Netmod WG
Internet-Draft
Intended status: Standards Track
Expires: 21 April 2022

O. Gonzalez de Dios
S. Barguil
Telefonica
M. Boucadair
Orange

18 October 2021

Extensions to the Access Control Lists (ACLs) YANG Model
draft-dbb-netmod-acl-00

Abstract

RFC 8519 defines a YANG data model for Access Control Lists (ACLs). This document discusses a set of extensions that fix many of the limitations of the ACL model as initially defined in RFC 8519.

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 https://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 21 April 2022.

Table of Contents

1. Introduction

[RFC8519] defines Acces control lists (ACLs) as a user-ordered set of filtering rules. The model targets the configuration of the filtering behaviour of a device. However, the model structure, as defined in [RFC8519], suffers from a set of limitations. This document describes these limitations and proposes an enhanced ACL structure.

The motivation of such enhanced ACL structure is discussed in detail in Section 3.

When managing ACLs, it is common for network operators to group matching elements in pre-defined sets. The consolidation into matches allows reducing the number of rules, especially in large scale networks. If it is needed, for example, to find a match against 100 IP addresses (or prefixes), 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 vs multiple rules matching.

The enhanced ACL structure 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 Access Control List (ACL) and defined sets is generalized so that it is not device-specific as per [RFC8519]. ACLs and defined sets may be defined at 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.

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, supporting means to easily map to the filtering rules conveyed in messages triggered by hese tools is valuable from a network operation standpoint.

1.1. Terminology

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in [RFC2119].

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

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

* Defined set: Refers to reusable description of one or multiple information elements (e.g., IP address, IP prefix, port number, ICMP type).

2. Approach

This first version of the document does not include on purpose any YANG module. This is because the authors are seeking a work direction from the netmod WG whether the missing features can be accomplished by means of augmentations or whether an ACL-bis document is more appropriate.

Future versions of the document will include a YANG module that will reflect the WG feedback. A network wide module, in adition to the device module, might be required. The decision on whether a single module is sufficient to handle both device and network levels or two separate ones will be based on WG feedback.

3. Problem Statement & Gap Analysis

3.1. Suboptimal Configuration: Lack of Manipulating Lists of Prefixes

IP prefix related data nodes, e.g., "destination-ipv4-network" or "destination-ipv6-network", do not allow manipulating a list of IP prefixes, which may lead to manipulating large files. The same issue is encountered when ACLs have to be in place to mitigate DDoS attacks (e.g., [RFC9132]) when a set of sources are involved in such an attack. The situation is even worse when both a list of sources and destination prefixes are involved.

Figure 1 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 1: Example Illustrating Sub-optimal Use of the ACL Model with a Prefix List.

Such configuration is suboptimal for both: - Network controllers that need to manipulate large files. All or a subset fo this configuration will need to be passed to the undelrying network devices. - Devices may receive such confirguration and thus will need to maintain it locally.

Figure 2 depicts an example of an optimized strcuture:

```
{
  "ietf-access-control-list:acls": {
    "acl": [
      {
        "name": "prefix-list-support",
        "type": "ipv6-acl-type",
        "aces": {
          "ace": [
            {
              "name": "my-test-ace",
              "matches": {
                "ipv6": {
                  "destination-ipv6-network": [
                    "2001:db8:6401:1::/64",
                    "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 2: Example Illustrating Optimal Use of the ACL Model in a Network Context.

3.2. Manageability: Impossibility to Use Aliases or Defined Sets

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

The defined sets are reusable definitions across several ACLs. Each category is modelled 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 protcols is identified by the protocol set, if present. Otherwise, a set apply to any protocol.

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

A candidate structure is shown in #example_sets:

```
   +--rw defined-sets
   |  +--rw prefix-sets
   |  |  +--rw prefix-set* [name mode]
   |  |     +--rw name        string
   |  |     +--rw mode        enumeration
   |  |     +--rw ip-prefix*   inet:ip-prefix
   |  +--rw port-sets
   |  |  +--rw port-set* [name]
   |  |     +--rw name    string
   |  |     +--rw port*   inet:port-number
   |  +--rw protocol-sets
   |  |  +--rw protocol-set* [name]
   |  |     +--rw name             string
   |  |     +--rw protocol-name*   identityref
   |  +--rw icmp-type-sets
   |     +--rw icmp-type-set* [name]
   |        +--rw name     string
   |        +--rw types* [type]
   |           +--rw type              uint8
   |           +--rw code?             uint8
   |           +--rw rest-of-header?   binary
```

Figure 3: Examples of Defined Sets.

3.3. 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 binding an ACL to multiple devices, not only (abstract) interfaces.

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

3.4. Partial or Lack of IPv4/IPv6 Fragment Handling

[RFC8519] does not support fragment handling capability for IPv6 but offers a partial support for IPv4 by means 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. Some examples to illustrate how 'fragment' can be used are provided below.

Figure 4 shows the content of a candidate 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": {
                  "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 4: Example Illustrating Canddiate Filtering of IPv4 Fragmented Packets.

Figure 5 shows an example of the body of a candidate 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": {
                   "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 5: Example Illustrating Canddiate Filtering of IPv6 Fragmented Packets.

3.5. Suboptimal TCP Flags Handling

[RFC8519] allows including flags in the TCP match fields, however that strcuture does not support matching operations as those supported in BGP Flow Spec. Definig this field to be defined as a flag bitmask together with a set of operations is meant to efficiently handle TCP flags filtering rules. Some examples to illustrate the use of such field are discussed below.

Figure 6 shows an example of a candidate 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": {
               "flags-bitmask": {
                 "operator": "not any",
                 "bitmask": 4095
               }
             }
           },
           "actions": {
             "forwarding": "drop"
           }
         }]
       }
     }]
   }
 }
```

Figure 6: Example to Deny TCP Null Attack Messages

3.6. 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 'reejct' (i.e., drop matching traffic and send an ICMP error message to the source). Howover, there are situations where the matching traffic can be accepted, but with a rate-limit policy. Such capability is not currently supported by the ACL model.

Figure 7 shows a candidate 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": {
               "flags-bitmask": {
                 "operator": "match",
                 "bitmask": 2
               }
             }
           },
           "actions": {
             "forwarding": "accept",
             "rate-limit": "20.00"
           }
         }]
       }
     }]
   }
 }
```

Figure 7: Example Rate-Limit Incoming TCP SYNs

3.7. 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 current version of the ACL model does not support filetering of encapsulated traffic.

3.8. Reuse the ACLs Content Across Several Devices

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

This network/device ACLs differentiation introduces several new requirements, e.g.:

* 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 VRF, a logical system, or a virtual node. ACLs can be applied in physical and logical infrastructure.

4. Overall Module Structure (TBC)

To be completed.

5. YANG Module (TBC)

To be completed.

6. Security Considerations (TBC)

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

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

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

* TBC

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

* TBC

7. IANA Considerations

7.1. URI Registration (TBC)

This document requests IANA to register the following URI in the "ns" subregistry within the "IETF XML Registry" [RFC3688]:

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

7.2. YANG Module Name Registration (TBC)

This document requests IANA to register the following YANG module in the "YANG Module Names" subregistry [RFC6020] within the "YANG Parameters" registry.

```
name: xxxx
namespace: urn:ietf:params:xml:ns:yang:ietf-xxx
maintained by IANA: N
prefix: xxxx
reference: RFC XXXX
```

8. Acknowledgements

Many thanks to Jon Shallow and Miguel Cros for the discussion when preparing this draft.

9. Normative References

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

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

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

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

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

[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>.

[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>.

[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>.

[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>.

Authors' Addresses

Oscar Gonzalez de Dios
Telefonica
Distrito T
Madrid

Email: oscar.gonzalezdedios@telefonica.com

Samier Barguil
Telefonica
Distrito T
Madrid

Email: samier.barguilgiraldo.ext@telefonica.com

Mohamed Boucadair
Orange
Rennes

Email: mohamed.boucadair@orange.com