     The Data Model of Network Infrastructure Device Data Plane Security
                                Baseline
                 draft-xia-sacm-nid-dp-security-baseline-03

Abstract

   This document proposes one part of the security baseline YANG for
   network infrastructure device (i.e., router, switch, firewall, etc.):
   data plane security baseline. The companion documents [I-D.ietf-lin-
   sacm-nid-mp-security-baseline], [I- D.ietf-dong-sacm-nid-infra-
   security-baseline] cover other parts of the security baseline YANG
   for network infrastructure device respectively: management plane
   security baseline, infrastructure layer security baseline.

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 April 25, 2019.

Table of Contents

1. Introduction

1.1. Objective

   Network security is an essential part of the overall network
   deployment and operation. Due to the following reasons, network
   infrastructure devices (e.g. switch, router, firewall) are always the
   objective and exploited by the network attackers, which bring damages
   to the victim network:

   o The existence of a lot of unsafe access channels: for the history
      reason, some old and unsafe protocols still run in the network
      devices, like: SNMP v1/v2, Telnet, etc., and are not mandatory to
      be replaced by the according safer protocols (SNMP v3, SSH).
      Attackers easily exploit them for attack (e.g., invalid login,
      message eavesdropping);

o The openness nature of TCP/IP network: despite the benefits of
   network architecture design and connectivity brought by the
   network openness, a lot of threats exist at the same time.
   Spoofing address, security weakness for various protocols, traffic
   flooding, and other kinds of threat are originated from the
   network openness;

o The security challenge by the network complexity: network are
   becoming more complex, with massive nodes, various protocols and
   flexible topology. Without careful design and strict management,
   as well as operation automation, the policy consistency of network
   security management cannot be ensured. It's common that part of
   the network infrastructure is subject to attack;

o The complex functionality of device: the complexity of device
   itself increases the difficulty of carrying out the security
   hardening measurements, as well as the skill requirements to the
   network administrator. As a result, the network administrator may
   not be capable of or willing to realize all the security
   measurements, in addition to implementing the other basic
   functionalities;

o The capacity and capability mismatching between the data plane and
   the control plane: there are a large mismatching of the traffic
   processing capacity and capability between different planes.
   Without effective control, the large volume of traffic from the
   data plane will flooding attack the other planes easily.

Therefore, the importance of ensuring the security of the network
infrastructure devices is out of question. To secure the network
infrastructure devices, one important task is to identify as far as
possible the threats and vulnerabilities in the device itself, such
as: unnecessary services, insecure configurations, abnormal status,
etc., then enforce the corresponding security hardening measurements,
such as: update the patch, modify the security configuration, enhance
the security mechanism, etc.. We call this task the developing and
deploying the security baseline for the network infrastructure, which
provides a solid foundation for the overall network security. This
document aims to describe the security baseline for the network
infrastructure, which is called security baseline in short in this
document.

1.2. Security Baseline

Basically, security baseline can be designed and deployed into
different layers of the devices:

o application layer: refers to the application platform security
   solution and the typical application security mechanisms it
   provided like: identity authentication, access control, permission
   management, encryption and decryption, auditing and tracking,
   privacy protection, to ensure secure application data
   transmission/exchange, secure storage, secure processing, ensuring
   the secure operation of the application system. Specific examples
   may be: web application security, software integrity protection,
   encryption of sensitive data, privacy protection, and lawful
   interception interfaces and secure third-party component;

o network layer: refers to a series of security measures, to protect
   the network resources and network services running on the device
   network platform. Network layer security over network product is
   complicated. Therefore, it is divided into data plane, control
   plane, management plane to consider:

   * data plane: focus on the security hardening configuration and
      status to protect the data plane traffic against eavesdropping,
      tampering, forging and flooding attacking the network;

   * control plane: focus on the control signaling security of the
      network infrastructure device, to protect their normal exchange
      against various attacks (i.e., eavesdropping, tampering,
      forging and flooding attack) and restrict the malicious control
      signaling, for ensuring the correct network topology and
      forwarding behavior;

   * management plane: focus on the management information and
      platform security. More specific, it includes all the security
      configuration and status involved in the network OAM process;

o infrastructure layer: refers to all the security design about the
   device itself and its running OS. As the foundation of the upper
   layer services, the secure infrastructure layer must be assured.
   The specific mechanisms include: OS security, key management,
   cryptography security, certificate management, software integrity.

1.3. Security Baseline Data Model Design

   The security baseline varies according to many factors, like:
   different device types (i.e., router, switch, firewall), the
   supporting security features of device, the specific security
   requirements of network operator. It's impossible to design a
   complete set for it, so this document and the companion ones are
   going to propose the most important and universal points of them.
   More baseline contents can be added in future following the data
   model scheme specified.

[I-D.ietf-birkholz-sacm-yang-content] defines a method of
constructing the YANG data model scheme for the security posture
assessment of the network infrastructure device by brokering of YANG
push telemetry via SACM statements. The basic steps are:

o use YANG push mechanism[I-D.ietf-netconf-yang-push]to collect the
   created streams of notifications (telemetry)
   [I-D.ietf-netconf-subscribed-notifications]providing SACM content
   on SACM data plane, and the filter expressions used in the context
   of YANG subscriptions constitute SACM content that is imperative
   guidance consumed by SACM components on SACM management plane;

o then encapsulate the above YANG push output into a SACM Content
   Element envelope, which is again encapsulated in a SACM statement
   envelope;

o lastly, publish the SACM statement into a SACM domain via xmpp-
   grid publisher.

In this document, we follow the same way as [I-D.ietf-birkholz-sacm-
yang-content] to define the YANG output for network infrastructure
device security baseline posture based on the SACM information model
definition [I-D.ietf-sacm-information-model].

1.4. Summary

The following contents propose part of the security baseline YANG
output for network infrastructure device: data plane security
baseline. The companion documents [I-D.ietf- dong-sacm-nid-cp-
security-baseline], [I-D.ietf-lin-sacm-nid-mp-security-baseline], [I-
D.ietf-xia-sacm-nid-app-infr-layers-security-baseline] cover other
parts of the security baseline YANG output for network infrastructure
device respectively: control plane security baseline, management
plane security baseline, application layer and infrastructure layer
security baseline.

2. Terminology

2.1. Key Words

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
document are to be interpreted as described in [RFC2119].

2.2. Definition of Terms

   This document uses the terms defined in [I-D.draft-ietf-sacm-
   terminology].

3. Tree Diagrams

   A simplified graphical representation of the data model is used in
   this document. The meaning of the symbols in these diagrams is as
   follows:

   o Brackets "[" and "]" enclose list keys.

   o Abbreviations before data node names: "rw" means configuration
      (read-write) and "ro" state data (read-only).

   o Symbols after data node names: "?" means an optional node and "*"
      denotes a "list" and "leaf-list".

   o Parentheses enclose choice and case nodes, and case nodes are also
      marked with a colon (":").

   o Ellipsis ("...") stands for contents of subtrees that are not
      shown.

4. Data Model Structure

   As the network infrastructure device, it makes decision of the
   forwarding path based on the IP/MAC address and sends the packet in
   data plane. The NP or ASIC are the main components for the data
   plane functions.

   This section describes the key data plane security baseline of the
   network infrastructure devices, and defines their specific data
   models.

4.1. Layer 2 protection

   Mac table is the key resource in terms of layer 2 forwarding, also
   easily attacked by learning massive invalid mac address. The mac
   limit function is to protect the mac table by limiting the maximum
   number of learned mac address in appointed interfaces. The mac
   address is not learned and the packet is discarded when the up-limit
   is reached, and the alarm is created possibly.

   If the broadcast traffic is not suppressed in layer 2 network (i.e.,
   Ethernet), a great amount of network bandwidth is consumed by a great
   deal of broadcast traffic. The network performance is degraded, even

interrupting the communication. In such a case, configuring the
broadcast traffic suppression on the device to ensure some bandwidth
can be reserved for unicast traffic forwarding when broadcast traffic
bursts across the network. It's flexible to configure the device to
suppress broadcast, multicast, and unknown unicast traffic on an
interface, a specified interface in a VLAN, a sub-interface, and over
a virtual switch instance (VSI) pseudo wire (PW).

```
module: ietf-layer2-protection
   +--rw mac-limit
   |  +--rw vlan-mac-limits
   |  |  +--rw vlan-mac-limit* [vlan-id]
   |  |     +--rw vlan-id       mac-vlan-id
   |  |     +--rw maximum       uint32
   |  |     +--rw rate?         uint32
   |  |     +--rw action?       mac-limit-forward
   |  |     +--rw alarm?        mac-enable-status
   |  +--rw bd-mac-limits
   |  |  +--rw bd-mac-limit* [bd-id]
   |  |     +--rw bd-id         uint32
   |  |     +--rw maximum       uint32
   |  |     +--rw rate?         uint32
   |  |     +--rw action?       mac-limit-forward
   |  |     +--rw alarm?        mac-enable-status
   |  +--rw vsi-mac-limits
   |  |  +--rw vsi-mac-limit* [vsi-name]
   |  |     +--rw vsi-name      string
   |  |     +--rw maximum       uint32
   |  |     +--rw rate?         uint32
   |  |     +--rw action?       mac-limit-forward
   |  |     +--rw alarm?        mac-enable-status
   |  +--rw pw-mac-limits
   |  |  +--rw pw-mac-limit* [vsi-name pw-name]
   |  |     +--rw vsi-name      string
   |  |     +--rw pw-name       string
   |  |     +--rw maximum       uint32
   |  |     +--rw rate?         uint32
   |  |     +--rw action?       mac-limit-forward
   |  |     +--rw alarm?        mac-enable-status
   |  +--rw if-mac-limits
   |  |  +--rw if-mac-limit* [if-name]
   |  |     +--rw if-name       string
   |  |     +--rw maximum       uint32
   |  |     +--rw rate?         uint32
   |  |     +--rw action?       mac-limit-forward
   |  |     +--rw alarm?        mac-enable-status
   |  +--rw subif-mac-limits
   |     +--rw subif-mac-limit* [if-name]
```

```
        │       +--rw if-name        string
        │       +--rw maximum        uint32
        │       +--rw rate?          uint32
        │       +--rw action?        mac-limit-forward
        │       +--rw alarm?         mac-enable-status
    +--rw traffic-suppress
       +--rw vlan-suppresses
       │  +--rw vlan-suppress* [vlan-id suppress-type direction]
       │     +--rw vlan-id        mac-vlan-id
       │     +--rw suppress-type  suppress-type
       │     +--rw direction      direction-type
       │     +--rw cir?           uint64
       │     +--rw cbs?           uint64
       +--rw bd-suppresses
       │  +--rw bd-suppress* [bd-id suppress-type direction]
       │     +--rw bd-id          uint32
       │     +--rw suppress-type  suppress-type
       │     +--rw direction      direction-type
       │     +--rw cir?           uint64
       │     +--rw cbs?           uint64
       +--rw vsi-suppresses
       │  +--rw vsi-suppress* [vsi-name suppress-type direction]
       │     +--rw vsi-name       string
       │     +--rw suppress-type  suppress-type
       │     +--rw direction      direction-type
       │     +--rw cir?           uint64
       │     +--rw cbs?           uint64
       +--rw pw-suppresses
       │  +--rw pw-suppress* [vsi-name pw-name suppress-type direction]
       │     +--rw vsi-name       string
       │     +--rw pw-name        string
       │     +--rw suppress-type  suppress-type
       │     +--rw direction      direction-type
       │     +--rw cir?           uint64
       │     +--rw cbs?           uint64
       +--rw if-suppresses
       │  +--rw if-suppress* [if-name suppress-type direction]
       │     +--rw if-name        string
       │     +--rw suppress-type  suppress-type
       │     +--rw direction      direction-type
       │     +--rw percent?       uint64
       │     +--rw packets?       uint64
       │     +--rw cir?           uint64
       │     +--rw cbs?           uint64
       +--rw sub-if-suppresses
       │  +--rw sub-if-suppress* [if-name suppress-type direction]
       │     +--rw if-name        string
       │     +--rw suppress-type  suppress-type
```

```
         │      +--rw direction        direction-type
         │      +--rw cir?             uint64
         │      +--rw cbs?             uint64
         +--rw if-storm-controls
            +--rw if-storm-control* [if-name]
               +--rw if-name                            string
               +--rw action?                            storm-ctrl-action-type
               +--rw trap-enable?                       enable-type
               +--rw log-enable?                        enable-type
               +--rw interval?                          uint64
               +--rw if-packet-control-rules
               │  +--rw if-packet-control-rule* [packet-type]
               │     +--rw packet-type        storm-ctrl-type
               │     +--rw rate-type?          storm-ctrl-rate-type
               │     +--rw min-rate            uint64
               │     +--rw max-rate            uint64
               +--ro if-storm-control-infos
                  +--ro ifstorm-control-info* [packet-type]
                     +--ro packet-type            storm-ctrl-type
                     +--ro punish-status?         storm-ctrl-action-type
                     +--ro last-punish-time?      string
```

4.2. ARP

   ARP security is a set of functions to protect the ARP protocol and
   networks against malicious attacks so that the network communication
   keeps stable and important user information is protected, which
   mainly includes:

      ARP anti-spoofing functions: protect devices against spoofing ARP
      attack packets, improving the security and reliability of network
      communication.

      ARP anti-flooding functions: relieve CPU load and prevent the ARP
      table overflow, ensuring normal network operation.

```
module: ietf-arp-sec
   +--rw arp-sec
      +--rw arp-packet-validate
      │  +--rw global-validate-type                 arp-validate-type
      │  +--rw if-validate-rule* [if-name]
      │     +--rw if-name                           string
      │     +--rw validate-type                     arp-validate-type
      +--rw arp-gratuitous-control
      │  +--rw global-send-gratuitous-enable        boolean
      │  +--rw global-receive-gratuitous-enable     boolean
      │  +--rw if-gratuitous-rule* [if-name]
      │     +--rw if-name              string
```

```
        |     +--rw send-gratuitous-enable          boolean
        |     +--rw receive-gratuitous-enable       boolean
        +--rw arp-learning-control
        |  +--rw global-strict-learning-enable      boolean
        |  +--rw if-learning-rule* [if-name]
        |     +--rw if-name                         string
        |     +--rw learning-disable                boolean
        |     +--rw strict-learning-enable          boolean
        +--rw arp-entry-limit
        |  +--rw if-limit-rule* [if-name]
        |  |  +--rw if-name                         string
        |  |  +--rw entry-maximum                   uint32
        |  +--rw if-vlan-limit-rule* [if-name vlan-begin]
        |     +--rw if-name                         string
        |     +--rw vlan-begin                      mac-vlan-id
        |     +--rw vlan-end                        mac-vlan-id
        |     +--rw entry-maximum                   uint32
        +--rw arp-rate-limit
        |  +--rw global-rate-limit                  uint32
        |  +--rw limit-by-source-mac
        |  |  +--rw rate-limit-per-source-mac       uint32
        |  |  +--rw source-mac-rule* [source-mac]
        |  |     +--rw source-mac                   mac-address
        |  |     +--rw rate-limit                   uint32
        |  +--rw limit-by-source-ip
        |  |  +--rw rate-limit-per-source-ip        uint32
        |  |  +--rw source-ip-rule* [source-ip]
        |  |     +--rw source-ip                    inet:ip-address
        |  |     +--rw rate-limit                   uint32
        |  +--rw limit-by-destination-ip
        |  |  +--rw rate-limit-per-destination-ip   uint32
        |  +--rw limit-by-interface
        |  |  +--rw rate-limit-per-interface        uint32
        |  |  +--rw interface-rule* [if-name]
        |  |     +--rw if-name                      string
        |  |     +--rw rate-limit                   uint32
        |  +--rw limit-by-vlan
        |     +--rw vlan-rule* [vlan-id]
        |        +--rw vlan-id                      mac-vlan-id
        |        +--rw rate-limit                   uint32
        +--rw arp-miss-rate-limit
        |  +--rw global-rate-limit                  uint32
        |  +--rw limit-by-source-ip
        |  |  +--rw rate-limit-per-source-ip        uint32
        |  |  +--rw source-ip-rule* [source-ip]
        |  |     +--rw source-ip                    inet:ip-address
        |  |     +--rw rate-limit                   uint32
        |  +--rw limit-by-interface
```

```
        │  │     +--rw rate-limit-per-interface          uint32
        │  │     +--rw interface-rule* [if-name]
        │  │        +--rw if-name                        string
        │  │        +--rw rate-limit                     uint32
        │  +--rw limit-by-vlan
        │     +--rw vlan-rule* [vlan-id]
        │        +--rw vlan-id                           mac-vlan-id
        │        +--rw rate-limit                        uint32
        +--ro sec-dis-arp-chks
           +--ro sec-dis-arp-chk* [slot-id check-type]
              +--ro slot-id                              string
              +--ro check-type                           arp-attack-type
              +--ro total-packets?                       uint64
              +--ro passed-packets?                      uint64
              +--ro dropped-packets?                     uint64
```

4.3. URPF

   Unicast Reverse Path Forwarding (URPF) is a technology used to defend
   against network attacks based on source address spoofing. Generally,
   upon receiving a packet, a router first obtains the destination IP
   address of the packet and then searches the forwarding table for a
   route to the destination address. If the router finds such a route,
   it forwards the packet; otherwise, it discards the packet. A URPF-
   enabled router, however, obtains the source IP address of a received
   packet and searches for a route to the source address. If the router
   fails to find the route, it considers that the source address is a
   forged one and discards the packet. In this manner, URPF can
   effectively protect against malicious attacks that are launched by
   changing the source addresses of packets.

   URPF can be performed in strict or loose mode. The strict mode
   checks both the existence of source address in the route table and
   the interface consistency, while loose mode only checks if the source
   address is in the route table. In some case, the router may have
   only one default route to the router of the ISP. Therefore, matching
   the default route entry needs to be supported.

   URPF can be performed over interface, defined flow and traffic sent
   to local CPU.

```
module: ietf-urpf-sec
   +--rw urpf-sec
      +--rw interface-urpf
      |  +--rw interface-rule* [if-name]
      |     +--rw if-name                        string
      |     +--rw urpf-mode                      urpf-mode-type
      |     +--rw allow-default                  boolean
      +--rw flow-urpf
      augment /policy:policies/policy:policy-entry +
             /policy:classifier-entry +
             /policy:classifier-action-entry-cfg +
             /policy:action-cfg-params:
      |  +--:(urpf)
      |     +--rw urpf-cfg
      |        +--rw urpf-mode                   urpf-mode-type
      |        +--rw allow-default               boolean
      +--rw local-urpf
         +--rw slot-rule* [slot-id]
            +--rw slot-id                        string
            +--rw urpf-mode                      urpf-mode-type
            +--rw allow-default                  boolean
```

4.4. DHCP Snooping

   DHCP, which is widely used on networks, dynamically assigns IP
   addresses to clients and manages configuration information in a
   centralized manner. During DHCP packet forwarding, some attacks may
   occur, such as bogus DHCP server attacks, DHCP exhaustion attacks,
   denial of service (DoS) attacks, and DHCP flooding attacks.

   DHCP snooping is a DHCP security feature that functions in a similar
   way to a firewall between DHCP clients and servers. A DHCP-snooping-
   capable device intercepts DHCP packets and uses information carried
   in the packets to create a DHCP snooping binding table. This table
   records hosts' MAC addresses, IP addresses, IP address lease time,
   VLAN, and interface information. The device uses this table to check
   the validity of received DHCP packets. If a DHCP packet does not
   match any entry in this table, the device discards the packet.

   Besides the binding table, DHCP snooping has other security features
   such as trusted interface, max DHCP user limit and whitelist to
   defend against the bogus DHCP server, DHCP flooding and other fine-
   grained DHCP attacks.

```
module: ietf-dhcp-snooping
   +--rw dhcp-snooping
      +--rw dhcp-snooping-enable
      |  +--rw global-enable                      boolean
```

```
      │    +--rw enable-vlan* [vlan-id]
      │    │  +--rw vlan-id                         uint16
      │    │  +--rw dhcp-snp-enable                 boolean
      │    +--rw enable-vlan-interface* [vlan-id if-name]
      │    │  +--rw vlan-id                         uint16
      │    │  +--rw if-name                         string
      │    │  +--rw dhcp-snp-enable                 boolean
      │    +--rw enable-interface* [if-name]
      │    │  +--rw if-name                         string
      │    │  +--rw dhcp-snp-enable                 boolean
      │    +--rw enable-bd* [bd-id]
      │       +--rw bd-id                           uint32
      │       +--rw dhcp-snp-enable                 boolean
      +--rw dhcp-snooping-trust
      │    +--rw trust-vlan* [vlan-id]
      │    │  +--rw vlan-id                         uint16
      │    │  +--rw dhcp-snp-trust                  boolean
      │    │  +--rw untrust-reply-alarm-enable      boolean
      │    │  +--rw untrust-reply-alarm-threshold   uint32
      │    +--rw trust-vlan-interface* [vlan-id if-name]
      │    │  +--rw vlan-id                         uint16
      │    │  +--rw if-name                         string
      │    │  +--rw dhcp-snp-trust                  boolean
      │    │  +--rw untrust-reply-alarm-enable      boolean
      │    │  +--rw untrust-reply-alarm-threshold   uint32
      │    +--rw trust-interface* [if-name]
      │    │  +--rw if-name                         string
      │    │  +--rw dhcp-snp-trust                  boolean
      │    │  +--rw untrust-reply-alarm-enable      boolean
      │    │  +--rw untrust-reply-alarm-threshold   uint32
      │    +--rw trust-bd* [bd-id]
      │       +--rw bd-id                           uint32
      │       +--rw dhcp-snp-trust                  boolean
      │       +--rw untrust-reply-alarm-enable      boolean
      │       +--rw untrust-reply-alarm-threshold   uint32
      +--rw dhcp-snooping-packet-check
      │    +--rw check-vlan* [vlan-id]
      │    │  +--rw vlan-id                         uint16
      │    │  +--rw check-vlan-rule* [check-type]
      │    │     +--rw check-type                   check-type
      │    │     +--rw check-enable                 boolean
      │    │     +--rw alarm-enable                 boolean
      │    │     +--rw alarm-threshold              uint32
      │    +--rw check-vlan-interface* [vlan-id if-name]
      │    │  +--rw vlan-id                         uint16
      │    │  +--rw if-name                         string
      │    │  +--rw check-vlan-if-rule* [check-type]
      │    │     +--rw check-type                   check-type
```

```
   │  │     +--rw check-enable                       boolean
   │  │     +--rw alarm-enable                       boolean
   │  │     +--rw alarm-threshold                    uint32
   │  +--rw check-interface* [if-name]
   │  │  +--rw if-name                               string
   │  │  +--rw check-if-rule* [check-type]
   │  │     +--rw check-type                         check-type
   │  │     +--rw check-enable                       boolean
   │  │     +--rw alarm-enable                       boolean
   │  │     +--rw alarm-threshold                    uint32
   │  +--rw check-bd* [bd-id]
   │     +--rw bd-id                                 uint32
   │     +--rw check-bd-rule* [check-type]
   │        +--rw check-type                         check-type
   │        +--rw check-enable                       boolean
   │        +--rw alarm-enable                       boolean
   │        +--rw alarm-threshold                    uint32
   +--rw dhcp-snooping-max-user-limit
   │  +--rw user-limit-vlan* [vlan-id]
   │  │  +--rw vlan-id                               uint16
   │  │  +--rw max-user-limit                        uint32
   │  │  +--rw alarm-enable                          boolean
   │  │  +--rw alarm-threshold                       uint32
   │  +--rw user-limit-vlan-interface* [vlan-id if-name]
   │  │  +--rw vlan-id                               uint16
   │  │  +--rw if-name                               string
   │  │  +--rw max-user-limit                        uint32
   │  │  +--rw alarm-enable                          boolean
   │  │  +--rw alarm-threshold                       uint32
   │  +--rw user-limit-interface* [if-name]
   │  │  +--rw if-name                               string
   │  │  +--rw max-user-limit                        uint32
   │  │  +--rw alarm-enable                          boolean
   │  │  +--rw alarm-threshold                       uint32
   │  +--rw user-limit-bd* [bd-id]
   │     +--rw bd-id                                 uint32
   │     +--rw max-user-limit                        uint32
   │     +--rw alarm-enable                          boolean
   │     +--rw alarm-threshold                       uint32
   +--rw dhcp-snooping-rate-limit
   │  +--rw global-check-enable                      boolean
   │  +--rw global-rate-limit                        uint32
   │  +--rw global-alarm-enable                      boolean
   │  +--rw global-alarm-threshold                   uint32
   │  +--rw rate-limit-vlan* [vlan-id]
   │  │  +--rw vlan-id                               uint16
   │  │  +--rw check-enable                          boolean
   │  │  +--rw rate-limit                            uint32
```

```
     |  | +--rw alarm-enable                        boolean
     |  | +--rw alarm-threshold                     uint32
     |  +--rw rate-limit-vlan-interface* [vlan-id if-name]
     |  | +--rw vlan-id                             uint16
     |  | +--rw if-name                             string
     |  | +--rw check-enable                        boolean
     |  | +--rw rate-limit                          uint32
     |  | +--rw alarm-enable                        boolean
     |  | +--rw alarm-threshold                     uint32
     |  +--rw rate-limit-interface* [if-name]
     |  | +--rw if-name                             string
     |  | +--rw check-enable                        boolean
     |  | +--rw rate-limit                          uint32
     |  | +--rw alarm-enable                        boolean
     |  | +--rw alarm-threshold                     uint32
     |  +--rw rate-limit-bd* [bd-id]
     |     +--rw bd-id                              uint32
     |     +--rw check-enable                       boolean
     |     +--rw rate-limit                         uint32
     |     +--rw alarm-enable                       boolean
     |     +--rw alarm-threshold                    uint32
     +--rw dhcp-snooping-static-binding-table
     |  +--rw vlan-static-bind-tbl* [vlan-id ip-address ce-vlan]
     |  | +--rw vlan-id                             uint16
     |  | +--rw ip-address                          inet:ip-address
     |  | +--rw mac-address?                        mac-address
     |  | +--rw if-name?                            string
     |  | +--rw ce-vlan                             uint16
     |  +--rw if-static-bind-tbl* [if-name ip-address pe-vlan ce-vlan]
     |  | +--rw if-name                             string
     |  | +--rw ip-address                          inet:ip-address
     |  | +--rw mac-address?                        mac-address
     |  | +--rw pe-vlan                             uint16
     |  | +--rw ce-vlan                             uint16
     |  +--rw bd-static-bind-tbl* [bd-id ip-address pe-vlan ce-vlan]
     |     +--rw bd-id                              uint32
     |     +--rw ip-address                         inet:ip-address
     |     +--rw mac-address?                       mac-address
     |     +--rw pe-vlan                            uint16
     |     +--rw ce-vlan                            uint16
     +--rw dhcp-snp-white-lists
     |  +--rw dhcp-snp-white-list* [wht-lst-name]
     |     +--rw wht-lst-name                       string
     |     +--rw apply-flag                         boolean
     |     +--rw dhcp-snp-white-rules
     |        +--rw dhcp-snp-white-rule* [rule-id]
     |           +--rw rule-id                      uint16
     |           +--rw src-ip?                      inet:ip-address
```

```
          │         +--rw src-mask?                    inet:ip-address
          │         +--rw dst-ip?                      inet:ip-address
          │         +--rw dst-mask?                    inet:ip-address
          │         +--rw src-port?                    dhcp-snp-port
          │         +--rw dst-port?                    dhcp-snp-port
          +--ro dhcp-snp-dyn-bind-tbls
          │  +--ro dhcp-snp-dyn-bind-tbl* [ip-address outer-vlan inner-vlan vsi-nam
          e vpn-name bridge-domain]
          │        +--ro ip-address                    inet:ip-address
          │        +--ro outer-vlan                    uint16
          │        +--ro inner-vlan                    uint16
          │        +--ro vsi-name                      string
          │        +--ro vpn-name                      string
          │        +--ro bridge-domain                 uint32
          │        +--ro mac-address?                  mac-address
          │        +--ro if-name?                      string
          │        +--ro lease?                        yang:date-and-time
          +--ro dhcp-snp-statistics
             +--ro pkt-cnt-drop-by-global-rate       uint32
             +--ro dhcp-snp-vlan-statistics
             │  +--ro dhcp-snp-vlan-statistic* [vlan-id]
             │     +--ro vlan-id                       uint16
             │     +--ro drop-arp-pkt-cnt?             uint32
             │     +--ro drop-ip-pkt-cnt?              uint32
             │     +--ro pkt-cnt-drop-by-user-bind?    uint32
             │     +--ro pkt-cnt-drop-by-mac-check?    uint32
             │     +--ro pkt-cnt-drop-by-untrust-reply?  uint32
             │     +--ro pkt-cnt-drop-by-rate?         uint32
             +--ro dhcp-snp-vlan-if-statistics
             │  +--ro dhcp-snp-vlan-if-statistic* [vlan-id if-name]
             │     +--ro vlan-id                       uint16
             │     +--ro if-name                       string
             │     +--ro drop-arp-pkt-cnt?             uint32
             │     +--ro drop-ip-pkt-cnt?              uint32
             │     +--ro pkt-cnt-drop-by-user-bind?    uint32
             │     +--ro pkt-cnt-drop-by-mac-check?    uint32
             │     +--ro pkt-cnt-drop-by-untrust-reply?  uint32
             │     +--ro pkt-cnt-drop-by-rate?         uint32
             +--ro dhcp-snp-if-statistics
             │  +--ro dhcp-snp-if-statistic* [if-name]
             │     +--ro if-name                       string
             │     +--ro drop-arp-pkt-cnt?             uint32
             │     +--ro drop-ip-pkt-cnt?              uint32
             │     +--ro pkt-cnt-drop-by-user-bind?    uint32
             │     +--ro pkt-cnt-drop-by-mac-check?    uint32
             │     +--ro pkt-cnt-drop-by-untrust-reply?  uint32
             │     +--ro pkt-cnt-drop-by-rate?         uint32
             +--ro dhcp-snp-bd-statistics
                +--ro dhcp-snp-bd-statistic* [if-name]
```

```
          +--ro bd-id                          uint32
          +--ro drop-arp-pkt-cnt?              uint32
          +--ro drop-ip-pkt-cnt?               uint32
          +--ro pkt-cnt-drop-by-user-bind?     uint32
          +--ro pkt-cnt-drop-by-mac-check?     uint32
          +--ro pkt-cnt-drop-by-untrust-reply? uint32
          +--ro pkt-cnt-drop-by-rate?          uint32
```

4.5. CPU Protection

   For the network device, there are maybe a large number of packets to
   be sent to its CPU, or malicious packets attempt to attack the device
   CPU. If the CPU receives excessive packets, it will be overloaded
   and support the normal services with very poor performance; In
   extreme cases, the system fails.

   More specifically, services are negatively affected when the CPU is
   attacked because of the following reasons:

   o Valid protocol packets are not distinguished from invalid protocol
      packets. The CPU is busy in processing a large number of invalid
      protocol packets. Consequently, the CPU usage rises sharply and
      valid packets cannot be processed properly

   o Packets of some protocols are sent to the CPU through the same
      channel. When excessive packets of a certain type of protocol
      packet block the channel, the transmission of other protocol
      packets is affected

   o The bandwidth of a channel is not set appropriately. When an
      attack occurs, processing of protocol packets on other channels is
      affected

   Accordingly, the following countermeasures can be taken by the
   network device for CPU protection:

   o Collect and classify protocols related to various services running
      on equipment

   o Use ACLs to filter the packets. Valid protocol packets are put
      into the whitelist and a user-defined flow, other packets are put
      into the blacklist

   o Plan the priorities, channel bandwidth, length of packets, and
      alarm function of the preceding three lists

   o Disable services that are not deployed on the equipment, and
      control the total forwarding bandwidth

In this manner, the number of packets sent to the CPU is under
control, and the bandwidth is ensured preferentially for services
with higher priorities. In addition, CPU overload is prevented and
an alarm is generated when an attack occurs.

```
module: ietf-cpu-defend
   +--rw cpu-defend
      +--rw cpu-defend-policies
      │  +--rw cpu-defend-policy* [policy-id]
      │     +--rw policy-id                      uint32
      │     +--rw description?                   string
      │     +--rw white-list-acl-number?         uint32
      │     +--rw black-list-acl-number?         uint32
      │     +--rw user-defined-flows
      │     │  +--rw user-defined-flow* [flow-id]
      │     │     +--rw flow-id                  uint32
      │     │     +--rw acl-number               uint32
      │     +--rw cpu-defend-car-rules
      │        +--rw cpu-defend-car-rule* [rule-type pkt-index flow-id protocol
-type tcp-ip-type]
      │           +--rw rule-type                rule-type
      │           +--rw pkt-index?               uint16
      │           +--rw flow-id?                 uint32
      │           +--rw protocol?                protocol-type
      │           +--rw tcp-ip-type?             tcp-ip-type
      │           +--rw car-attr
      │           │  +--rw cir?                  uint32
      │           │  +--rw cbs?                  uint32
      │           │  +--rw pir?                  uint32
      │           │  +--rw pbs?                  uint32
      │           │  +--rw min-pkt-len?          uint32
      │           │  +--rw pkt-rate?             uint32
      │           │  +--rw weight?               uint16
      │           +--rw priority?                priority-type
      │           +--rw drop-alarm
      │              +--rw enable                boolean
      │              +--rw packets-threshold?    uint32
      │              +--rw interval?             uint16
      │              +--rw speed-threshold?      uint32
      +--rw cpu-defend-policy-bindings
      │  +--rw cpu-defend-policy-binding* [slot-id]
      │     +--rw slot-id                        string
      │     +--rw policy-id                      uint32
      +--ro cpu-defend-cars-cfgs
      │  +--ro cpu-defend-cars-cfg* [slot-id pkt-index]
      │     +--ro slot-id                        string
      │     +--ro pkt-index                      uint16
      │     +--ro cir?                           uint32
      │     +--ro cbs?                           uint32
```

```
      │      +--ro min-pkt?                      uint32
      │      +--ro priority?                     priority-type
      │      +--ro protocol                      protocol-type
      +--ro protocol-stats
      │   +--ro protocol-stat* [slot-id protocol]
      │      +--ro slot-id                       string
      │      +--ro protocol                      protocol-type
      │      +--ro default-act                   action-type
      │      +--ro default-cir                   uint32
      │      +--ro default-cbs                   uint32
      +--ro sec-non-car-stats
      │   +--ro sec-non-car-stat* [slot-id policy-type protocol]
      │      +--ro slot-id                       string
      │      +--ro policy-type                   policy-type
      │      +--ro protocol                      protocol-type
      │      +--ro total-packets?                uint64
      │      +--ro passed-packets?               uint64
      │      +--ro dropped-packets?              uint64
      +--ro sec-car-stats
      │   +--ro sec-car-stat* [slot-id policy-type policy-index]
      │      +--ro slot-id                       string
      │      +--ro policy-type                   policy-type
      │      +--ro policy-index                  uint32
      │      +--ro app-enable?                   boolean
      │      +--ro app-default-act?              action-type
      │      +--ro proto-enable?                 boolean
      │      +--ro passed-packets?               uint64
      │      +--ro dropped-packets?              uint64
      │      +--ro cfg-cir?                      uint32
      │      +--ro cfg-cbs?                      uint32
      │      +--ro actual-cir?                   uint32
      │      +--ro actual-cbs?                   uint32
      │      +--ro priority?                     priority-type
      │      +--ro min-pkt-len?                  uint32
      │      +--ro acl-deny-packets?             uint64
      │      +--ro hist-pps?                     uint64
      │      +--ro hist-pps-time?                yang:date-and-time
      │      +--ro average-drop-rate?            uint64
      │      +--ro drop-begin-time?              yang:date-and-time
      │      +--ro drop-end-time?                yang:date-and-time
      │      +--ro total-dropped-packets?        uint64
      +--ro total-packet-stats
      │   +--ro total-packet-stat* [slot-id]
      │      +--ro slot-id                       string
      │      +--ro total-packets?                uint64
      │      +--ro passed-packets?               uint64
      │      +--ro dropped-packets?              uint64
      +--rw hostcar-policies
```

```
         │   +--rw hostcar-policy* [slot-id host-car-type]
         │      +--rw slot-id                      string
         │      +--rw host-car-type                host-car-type
         │      +--rw cir?                         uint32
         │      +--rw pir?                         uint32
         │      +--rw cbs?                         uint32
         │      +--rw pbs?                         uint32
         │      +--rw automatic-adjustment
         │         +--rw enable?                   boolean
         │         +--rw drop-threshold?           uint32
         │         +--rw interval?                 uint32
         +--ro host-car-stats
         │   +--ro host-car-stat* [slot-id host-car-type stat-type host-car-id http
       -host-car-id vlan-host-car-id]
         │      +--ro slot-id                      string
         │      +--ro host-car-type                host-car-type
         │      +--ro stat-type                    stat-type
         │      +--ro host-car-id                  uint32
         │      +--ro http-host-car-id             uint32
         │      +--ro vlan-host-car-id             uint32
         │      +--ro passed-bytes?                uint64
         │      +--ro dropped-bytes?               uint64
         +--ro host-car-cfgs
            +--ro host-car-cfg* [slot-id]
               +--ro slot-id                      string
               +--ro host-car-type?               host-car-type-enum
               +--ro default-cir?                 uint32
               +--ro default-pir?                 uint32
               +--ro default-cbs?                 uint32
               +--ro default-pbs?                 uint32
               +--ro actual-cir?                  uint32
               +--ro actual-pir?                  uint32
               +--ro actual-cbs?                  uint32
               +--ro actual-pbs?                  uint32
               +--ro droprate-en?                 boolean
               +--ro log-interval?                uint32
               +--ro log-threshold?               uint32
```

4.6. TCP/IP Attack Defense

   Defense against TCP/IP attacks is applied to the router on the edge
   of the network or other routers that are easily to be attacked by
   illegal TCP/IP packets. Defense against TCP/IP attacks can protect
   the CPU of the router against malformed packets, fragmented packets,
   TCP SYN packets, and UDP packets, ensuring that normal services can
   be processed.

```
module: ietf-tcp-ip-attack-defense
    +--rw tcp-ip-attack-defense
       +--rw anti-enable?                         boolean
       +--rw abnormal-enable?                     boolean
       +--rw udp-flood-enable?                    boolean
       +--rw tcp-syn-enable?                      boolean
       +--rw icmp-flood-enable?                   boolean
       +--rw fragment-enable?                     boolean
       +--rw sec-anti-attack-car-cfg
       │  +--rw cir-fragment?                     uint32
       │  +--rw cir-icmp?                         uint32
       │  +--rw cir-tcp?                          uint32
       +--rw sec-anti-attack-stats
          +--ro sec-anti-attack-stat* [attack-type]
             +--ro attack-type                    anti-attack-type
             +--ro total-count?                   uint64
             +--ro drop-count?                    uint64
             +--ro pass-count?                    uint64
```

5. Network Infrastructure Device Security Baseline Yang Module

   <CODE BEGINS> file "ietf-mac-limit@2018-06-04.yang"


```
module ietf-mac-limit {
namespace "urn:ietf:params:xml:ns:yang:ietf-mac-limit";
prefix mac-limit;
organization
  "IETF SACM Working Group";
contact
  "Liang Xia: Frank.xialiang@huawei.com;
  Guangying Zheng: Zhengguangying@huawei.com";
description
  "MAC address limit.";

revision 2018-06-04 {
  description
    "Init revision";
  reference "xxx.";
}

/*
 * Typedefs
 */
typedef mac-limit-forward {
  type enumeration {
    enum "forward" {
```

```
        description
          "Forward.";
      }
      enum "discard" {
        description
          "Discard.";
      }
    }
    description
      "MAC Limit Forward";
  }
  typedef mac-enable-status {
    type enumeration {
      enum "enable" {
        description
          "Enable.";
      }
      enum "disable" {
        description
          "Disable.";
      }
    }
    description
      "MAC Enable Status";
  }
  typedef mac-vlan-id {
    type uint16 {
      range "1..4094";
    }
    description
      "MAC Vlan Id";
  }
  typedef mac-type {
    type enumeration {
      enum "static" {
        description
          "Static MAC address entry.";
      }
      enum "dynamic" {
        description
          "Dynamic MAC address entry.";
      }
      enum "black-hole" {
        description
          "Blackhole MAC address entry";
      }
      enum "sticky" {
        description
```

```
            "sticky MAC address entry";
      }
      enum "security" {
        description
          "security MAC address entry";
      }
      enum "evn" {
        description
          "EVN MAC address entry.";
      }
      enum "mux" {
        description
          "MUX MAC address entry.";
      }
      enum "snooping" {
        description
          "SNOOPING MAC address entry.";
      }
      enum "tunnel" {
        description
          "TUNNEL MAC address entry.";
      }
      enum "authen" {
        description
          "AUTHEN MAC address entry.";
      }
    }
    description
      "MAC Type";
  }
  typedef suppress-type {
    type enumeration {
      enum "broadcast" {
        description
          "Broadcast.";
      }
      enum "multicast" {
        description
          "Multicast.";
      }
      enum "unknown-unicast" {
        description
          "Unknown unicast.";
      }
      enum "unicast" {
        description
          "Unicast.";
      }
```

```
      }
    description
      "Suppress Type";
  }
  typedef limit-type {
    type enumeration {
      enum "-mac-limit" {
        description
          "Interface MAC rule limit.";
      }
      enum "mac-apply" {
        description
          "Interface MAC rule application.";
      }
    }
    description
      "Limit Type";
  }

  typedef mac-pw-encap-type {
    type enumeration {
      enum "ethernet" {
        description
          "Ethernet.";
      }
      enum "vlan" {
        description
          "VLAN.";
      }
    }
    description
      "MAC PW Encapsulation Type";
  }

  typedef suppress-style {
    type enumeration {
      enum "percent" {
        description
          "Percent.";
      }
      enum "absolute-value" {
        description
          "Absolute value.";
      }
    }
    description
      "Suppress Style";
  }
```

```
   typedef direction-type {
     type enumeration {
       enum "inbound" {
         description
           "Inbound.";
       }
       enum "outbound" {
         description
           "Outbound.";
       }
     }
     description
       "Direction Type";
   }

   typedef storm-ctrl-action-type {
     type enumeration {
       enum "normal" {
         description
           "Normal.";
       }
       enum "error-down" {
         description
           "Error down.";
       }
       enum "block" {
         description
           "Block.";
       }
       enum "suppress" {
         description
           "Suppress";
       }
     }
     description
       "Storm Ctrl Action Type";
   }

   typedef enable-type {
     type enumeration {
       enum "disable" {
         description
           "Disable.";
       }
       enum "enable" {
         description
           "Enable.";
       }
```

```
      }
    description
      "Enable Type";
  }

  typedef storm-ctrl-type {
    type enumeration {
      enum "broadcast" {
        description
          "Broadcast.";
      }
      enum "multicast" {
        description
          "Multicast.";
      }
      enum "unicast" {
        description
          "Unicast.";
      }
      enum "unknown-unicast" {
        description
          "Unknown unicast.";
      }
    }
    description
      "Storm Ctrl Type";
  }

  typedef storm-ctrl-rate-type {
    type enumeration {
      enum "pps" {
        description
          "Packets per second.";
      }
      enum "percent" {
        description
          "Percent.";
      }
      enum "kbps" {
        description
          "Kilo bits per second.";
      }
    }
    description
      "Storm Ctrl Rate Type";
  }
```

```
   container mac {
     description
       "MAC address forwarding. ";
     container mac-limit-rules {
       description
         "Global MAC address learning limit rule.";
       list mac-limit-rule {
         key "rule-name";
         description
           "Global MAC address learning limit.";
         leaf rule-name {
           type string {
             length "1..31";
           }
           description
             "Global MAC address learning limit rule name.";
         }
         leaf maximum {
           type uint32 {
             range "0..131072";
           }
           mandatory true;
           description
             "Maximum number of MAC addresses that can be learned.";
         }
         leaf rate {
           type uint16 {
             range "0..1000";
           }
           default "0";
           description
             "Interval at which MAC addresses are learned.";
         }
         leaf action {
           type mac-limit-forward;
           default "discard";
           description
             "Discard or forward after the number of learned MAC addresses reache
s the maximum number.";
         }
         leaf alarm {
           type mac-enable-status;
           default "enable";
           description
             "Whether an alarm is generated after the number of learned MAC addre
sses reaches the maximum number.";
         }
       }
     }
     container vlan-mac-limits {
```

```
      description
        "VLAN MAC address limit list.";
      list vlan-mac-limit {
        key "vlan-id";
        description
          "VLAN MAC address limit.";
        leaf vlan-id {
          type mac-vlan-id;
          description
            "VLAN ID.";
        }
        leaf maximum {
          type uint32 {
            range "0..130048";
          }
          mandatory true;
          description
            "Maximum number of MAC addresses that can be learned in a VLAN.";
        }
        leaf rate {
          type uint16 {
            range "0..1000";
          }
          default "0";
          description
            "Interval at which MAC addresses are learned in a VLAN.";
        }
        leaf action {
          type mac-limit-forward;
          default "discard";
          description
            "Discard or forward after the number of learned MAC addresses reache
s the maximum number in a VLAN.";
        }
        leaf alarm {
          type mac-enable-status;
          default "enable";
          description
            "Whether an alarm is generated after the number of learned MAC addre
sses reaches the maximum number in a VLAN.";
        }
      }
    }
    container vsi-mac-limits {
      description
        "VSI MAC address limit list.";
      list vsi-mac-limit {
        key "vsi-name";
        description
          "VSI MAC address limit.";
```

```
        leaf vsi-name {
          type string {
            length "1..31";
          }
          description
            "VSI name.";
        }
        leaf maximum {
          type uint32 {
            range "0..524288";
          }
          mandatory true;
          description
            "Maximum number of MAC addresses that can be learned in a VSI.";
        }
        leaf rate {
          type uint16 {
            range "0..1000";
          }
          default "0";
          description
            "Interval at which MAC addresses are learned in a VSI.";
        }
        leaf action {
          type mac-limit-forward;
          default "discard";
          description
            "Discard or forward after the number of learned MAC addresses reache
s the maximum number in a VSI.";
        }
        leaf alarm {
          type mac-enable-status;
          default "disable";
          description
            "Whether an alarm is generated after the number of learned MAC addre
sses reaches the maximum number in a VSI.";
        }
        leaf up-threshold {
          type uint8 {
            range "80..100";
          }
          mandatory true;
          description
            "Upper limit for the number of MAC addresses.";
        }
        leaf down-threshold {
          type uint8 {
            range "60..100";
          }
          mandatory true;
```

```
          description
            "Upper limit for the number of MAC addresses.";
        }
      }
    }
    container bd-mac-limits {
      description
        "BD MAC address limit list.";
      list bd-mac-limit {
        key "bd-id";
        description
          "BD MAC address limit.";
        leaf bd-id {
          type uint32 {
            range "1..16777215";
          }
          description
            "Specifies the ID of a bridge domain.";
        }
        leaf maximum {
          type uint32 {
            range "0..130048";
          }
          mandatory true;
          description
            "Maximum number of MAC addresses that can be learned in a BD.";
        }
        leaf rate {
          type uint16 {
            range "0..1000";
          }
          default "0";
          description
            "Interval at which MAC addresses are learned in a BD.";
        }
        leaf action {
          type mac-limit-forward;
          default "discard";

          description
            "Forward or discard the packet.";
        }
        leaf alarm {
          type mac-enable-status;
          default "enable";
          description
            "Whether an alarm is generated after the number of learned MAC addre
sses reaches the maximum number.";
        }
```

```
      }
    }
    container pw-mac-limits {
      description
        "PW MAC address limit list.";
      list pw-mac-limit {
        key "vsi-name pw-name";
        description
          "PW MAC address limit.";
        leaf vsi-name {
          type string {
            length "1..31";
          }
          description
            "VSI name.";
        }
        leaf pw-name {
          type string {
            length "1..15";
          }
          description
            "PW name.";
        }
        leaf maximum {
          type uint32 {
            range "0..130048";
          }
          mandatory true;
          description
            "Maximum number of MAC addresses that can be learned in a PW.";
        }
        leaf rate {
          type uint16 {
            range "0..1000";
          }
          default "0";
          description
            "Interval at which MAC addresses are learned in a PW.";
        }
        leaf action {
          type mac-limit-forward;
          default "discard";
          description
            "Discard or forward after the number of learned MAC addresses reache
s the maximum number in a PW.";
        }
        leaf alarm {
          type mac-enable-status;
          default "enable";
```

```
          description
            "Whether an alarm is generated after the number of learned MAC addre
sses reaches the maximum number in a PW.";
        }
      }
    }
    container if-mac-limits {
      description
        "Interface MAC address limit list.";
      list if-mac-limit {
        key "if-name limit-type";
        description
          "Interface MAC address limit.";
        leaf if-name {
          type string;
          description
            "Interface name.";
        }
        leaf limit-type {
          type limit-type;
          description
            "Interface MAC limit type.";
        }
        leaf rule-name {
          type leafref {
            path "/mac/mac-limit-rules/mac-limit-rule/rule-name";
          }
          description
            "Rule name.";
        }
        leaf maximum {
          type uint32 {
            range "0..131072";
          }
          mandatory true;
          description
            "Maximum number of MAC addresses that can be learned on an interface
.";
        }
        leaf rate {
          type uint16 {
            range "0..1000";
          }
          default "0";
          description
            "Interval (ms) at which MAC addresses are learned on an interface.";
        }
        leaf action {
          type mac-limit-forward;
          default "discard";
```

```
            description
              "Discard or forward after the number of learned MAC addresses reache
s the maximum number on an interface";
          }
          leaf alarm {
            type mac-enable-status;
            default "enable";
            description
              "Whether an alarm is generated after the number of learned MAC addre
sses reaches the maximum number on an interface.";
          }
        }
      }
    }
    container if-vlan-mac-limits {
      description
        "Interface + VLAN MAC address limit list.";
      list if-vlan-mac-limit {
        key "if-name vlan-begin limit-type";
        config false;
        description
          "Interface + VLAN MAC address limit.";
        leaf if-name {
          type string;
          description
            "-name of an interface. ";
        }
        leaf vlan-begin {
          type mac-vlan-id;
          description
            "Start VLAN ID.";
        }
        leaf vlan-end {
          type mac-vlan-id;
          description
            "End VLAN ID.";
        }
        leaf limit-type {
          type limit-type;
          description
            "Interface MAC limit type.";
        }
        leaf rule-name {
          type leafref {
            path "/mac/mac-limit-rules/mac-limit-rule/rule-name";
          }
          description
            "Rule name.";
        }
        leaf maximum {
          type uint32 {
```

```
            range "0..131072";
          }
          mandatory true;
          description
            "Maximum number of MAC addresses that can be learned on an interface
.";
        }
        leaf rate {
          type uint16 {
            range "0..1000";
          }
          mandatory true;
          description
            "Interval (ms) at which MAC addresses are learned on an interface.";
        }
        leaf action {
          type mac-limit-forward;
          default "discard";
          description
            "Discard or forward the packet.";
        }
        leaf alarm {
          type mac-enable-status;
          default "enable";
          description
            "Whether an alarm is generated after the number of learned MAC addre
sses reaches the maximum number.";
        }
      }
    }
    container subif-mac-limits {
      description
        "Sub-interface MAC address limit list.";
      list subif-mac-limit {
        key "if-name limit-type";
        description
          "Sub-interface MAC address limit.";
        leaf if-name {
          type string;
          description
            "-name of a sub-interface. ";
        }
        leaf limit-type {
          type limit-type;
          description
            "Sub-interface MAC limit type.";
        }
        leaf vsi-name {
          type string {
            length "1..36";
```

```
            }
            config false;
            mandatory true;
            description
              "VSI name , EVPN name or bridge domain ID.";
          }
          leaf rule-name {
            type string {
              length "1..31";
            }
            mandatory true;
            description
              "Rule name.";
          }
          leaf maximum {
            type uint32 {
              range "0..131072";
            }
            mandatory true;
            description
              "Maximum number of MAC addresses that can be learned on a sub-interf
ace.";
          }
          leaf rate {
            type uint16 {
              range "0..1000";
            }
            default "0";
            description
              "Interval (ms) at which MAC addresses are learned on a sub-interface
.";
          }
          leaf action {
            type mac-limit-forward;
            default "discard";
            description
              "Discard or forward after the number of learned MAC addresses reache
s the maximum number on a sub-interface.";
          }
          leaf alarm {
            type mac-enable-status;
            default "enable";
            description
              "Whether an alarm is generated after the number of learned MAC addre
sses reaches the maximum number on a sub-interface.";
          }
        }
      }
    }
    container vsi-storm-supps {
      description
        "VSI Suppression List.";
      list vsi-storm-supp {
```

```
        key "vsi-name suppress-type";
        description
          "VSI Suppression.";
        leaf vsi-name {
          type string {
            length "1..31";
          }
          description
            "VSI name.";
        }
        leaf suppress-type {
          type suppress-type;
          description
            "Traffic suppression type.";
        }
        leaf cir {
          type uint64 {
            range "0..4294967295";
          }
          default "0";
          description
            "CIR value.";
        }
        leaf cbs {
          type uint64 {
            range "0..4294967295";
          }
          description
            "CBS value.";
        }
      }
    }
    container vlan-storm-supps {
      description
        "VLAN Suppression List.";
      list vlan-storm-supp {
        key "vlan-id suppress-type";
        description
          "VLAN Suppression.";
        leaf vlan-id {
          type mac-vlan-id;
          description
            "VLAN ID.";
        }
        leaf suppress-type {
          type suppress-type;
          description
            "Traffic suppression type.";
```

```
        }
        leaf cir {
          type uint64 {
            range "64..4294967295";
          }
          default "64";
          description
            "CIR value.";
        }
        leaf cbs {
          type uint64 {
            range "10000..4294967295";
          }
          description
            "CBS value.";
        }
      }
    }
  }
  container sub-if-suppresss {
    description
      "Sub-interface traffic suppression list.";
    list sub-if-suppress {
      key "if-name suppress-type direction";
      description
        "Sub-Interface traffic suppression.";
      leaf if-name {
        type string;
        description
          "Sub-interface name.";
      }
      leaf suppress-type {
        type suppress-type;
        description
          "Suppression type.";
      }
      leaf direction {
        type direction-type;
        description
          "Suppression direction.";
      }
      leaf cir {
        type uint64 {
          range "0..4294967295";
        }
        default "0";
        description
          "CIR value.";
      }
```

```
          leaf cbs {
            type uint64 {
              range "0..4294967295";
            }
            description
              "CBS value.";
          }
        }
      }
    }
    container pw-suppresss {
      description
        "PW traffic suppress list.";
      list pw-suppress {
        key "vsi-name pw-name suppress-type";
        description
          "PW traffic suppression.";
        leaf vsi-name {
          type string {
            length "1..31";
          }
          description
            "VSI name.";
        }
        leaf pw-name {
          type string {
            length "1..15";
          }
          description
            "PW name.";
        }
        leaf suppress-type {
          type suppress-type;
          description
            "Traffic suppression type.";
        }
        leaf cir {
          type uint64 {
            range "100..4294967295";
          }
          default "100";
          description
            "CIR value.";
        }
        leaf cbs {
          type uint64 {
            range "100..4294967295";
          }
          description
```

```
            "CBS value.";
        }
      }
    }

    container vsi-in-suppressions {
      description
        "VSI inbound traffic suppression list.";
      list vsi-in-suppression {
        key "vsi-name";
        description
          "VSI inbound traffic suppression.";
        leaf vsi-name {
          type string {
            length "1..31";
          }
          description
            "VSI name.";
        }
        leaf inbound-supp {
          type mac-enable-status;
          default "enable";
          description
            "Inbound suppression.";
        }
      }
    }
    container vsi-out-suppressions {
      description
        "VSI outbound traffic suppression list.";
      list vsi-out-suppression {
        key "vsi-name";
        description
          "VSI outbound traffic suppression.";
        leaf vsi-name {
          type string {
            length "1..31";
          }
          description
            "VSI name.";
        }
        leaf out-bound-supp {
          type mac-enable-status;
          default "enable";
          description
            "Outbound suppression.";
        }
      }
```

```
        }
     container vsi-suppresss {
       description
         "VSI traffic suppression list.";
       list vsi-suppress {
         key "sub-if-name";
         description
           "VSI traffic suppression.";
         leaf vsi-name {
           type string {
             length "1..31";
           }
           mandatory true;
           description
             "VSI name.";
         }

         leaf sub-if-name {
           type string;
           description
             "Sub-interface name.";
         }
         leaf is-enable {
           type boolean;
           default "true";
           description
             "Enable status.";
         }
         leaf suppress-type {
           type suppress-style;
           default "percent";
           description
             "Traffic suppression type.";
         }
         leaf broadcast {
           type uint32 {
             range "0..200000000";
           }
           default "64";
           description
             "Broadcast suppression (kbit/s)";
         }
         leaf broadcast-percent {
           type uint32 {
             range "0..100";
           }
           default "1";
           description
```

```
          "Broadcast suppression.";
      }
      leaf unicast {
        type uint32 {
          range "0..200000000";
        }
        default "64";
        description
          "Unknown unicast suppression (kbit/s).";
      }
      leaf unicast-percent {
        type uint32 {
          range "0..100";
        }
        default "1";
        description
          "Unknown unicast suppression.";

      }
      leaf multicast {
        type uint32 {
          range "0..200000000";
        }
        default "64";
        description
          "Multicast suppression (kbit/s).";
      }
      leaf multicast-percent {
        type uint32 {
          range "0..100";
        }
        default "1";
        description
          "Multicast suppression.";
      }
    }
  }
  container vsi-total-numbers {
    description
      "List of MAC address total numbers in a VSI.";
    list vsi-total-number {
      key "vsi-name slot-id mac-type";
      config false;
      description
        "Total number of MAC addresses in a VSI.";
      leaf vsi-name {
        type string {
          length "1..31";
```

```
          }
          description
            "VSI name.";
        }
        leaf slot-id {
          type string {
            length "1..24";
          }
          description
            "Slot ID.";
        }
        leaf mac-type {
          type mac-type;
          description
            "MAC address type.";
        }
        leaf number {
          type uint32;
          mandatory true;
          description
            "Number of MAC addresses.";
        }
      }
    }
    container if-storm-supps {
      description
        "Interface traffic suppression list.";
      list if-storm-supp {
        key "if-name suppress-type";
        description
          "Interface traffic suppression.";
        leaf if-name {
          type string;
          description
            "-name of an interface. ";
        }
        leaf suppress-type {
          type suppress-type;
          description
            "Suppression type.";
        }
        leaf percent {
          type uint64 {
            range "0..99";
          }
          description
            "Percent.";
        }
```

```
        leaf packets {
          type uint64 {
            range "0..148810000";
          }
          description
            "Packets per second.";
        }
        leaf cir {
          type uint64 {
            range "0..100000000";
          }
          description
            "CIR(Kbit/s).";
        }
        leaf cbs {
          type uint64 {
            range "10000..4294967295";
          }
          description
            "CBS(Bytes).";
        }
      }
    }
    container if-storm-blocks {
      description
        "Interface traffic block list.";
      list if-storm-block {
        key "if-name block-type direction";
        description
          "Interface traffic suppression.";
        leaf if-name {
          type string;
          description
            "-name of an interface. ";
        }
        leaf block-type {
          type suppress-type;
          description
            "Block type.";
        }
        leaf direction {
          type direction-type;
          description
            "Direction.";
        }
      }
    }
    container if-storm-contrls {
```

```
       description
         "Interface storm control list.";
       list if-storm-contrl {
         key "if-name";
         description
           "Interface storm control.";
         leaf if-name {
           type string;
           description
             "-name of an interface. ";
         }
         leaf action {
           type storm-ctrl-action-type;
           default "normal";
           description
             "Action type.";
         }
         leaf trap-enable {

           type enable-type;
           default "disable";
           description
             "Trap state.";
         }
         leaf log-enable {
           type enable-type;
           default "disable";
           description
             "Log state.";
         }
         leaf interval {
           type uint64 {
             range "1..180";
           }
           default "5";
           description
             "Detect interval.";
         }
         container if-packet-contrl-attributes {
           description
             "Storm control rate list.";
           list if-packet-contrl-attribute {
             key "packet-type";
             description
               "Storm control rate.";
             leaf packet-type {
               type storm-ctrl-type;
               description
```

```
                 "Packet type.";
            }
            leaf rate-type {
              type storm-ctrl-rate-type;
              default "pps";
              description
                "Storm control rate type.";
            }
            leaf min-rate {
              type uint32 {
                range "1..148810000";
              }
              mandatory true;
              description
                "Storm control min rate.";
            }
            leaf max-rate {
              type uint64 {
                range "1..148810000";
              }
              mandatory true;
              description
                "Storm control max rate.";
            }
          }
        }
        container ifstorm-contrl-infos {
          description
            "Storm control info list.";
          list ifstorm-contrl-info {
            key "packet-type";
            config false;
            description
              "Storm control info";
            leaf packet-type {
              type storm-ctrl-type;
              description
                "Packet type.";
            }
            leaf punish-status {
              type storm-ctrl-action-type;
              description
                "Storm control status.";
            }
            leaf last-punish-time {
              type string {
                length "1..50";
              }
```

```
                  description
                    "Last punish time.";
              }
            }
          }
        }
      }
    }
}


    <CODE ENDS>
```

6. IANA Considerations

    This document makes no request of IANA.

    Note to RFC Editor: this section may be removed on publication as an
    RFC.

7. Security Considerations

    To be added.

8. Acknowledgements

9. References

9.1. 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>.

9.2. Informative References

    [I-D.ietf-netconf-subscribed-notifications]
               Voit, E., Clemm, A., Prieto, A., Nilsen-Nygaard, E., and
               A. Tripathy, "Customized Subscriptions to a Publisher's
               Event Streams", draft-ietf-netconf-subscribed-
               notifications-17 (work in progress), September 2018.

    [I-D.ietf-netconf-yang-push]
               Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-
               Nygaard, E., Bierman, A., and B. Lengyel, "YANG Datastore
               Subscription", draft-ietf-netconf-yang-push-19 (work in
               progress), September 2018.

   [I-D.ietf-sacm-information-model]
              Waltermire, D., Watson, K., Kahn, C., Lorenzin, L., Cokus,
              M., Haynes, D., and H. Birkholz, "SACM Information Model",
              draft-ietf-sacm-information-model-10 (work in progress),
              April 2017.

Authors' Addresses

   Liang Xia
   Huawei

   Email: frank.xialiang@huawei.com

   Guangying Zheng
   Huawei

   Email: zhengguangying@huawei.com

   Wei Pan
   Huawei

   Email: william.panwei@huawei.com