth-header-fields {
        description "fields in ethernet header";

        leaf destination-mac-address {
            type yang:mac-address;
        }

        leaf destination-mac-address-mask {
            type yang:mac-address;
        }

        leaf source-mac-address {
            type yang:mac-address;
        }

        leaf source-mac-address-mask {
            type yang:mac-address;
        }
    }

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

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

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-list/ietf-acl:access-list-entries/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-list/ietf-acl:access-list-entries/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 hierarchitacl-policer{
              type string;
           }
         }
      }
    }

  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";
         }
       }
     }
   }
}
```

Dratf 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 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-list>
               <acl-name>sample-ip-acl</acl-name>
               <access-list-entries>
             <rule-name>telnet-block-rule</rule-name>
             <matches>
               <destination-ipv4-address>2.2.2.2</destination-ipv4-address>
               <source-ipv4-address>1.1.1.1</source-ipv4-address>
             </matches>
             <action>
               <packet-handling>deny</packet-handling>
             </action>
               </access-list-entries>
             </access-list>
           </top>
         </config>
           </edit-config>
         </rpc>
```

Figure 2

5. 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.

6. 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.

7. 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

8. Acknowledgements

9. Change log [RFC Editor: Please remove]

10. References

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

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

[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.

Authors' Addresses

Dean Bogdanovic
Juniper Networks

Email: deanb@juniper.net

Kiran Agrahara Sreenivasa
Brocade Communications System

Email: kkoushik@brocade.com

Dana Blair
Cisco Systems

Email: dblair@cisco.com