Network Working Group Y. Dong
Internet-Draft L. Xia
Intended status: Standards Track Huawei
Expires: January 1, 2019 June 30, 2018

Configuration of Advanced Security Functions with I2NSF Security Controller
draft-dong-i2nsf-asf-config-00

Abstract

This draft defines a network security function (NSF-) facing interface of the security controller for the purpose of configuring some advanced security functions. These advanced security functions include antivirus, anti-ddos, and intrusion prevention system (IPS). The interface is presented in a YANG data model fashion and can be used to deploy a large amount of NSF blocks that all support above mentioned functions in the software defined network (SDN) based paradigm.

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 January 1, 2019.

Table of Contents

1. Introduction

I2NSF provides a technology and vendor independent way for a centralized security controller in a SDN environment to manage and configure the distributed NSFs [RFC8329]. The NSFs are automatically customized in a programmable manner via a standard interface [I-D.ietf-i2nsf-sdn-ipsec-flow-protection]. In the draft [I-D.ietf-i2nsf-nsf-facing-interface-dm], it proposed a generic NSF-facing interface to manage which action should be applied on which traffic.

In addition, there is another draft that defined the NSF-facing interface for management, including configuration and monitoring, of IPsec SAs [I-D.ietf-i2nsf-sdn-ipsec-flow-protection]. In this document, we defined another NSF-facing interface for security controller to configure some advanced security functions including the antivirus, anti-ddos, and IPS profiles.

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.ietf-i2nsf-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:

- Brackets "[" and "]" enclose list keys.
- Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).
- Symbols after data node names: "?" means an optional node and "*" denotes a "list" and "leaf-list".
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- Ellipsis ("...") stands for contents of subtrees that are not shown.

4. Data Model Structure

4.1. Antivirus

The following tree diagram shows the interface for configuring antivirus detections on incoming and outgoing files. The file transfer protocol type, direction of file transfer, and the action applied on the detected virus are able to be configured. In addition, this interface also supports to configure the application and signature exception features to apply specific actions on certain applications and detected virus respectively. The anti-virus also supports to configure a whitelist for trusted files.

```
   module: antivirus
      +--rw antivirus
         +--rw antivirus-enable?   boolean
         +--rw profiles
            +--rw profile* [name]
               +--rw name                                string
               +--rw description?                        string
               +--rw collect-attack-evidence-enable?   boolean
               +--rw sandbox-detection-enable?         boolean
               +--rw heuristic-detection-enable?       boolean
               +--rw detect* [protocol-type]
               |  +--rw protocol-type              identityref
               |  +--rw detect-configuration
               |     +--rw direction?         detect-direction
               |     +--rw action?               detect-action
               +--rw exception-application* [application-name]
               |  +--rw application-name                string
               |  +--rw application-action?      detect-action
               +--rw exception-signature-id*             uint64
               +--rw whitelists {antivirus-whitelists}?
                  +--rw match-rules
                  |  +--rw match-rule* [scope type value]
                  |     +--rw scope                match-scope
                  |     +--rw type                  match-type
                  |     +--rw value                     string
                  +--rw source-address*          inet:ip-address
                  +--rw source-range* [start-address end-address]
                  |  +--rw start-address         inet:ip-address
                  |  +--rw end-address           inet:ip-address
                  +--rw destination-address*     inet:ip-address
                  +--rw destin-range* [start-address end-address]
                     +--rw start-address         inet:ip-address
                     +--rw end-address           inet:ip-address
```

4.2. Anti-ddos

The following tree diagram shows the configuration parameters of DDoS detection and prevention functions of different types of DDoS attacks.

* SYN flood: The total number of packets that have the same destination address are counted in a period of time. If the counted packets number exceeds a pre-defined threshold, the anti-ddos function will alert the user/administrator and start up source address inspection.

* UPD flood: The UDP flood packets normally have the same payload or the payload changes regularly. The anti-ddos system is able to

automatically learn this payload characteristics, which is so called fingerprint of the UDP flood attack packets. And then if a packet matches the learned fingerprint, it will be discarded. For some UDP flood attack that does not has a fingerprint, a threshold bandwidth will be configured to limit the UDP traffic.

* HTTP and HTTPS flood: The detection mechanisms for these two attacks are similar to SYN flood detection. The total number of packets that have the same destination address are counted in a period of time. A threshold is set for the purpose of alerting.

* DNS request flood: The anti-ddos system counts the number of DNS request packets that have the same destination address in a period of time. Once this number exceeds a configured threshold, the prevention function is triggered. The anti-ddos utilize the DNS retransmission mechanism. It discards the first request packet, and then waits for the retransmission of another DNS request passively. Or the anti-ddos system sends a response to the client to ask for another request with a TCP connection, and then verify the source address.

* DNS reply flood: The anti-ddos system counts the number of DNS reply packets that have the same destination address in a period of time. Once this number exceeds a configured threshold, the source address inspection is triggered. The anti-ddos ask the sender to send the reply message again with a new query ID and port number. If the second reply message is received and the query ID and port number match with the asked one. This source address will be added into the white list.

* ICMP flood: A threshold is configured to limit the rate of ICMP traffic.

* SIP flood: A threshold is configured to limit the rate of SIP traffic.

```
module: antiddos
   +--rw antiddos-config
      +--rw syn-flood
      |  +--rw alert-rate?       uint32
      +--rw udp-flood
      |  +--rw fingerprint
      |  |  +--rw alert-speed?   uint16
      |  +--rw frag-fingerprint
      |  |  +--rw alert-speed?   uint16
      |  +--rw udp-max-speed
      |     +--rw max-speed?     uint32
      +--rw icmp-flood
      |  +--rw max-speed?        uint32
      +--rw http-flood
      |  +--rw mode?             enumeration
      |  +--rw alert-rate?       uint32
      +--rw https-flood
      |  +--rw alert-rate?       uint32
      +--rw dns-requery-flood
      |  +--rw basic-alert-rate? uint32
      |  +--rw authorized-alert-rate?   uint32
      +--rw dns-reply-flood
      |  +--rw alert-rate?       uint32
      +--rw sip-flood
      |  +--rw alert-rate?       uint32
      +--rw detect-mode?         enumeration
      +--rw baseline-learn
         +--rw auto-apply?       boolean
         +--rw start?            boolean
         +--rw mode?             enumeration
         +--rw tolerance-value?  uint16
         +--rw learn-duration?   uint32
         +--rw learn-interval?   uint32
```

4.3. Intrusion prevention system

The following tree diagram shows the interface for configuring the IPS. This interface supports to configure a set of IPS signature-based filters to detect known type of attacks and to respond with user defined actions such as sending an alert or block the matched packets. The IPS is also able to be configured for preventing DNS and HTTP protocol by checking the DNS request packets and HTTP headers respectively.

```
module: intrusion-prevention-system
   +--rw ips-config
      +--rw ips-enable?            boolean
      +--rw profiles
```

```
            +--rw profile* [name]
               +--rw name                  string
               +--rw description?          string
               +--rw collect-attack-evidence-enable?  boolean
               +--rw associated-check-enable?         boolean
               +--rw domain-filter
               |  +--rw domain-filter-enable?   boolean
               |  +--rw action?                 action-type
               +--rw signature-sets
               |  +--rw signature-set* [name]
               |     +--rw name                       string
               |     +--rw action                   action-type
               |     +--rw application
               |     |  +--rw all-application         boolean
               |     |  +--rw specifed-application*   string
               |     +--rw target?                  target-type
               |     +--rw severity*           severity-type
               |     +--rw operating-system*
               |     |          operating-system-type
               |     +--rw protocol
               |     |  +--rw all-protocol         boolean
               |     |  +--rw specifed-protocol*   string
               |     +--rw category
               |        +--rw all-category         boolean
               |        +--rw specifed-category* [name]
               |           +--rw name                string
               |           +--rw all-sub-category    boolean
               |           +--rw sub-category* [name]
               |              +--rw name    string
               +--rw exception-signatures
               |  +--rw exception-signature* [id]
               |     +--rw id              uint32
               |     +--rw action?         action-type
               |     +--rw timeout?        uint32
               +--rw protocol-controlling
                  +--rw dns-check
                  |  +--rw malformed-packet-check-action    action-type
                  |  +--rw domain-name-check-action        action-type
                  |  +--rw domain-name-length-check
                  |  |  +--rw max-length                    unit16
                  |  |  +--rw action                        action-type
                  |  +--rw request-times-check
                  |  |  +--rw max-time                     unit16
                  |  |  +--rw action                       action-type
                  |  +--rw request-type-check* [start-type end-type]
                  |     +--rw start-type  unit32
                  |     +--rw end-type    unit32
                  |     +--rw action      action-type
```

```
                  +--rw http-check
                     +--rw multi-host-check-action?      action-type
                     +--rw ssh-over-http-check-action?   action-type
                     +--rw x-online-host
                     |  +--rw check-type-action
                     |  |  +--rw type?         http-check-type
                     |  |  +--rw action?           action-type
                     |  +--rw blacklist*               string
                     +--rw x-forwarded-for
                        +--rw check-type-action
                        |  +--rw type?         http-check-type
                        |  +--rw action?           action-type
                        +--rw whitelist*       inet:ip-address
```

5. YANG Modules

```
  module ietf-antivirus {
    yang-version 1.1;

    namespace "urn:ietf:params:xml:ns:yang:ietf-antivirus";
    prefix "antivirus";

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

    organization
      "Huawei Technologies";

    contact
      "Yue Dong: dongyue6@huawei.com";
      "Liang Xia: Frank.xialiang@huawei.com";

    description
      "This module contains a collection of yang definitions
      for configuring antivirus";

    identity detect-protocol {
      description
        "Base type of antivirus detect protocol.";
    }

    identity http {
      base detect-protocol;
      description "Http protocol.";
    }

    identity ftp {
```

```
      base detect-protocol;
      description "Ftp protocol.";
    }

    identity smtp {
      base detect-protocol;
      description "Smtp protocol.";
    }

    identity pop3 {
      base detect-protocol;
      description "Pop3 protocol.";
    }

    identity imap {
      base detect-protocol;
      description "Imap protocol.";
    }

    identity nfs {
      base detect-protocol;
      description "Smb protocol.";
    }

    identity smb {
      base detect-protocol;
      description "Smb protocol.";
    }

    //typedef

    typedef detect-action {
      type enumeration {
        enum alert;
        enum allow;
        enum block;
        enum declare;
        enum delete-attachment;
        enum block-source-ip;
      }
      description
        "This is detect action type in antivirus profile.";
    }

    typedef detect-direction {
      type enumeration {
        enum none;
        enum download;
```

```
        enum upload;
        enum both;
      }
      description
        "This is detect direction type in antivirus profile.";
    }

    typedef match-scope {
      type enumeration {
        enum url;
        enum host;
        enum referer;
      }
      description "This is antivirus whitelist match scope.";
    }

    typedef match-type {
      type enumeration {
        enum prefix;
        enum suffix;
        enum fuzzy;
        enum exact;
      }
      description "This is antivirus whitelist match type.";
    }

    //features

    feature sandbox-detect {
      description
        "This feature means the antivirus function supports
        sandbox detection.";
    }

    feature heuristic-detect {
      description
        "This feature means the antivirus function supports
        heuristic detection.";
    }

    feature antivirus-whitelists {
      description
        "This feature means the antivirus function supports
        whitelists.";
    }

    //groupings
```

```
    grouping address-range {
      leaf start-address {
        type inet:ip-address;
        description
          "Start address.";
      }

      leaf end-address {
        type inet:ip-address;
        description
          "End address.";
      }
    }

    //configuration

    container antivirus {
      leaf antivirus-enable {
        type boolean;
        description
          "Enable/disable the antivirus.";
      }

      container profiles {
        list profile {
          key "name";

          leaf name {
            type string {
              length "1..32";
            }
            description "The name of the profile.";
          }

          leaf description {
            type string {
              length "1..128";
            }
            description "The description of the profile.";
          }

          leaf collect-attack-evidence-enable {
            type boolean;
            description "Enable/disable collect attack evidence.";
          }

          leaf sandbox-detection-enable {
            if-feature sandbox-detect;
```

```
            type boolean;
            description "Enable/disable interworking detect.";
          }

          leaf heuristic-detection-enable {
            if-feature heuristic-detect;
            type boolean;
            description "Enable/disable heuristic detect.";
          }

          list detect {
            key "protocol-type";

            leaf protocol-type {
              type identityref {
                base detect-protocol;
              }
              description "The protocol type of detect.";
            }

            container detect-configuration {
              uses antivirus-detect;
              description "The configuration of antivirus detect.";
            }
          }

          list exception-application {
            key "application-name";

            leaf application-name {
              type string {
                length "1..32";
              }
              description "The exception name of application.";
            }

            leaf application-action {
              type detect-action;
              description "The exception action of application.";
            }
          }

        leaf-list exception-signature-id {
          type uint64;
          description
            "The exception id of antivirus signature, the action
            is allow.";
        }
```

```
          container whitelists {
            description "The whitelist of antivirus.";

            if-feature antivirus-whitelists;
            container match-rules {
              description "The match rules of antivirus whitelist.";

              list match-rule {
                key "scope type value";

                leaf scope {
                  type match-scope;
                  description
                    "The scope of antivirus whitelist match rule.";
                }

                leaf type {
                  type match-type;
                  description
                    "The type of antivirus whitelist match rule.";
                }

                leaf value {
                  type string {
                    length "1..128";
                  }
                  description
                    "The value of antivirus whitelist match rule.";
                }
              }
            }

            leaf-list source-address {
              type inet:ip-address;
              description "The source-address of whitelist.";
            }

            list source-address-range {
              key "start-address end-address";
              uses address-range;
            }

            leaf-list destination-address {
              type inet:ip-address;
              description "The destination-address of whitelist.";
            }

            list destination-address-range {
```

```
            key "start-address end-address";
            uses address-range;
          }
        }
      }
    }
  }
}

module ietf-antiddos {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-antiddos";
  prefix "antiddos";

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

  organization
    "Huawei Technologies";

  contact
    "Yue Dong: dongyue6@huawei.com";
    "Liang Xia: Frank.xialiang@huawei.com";

  description
    "This module contains a collection of yang definitions
    for configuring anti-ddos";

  container antiddos-config {
    container syn-flood {
      leaf alert-rate {
        description "syn-flood source-detect alert-rate";
        type uint32 {
          range "1..80000000";
        }
        units pps;
      }
    }

    container udp-flood {
      container fingerprint {
        description
          "anti-ddos udp-flood dynamic-fingerprint-learn";

        leaf  alert-speed {
          description
```

```
            "udp-flood dynamic-fingerprint-learn alert-speed";
          type uint16 {
            range "1..10240";
          }
          units Mbps;
        }
      }

      container frag-fingerprint {
        description
          "anti-ddos udp-frag-flood dynamic-fingerprint-learn";

        leaf  alert-speed {
          description
            "udp-frag-flood dynamic-fingerprint-learn alert-speed";
          type uint16 {
            range "1..10240";
          }
          units Mbps;
        }
      }

      container udp-max-speed{
        description "anti-ddos udp-max-speed";

        leaf max-speed {
          description "udp max speed";
          type uint32 {
            range "1..10240";
          }
          units Mbps;
        }
      }
    }

    container icmp-flood {
      leaf max-speed {
        description "icmp max speed";
        type uint32 {
          range "1..1200000";
        }
        units pps;
      }
    }

    container http-flood {
      leaf mode {
        description "anti-ddos http flood source detect mode";
```

```
          type enumeration {
            enum advanced {
              description "Indicate advanced source detection";
            }

            enum basic {
              description "Indicate basic source detection";
            }

            enum redirect {
              description "Indicate redirect source detection";
            }
          }
        }

        leaf alert-rate  {
          description "https-flood source-detect alert-rate";
          type uint32 {
            range "1..80000000";
          }
          units pps;
        }
      }

      container https-flood {
        leaf alert-rate  {
          description "https-flood source-detect alert-rate";
          type uint32 {
            range "1..80000000";
          }
          units pps;
        }
      }

      container dns-request-flood {
        leaf basic-alert-rate {
          description
            "dns-requery-flood source-detect basic alert-rate";
          type uint32 {
            range "1..80000000";
          }
          units pps;
        }

        leaf authorized-alert-rate {
          description
            "dns-requery-flood source-detect authorized alert-rate";
          type uint32 {
```

```
            range "1..80000000";
          }
          units pps;
        }
      }

      contianer dns-reply-flood {
        leaf alert-rate {
          description "dns-reply-flood source-detect alert-rate";
          type uint32 {
            range "1..80000000";
          }
          units pps;
        }
      }

      container sip-flood {
        leaf alert-rate {
          description "sip-flood source-detect alert-rate";
          type uint32 {
            range "1..80000000";
          }
          units pps;
        }
      }

      leaf detect-mode {
        description "DDoS detect mode";
        type enumeration {
          enum detect-clean {
            description "Detect DDoS attacks and defend against them";
          }

          enum detect-only{
            description "Detect DDoS attacks only";
          }
        }
      }

      container baseline-learn {
        leaf auto-apply {
          description "Apply baseline learning results";
          type boolean;
        }

        leaf start {
          description "Enable baseline learning";
          type boolean;
```

```
        }

        leaf mode {
          description "Indicate the baseline learning mode";
          type enumeration {
            enum loop {
              description
                "Indicate that baseline learning is performed
periodically";
            }
          }
        }

        leaf tolerance-value {
          description "Indicate the baseline learning tolerance value";
          type uint16 {
            range "0..4000";
          }
        }

        leaf learn-duration {
          description "Indicate the baseline learning duration";
          type uint32 {
            range "1..1200000";
          }
          units minutes;
        }

        leaf learn-interval {
          description "Indicate the interval for baseline learning";
          type uint32 {
            range "1..1200000";
          }
          units minutes;
        }
      }
    }
  }

  module ietf-ips {
    yang-version 1.1;

    namespace "urn:ietf:params:xml:ns:yang:ietf-ips";
    prefix "ips";

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

```
    organization
      "Huawei Technologies";

    contact
      "Yue Dong: dongyue6@huawei.com";
      "Liang Xia: Frank.xialiang@huawei.com";

    description
      "This module contains a collection of yang definitions for
      configuring ips";

    typedef operating-system-type {
      type enumeration {
        enum android;
        enum ios;
        enum unix-like;
        enum windows;
        enum other;
      }
      description "The operating system type.";
    }

    typedef severity-type {
      type enumeration {
        enum high;
        enum medium;
        enum low;
        enum information;
      }
      description "The severity filter type.";
    }

    typedef target-type {
      type enumeration {
        enum both;
        enum client;
        enum server;
      }
      description "The target type.";
    }

    typedef action-type {
      type enumeration {
        enum default-type;
        enum alert;
        enum block;
        enum allow;
        enum block-source-ip;
```

```
        enum block-destination-ip;
      }
      description "The action type.";
    }

    typedef block-ip-type {
      type enumeration {
        enum block-source-ip;
        enum block-destination-ip;
        enum block-both;
      }
      description "The block ip type.";
    }

    typedef condition-operation {
      type enumeration {
        enum pattern-match;
        enum equal;
        enum greater-than;
        enum less-than;
        enum not-equal;
        enum prefix-match;
      }
      description "The operation type of condition.";
    }

    typedef check-scope-type {
      type enumeration {
        enum flow;
        enum message;
        enum packet;
      }
      description "The checked scope type.";
    }

    typedef check-sequence-type {
      type enumeration {
        enum sequential;
        enum random-order;
      }
      description "The checked sequence type.";
    }

    typedef correlation-type {
      type enumeration {
        enum source;
        enum destination;
        enum source-destination;
```

```
      }
      description "The correlation by type.";
    }

    typedef state-type {
      type enumeration {
        enum enable;
        enum disable;
        enum retired;
      }
      description "The status type.";
    }

    typedef http-check-type {
      type enumeration {
        enum any;
        enum multiple;
        enum blacklist;
        enum whitelist;
      }
      description "The http check type.";
    }

    typedef direction-type {
      type enumeration {
        enum both;
        enum from-client;
        enum from-server;
      }
      description "The direction type of packets.";
    }

    container ips-config {
      leaf ips-enable {
        type boolean;
        description
          "Enable/disable the IPS function."
      }

      container profiles {
        list profile {
          key "name";

          leaf name {
            type string;
            description "The name of a profile.";
          }
```

```
          leaf description {
            type string;
            description "The description of a profile.";
          }

          leaf collect-attack-evidence-enable {
            type boolean;
            description "Enable/disable attack evidence collection.";
          }

          leaf associated-check-enable {
            type boolean;
            description "Enable/disable associate check.";
          }

          container domain-filter {
            leaf domain-filter-enable {
              type boolean;
              description
                "The commanding and controlling domain filter is enable
                or disable.";
             }

            leaf action {
              when "./domain-filter-enable = 'true' ";
              type action-type;
              description
                "The action type of a commanding and controlling domain
                filter.";
            }
          }

          container signature-sets {
            list signature-set {
              key "name";

              leaf name {
                type string;
                description "The name of a signature set.";
              }

              leaf action {
                type action-type;
                description "The action for a signature set.";
              }

              container application {
                leaf all-application {
```

```
        type boolean;
        mandatory true;
        description
          "The all application filtering conditions of a
          signature set.";
      }

      leaf-list specifed-application {
        when "./all-application = 'false'";
        type string;
        description
          "The specifed application filtering conditions of
          a signature set.";
      }
    }

    leaf-leaf target {
      type target-type;
      description
        "The target type of a signature set.";
    }

    leaf-list severity {
      type severity-type;
      description
        "The severity type of a signature set.";
    }

    leaf-list operating-system {
      type operating-system-type;
      description
        "The operating system of a signature set.";
    }

    container protocol {
      leaf all-protocol {
        type boolean;
        mandatory true;
        description
          "The all protocol filtering conditions of a
          signature set.";
      }

      leaf-list specifed-protocol {
        when "./all-protocol = 'false'";
        type string;
        description
          "The specifed protocol filtering conditions of
```

```
                    a signature set.";
                }
              }

              container category {
                leaf all-category {
                  type boolean;
                  mandatory true;
                  description
                    "The all category filtering conditions of
                    a signature set.";
                }

                list specifed-category {
                  key "name";
                  when "../all-category = 'false'";

                  leaf name {
                    type string;
                    description
                      "The specifed name of category
                      filtering conditions of a signature set.";
                  }

                  leaf all-sub-category {
                    type boolean;
                    mandatory true;
                    description
                      "The all sub-category filtering
                      conditions of a signature set.";
                  }

                  list sub-category {
                    key "name";

                    leaf name {
                      type string;
                      description
                        "The specifed name of sub-category filtering
                        conditions of a signature set.";
                    }
                  }
                }
              }
            }
          }

          container exception-signature {
```

```
            list exception-signature {
              key "id";

              leaf id {
                type uint32;
                description "The ID of an exception signature.";
              }

              leaf action {
                type action-type;
                description
                  "This action type of an exception signature.";
              }

              leaf timeout {
                when "./action = 'block-source-ip' \
                     or ./action = 'block-destination-ip'";
                type uint32;
                units 'minute';
                description
                  "The blocking time of an exception signature.";
              }
            }
          }

          container protocol-controlling {
            containter dns-check {
              leaf malformed-packet-check-action {
                type action-type;
                description
                  "The action taking on the checked malformed
packets.";
              }

              leaf domain-name-check-action {
                type action-type;
                description
                  "The action taking if the domain name has anomaly
                  characters";
              }

              container domain-name-length-check {
                leaf max-length {
                  type unit16;
                  description
                    "Maximum length allowed for domain name";
                }

                leaf action {
```

```
            type action-type;
            description
              "The action taking if the length of the domain name
              exceed the max-length.";
          }
        }

        container request-times-check {
          leaf max-time {
            type unit16;
            description
              "The maximum number of request allowed."
          }

          leaf action {
            type action-type;
            description
              "The action taking if the number of request exceed
              the max-times";
          }
        }

        list request-type-check {
          key "start-type end-type";

          leaf start-type {
            type unit32;
            description
              "The checked start type of dns request-type.";
          }

          leaf end-type {
            type unit32;
            description
              "The checked end type of dns request-type.";
          }

          leaf action {
            type action-type;
            description
              "The action applied.";
          }
        }
      }

      contaienr http-check {
        leaf  multi-host-action {
          type action-type;
```

```
                description
                  "The checked action type of http multi-host.";
              }

              leaf ssh-over-http-action {
                type action-type;
                description
                  "The checked action type of http ssh-over-http.";
              }

              container x-online-host {
                description
                  "The checked information of http x-online-host.";

                container check-type-action {
                  description
                    "The checked type and action of http x-online-
host.";

                  leaf type {
                    type http-check-type;
                    description "The checked type of http x-online-
host.";
                  }

                  leaf action {
                    type action-type;
                    description
                      "The checked action type of http x-online-host.";
                  }
                }

                leaf-list blacklist {
                  type string;
                  description
                    "The checked domain name or IP address of http
                    x-online-host.";
                }
              }

              container x-forwarded-for {
                description
                  "The checked information of http x-forwarded-for.";

                container check-type-action {
                  description
                    "The checked type and action of http x-forwarded-
for.";

                  leaf type {
                    type http-check-type;
```

```
                    description "The checked type of http x-forwarded-
for.";
                  }

                  leaf action {
                    type action-type;
                    description
                      "The checked action type of http x-forwarded-
for.";
                  }
                }

                leaf-list whitelist {
                  type inet:ip-address;
                  description
                    "The checked ipv4 address of http x-forwarded-
for.";
                }
              }
            }
          }
        }
      }
    }
  }
```

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

TBD.

8. Acknowledgements

TBD

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-i2nsf-nsf-facing-interface-dm]
Kim, J., Jeong, J., Jung-Soo, P., Hares, S., and l. linqiushi@huawei.com, "I2NSF Network Security Function-Facing Interface YANG Data Model", draft-ietf-i2nsf-nsf-facing-interface-dm-00 (work in progress), March 2018.

[I-D.ietf-i2nsf-sdn-ipsec-flow-protection]
Lopez, R. and G. Lopez-Millan, "Software-Defined Networking (SDN)-based IPsec Flow Protection", draft-ietf-i2nsf-sdn-ipsec-flow-protection-01 (work in progress), March 2018.

[I-D.ietf-i2nsf-terminology]
Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-05 (work in progress), January 2018.

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

Authors' Addresses

Yue Dong
Huawei

Email: dongyue6@huawei.com

Liang Xia
Huawei

Email: frank.xialiang@huawei.com