

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 18, 2019

W. Pan
L. Xia
Huawei
October 15, 2018

Configuration of Advanced Security Functions with I2NSF Security
Controller
draft-dong-i2nsf-asf-config-01

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 April 18, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

Internet-Draft Config. Advanced Sec. Func. in I2NSF

October 2018

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	2
2.	Terminology	3
2.1.	Key Words	3
2.2.	Definition of Terms	3
3.	Tree Diagrams	3
4.	Data Model Structure	3
4.1.	Antivirus	3
4.2.	Anti-ddos	4
4.3.	Intrusion prevention system	6
5.	YANG Modules	7
5.1.	Antivirus	7
5.2.	Anti-ddos	13
5.3.	Intrusion prevention system	20
6.	IANA Considerations	26
7.	Security Considerations	26
8.	Acknowledgements	26
9.	References	26
9.1.	Normative References	26
9.2.	Informative References	26
	Authors' Addresses	27

[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. 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. With the variety and complexity of the advanced security functions, it is

hardly to define all the interfaces to configure each advanced security function. The antivirus, anti-ddos and IPS profiles, these three functions are the most common and well-developed advanced security functions and have been widely used. Standardizing the interface of these three functions can minimize the cost of

management and configuration of the security controller with a vendor independent way.

[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:

- 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

[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: ietf-i2nsf-asf-config-antivirus
  +--rw antivirus
    +--rw profiles
      +--rw profile* [name]
        +--rw name string
        +--rw description? string
        +--rw detect* [protocol-type direction]
          | +--rw protocol-type detect-protocol
          | +--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* [signature-id]
          | +--rw signature-id uint64
          | +--rw signature-action? detect-action
        +--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-address-range*
            [start-address end-address]
            | +--rw start-address inet:ip-address
            | +--rw end-address inet:ip-address
          +--rw destination-address* inet:ip-address
          +--rw destination-address-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 prevention function is triggered. The anti-ddos system will alert the user/administrator, and start up source address inspection or TCP proxy function as configured.

* 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. If the UDP packet is associated with some TCP packets, the anti-ddos system can trigger the TCP protection measures and use the generated white list to determine whether to discard the UDP packets.

* 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 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: The anti-ddos system counts the number of SIP request packets that have the same destination address in a period of time. If the counted packets number exceeds a pre-defined threshold, the source authentication is triggered. The anti-ddos system sends an OPTIONS request packet with a specific branch value to verify whether the source address exists. If the reply message is in response to the OPTIONS packet, this source address will be added into the white list.

```
module: ietf-i2nsf-asf-config-antiddos
  +--rw antiddos
    +--rw profiles
      +--rw profile* [name]
        +--rw name string
        +--rw description? string
        +--rw syn-flood* [action]
          | +--rw action syn-flood-action
          | +--rw alert-rate? uint32
        +--rw udp-flood* [action]
          | +--rw action udp-flood-action
          | +--rw alert-rate? uint32
        +--rw http-flood* [action]
          | +--rw action http-flood-action
          | +--rw alert-rate? uint32
```

```

+--rw https-flood* [action]
|   +--rw action                https-flood-action
|   +--rw alert-rate?           uint32
+--rw dns-request-flood* [action]
|   +--rw action                dns-request-flood-action
|   +--rw alert-rate?           uint32
+--rw dns-reply-flood* [action]
|   +--rw action                dns-reply-flood-action
|   +--rw alert-rate?           uint32
+--rw icmp-flood * [action]
|   +--rw action                icmp-flood-action
|   +--rw alert-rate?           uint32
+--rw sip-flood* [action]
|   +--rw action                sip-flood-action
|   +--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.

```

module: ietf-i2nsf-asf-config-ips
+--rw ips
+--rw profiles
+--rw profile* [name]
+--rw name                string
+--rw description?        string
+--rw signature-sets
|   +--rw signature-set* [name]

```

```

|      +--rw name                                string
|      +--rw action?                             action-type
|      +--rw application
|      |   +--rw all-application                  boolean
|      |   +--rw specified-application*           string
|      +--rw target?                             target-type
|      +--rw severity*                           severity-type
|      +--rw operating-system*                   operating-system-type
|      +--rw protocol
|      |   +--rw all-protocol                     boolean
|      |   +--rw specified-protocol*              string
|      +--rw category
|      |   +--rw all-category                     boolean
|      |   +--rw specified-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

```

5. YANG Modules

5.1. Antivirus

```

module ietf-i2nsf-asf-config-antivirus {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-i2nsf-asf-config-antivirus";
  prefix
    asf-config-antivirus;

  import ietf-inet-types{
    prefix inet;
  }

  organization
    "Huawei Technologies";

```



```
"Wei Pan: william.panwei@huawei.com
Liang Xia: Frank.xialiang@huawei.com";
```

```
description
```

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

```
revision 2018-10-15 {
  description
    "Init revision.";
  reference "xxx.";
}
```

```
typedef detect-protocol {
  type enumeration {
    enum http {
      description "HTTP.";
    }
    enum ftp {
      description "FTP.";
    }
    enum smtp {
      description "SMTP.";
    }
    enum pop3 {
      description "POP3.";
    }
    enum imap {
      description "IMAP.";
    }
    enum nfs {
      description "NFS.";
    }
    enum smb {
      description "SMB.";
    }
  }
  description
    "This is detect protocol type in antivirus profile.";
}
```

```
typedef detect-direction {
  type enumeration {
    enum none {
      description "None.";
    }
    enum download {
```

```
        description "Download.";
    }
    enum upload {
        description "Upload.";
    }
    enum both {
        description "Both directions.";
    }
}
description
    "This is detect direction type in antivirus profile.";
}

typedef detect-action {
    type enumeration {
        enum alert {
            description "Permit files and generate virus logs.";
        }
        enum allow {
            description "Permit files.";
        }
        enum block {
            description "Block files and generate virus logs.";
        }
        enum declare {
            description
                "Permit virus-infected email messages, then add information to
                announce the detection of viruses and generate virus logs.";
        }
        enum delete-attachment {
            description
                "Permit virus-infected email messages with deleting there
                attachments, add information to announce the detection of
                viruses and generate virus logs.";
        }
    }
    description
        "This is detect action type in antivirus profile.";
}

typedef match-scope {
    type enumeration {
        enum url {
            description "URL.";
        }
        enum host {
```

```
    description "Host.";
}
```

```
    enum referer {
        description "Referer.";
    }
}
description "This is antivirus whitelist match scope.";
}
```

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

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

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

```

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

```

```

container antivirus {
  description "Antivirus.";
}

```

```

container profiles {
  description "Profiles.";
  list profile {
    key "name";
    description "Antivirus profile.";

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

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

    list detect {
      key "protocol-type direction";
      description "Antivirus detect.";

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

      leaf direction {
        type detect-direction;
        description "The direction of detect.";
      }

      leaf action {
        type detect-action;
      }
    }
  }
}

```

```

        description "The action of detect.";
    }
}

list exception-application {
    key "application-name";
    description "Exceptional application.";

    leaf application-name {
        type string;
        description "The name of exceptional application.";
    }

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

```

```

    }
}

list exception-signature {
    key "signature-id";
    description "Exceptional signature.";

    leaf signature-id {
        type uint64;
        description "The exception id of antivirus signature.";
    }

    leaf signature-action {
        type detect-action;
        description "The action of exceptional signature.";
    }
}

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

    container match-rules {
        description "The match rules of antivirus whitelist.";
    }
}

```

```

list match-rule {
  key "scope type value";
  description "The match rule of antivirus whitelist.";

  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;
    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";
  description "The source-address range of whitelist.";
  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";
}

```

```

        description "The destination-address range of whitelist.";
        uses address-range;
    }
}
}
}
}
}
}
}
}
}
}

```

5.2. Anti-ddos

```

module ietf-i2nsf-asf-config-antiddos {
    yang-version 1.1;
    namespace
        "urn:ietf:params:xml:ns:yang:ietf-i2nsf-asf-config-antiddos";
    prefix
        asf-config-antiddos;

    organization
        "Huawei Technologies";

    contact
        "Wei Pan: william.panwei@huawei.com
        Liang Xia: Frank.xialiang@huawei.com";

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

    revision 2018-10-15 {

```

```

        description
            "Init revision.";
        reference "xxx.";
    }

    typedef syn-flood-action {
        type enumeration {
            enum tcp-proxy {
                description
                    "TCP proxy function.";
            }
        }
    }

```

```

    enum tcp-source-authentication {
        description
            "Authenticate the source addresses of TCP packets.";
    }
}
description
    "This is detect action type of syn-flood.";
}

typedef udp-flood-action {
    type enumeration {
        enum fingerprint-learning {
            description
                "Learn the fingerprint of UDP packets.";
        }
        enum udp-tcp-association {
            description
                "Authenticate the source addresses of TCP packets
                associated with UDP packets.";
        }
        enum traffic-limit {
            description
                "Limit the UDP traffic.";
        }
    }
    description
        "This is detect action type of udp-flood.";
}

typedef http-flood-action {
    type enumeration {
        enum source-authentication-meta-refresh {
            description
                "Authenticate the source addresses of HTTP packets by a way of
                meta-refresh.";
        }
        enum source-authentication-code-based {

```

```

        description
            "Authenticate the source addresses of HTTP packets by a way of
            code-based.";
    }

```



```

    enum source-authentication-302-redirect {
        description
            "Authenticate the source addresses of HTTP packets by a way of
            302-redirect.";
    }
}
description
    "This is detect action type of http-flood.";
}

typedef https-flood-action {
    type enumeration {
        enum source-authentication {
            description
                "Authenticate the source addresses of HTTPS packets.";
        }
    }
    description
        "This is detect action type of https-flood.";
}

typedef dns-request-flood-action {
    type enumeration {
        enum source-authentication-dns-cache-server {
            description
                "Authenticate the source addresses of DNS request packets for
                the DNS Cache Server.";
        }
        enum source-authentication-dns-authoritative-server {
            description
                "Authenticate the source addresses of DNS request packets for
                the DNS Authoritative Server.";
        }
    }
    description
        "This is detect action type of dns-request-flood.";
}

typedef dns-reply-flood-action {
    type enumeration {
        enum source-authentication {
            description
                "Authenticate the source addresses of DNS reply packets.";
        }
    }
}

```

```
    }
    description
        "This is detect action type of dns-reply-flood.";
}

typedef icmp-flood-action {
    type enumeration {
        enum traffic-limit {
            description
                "Limit the ICMP traffic.";
        }
    }
    description
        "This is detect action type of icmp-flood.";
}

typedef sip-flood-action {
    type enumeration {
        enum source-authentication {
            description
                "Authenticate the source addresses of SIP packets.";
        }
    }
    description
        "This is detect action type of sip-flood.";
}

container antiddos {
    description "Anti-ddos.";
    container profiles {
        description "Profiles.";
        list profile {
            key "name";
            description "Anti-ddos profile.";

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

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

        list syn-flood {
            key "action";
```

description "SYN flood detect.";

```
    leaf action {
      type syn-flood-action;
      description "The action of syn-flood detect.";
    }

    leaf alert-rate {
      type uint32;
      description "The alert rate of syn-flood detect.";
    }
  }

  list udp-flood {
    key "action";
    description "UDP flood detect.";

    leaf action {
      type udp-flood-action;
      description "The action of udp-flood detect.";
    }

    leaf alert-rate {
      type uint32;
      description "The alert rate of udp-flood detect.";
    }
  }

  list http-flood {
    key "action";
    description "HTTP flood detect.";

    leaf action {
      type http-flood-action;
      description "The action of http-flood detect.";
    }

    leaf alert-rate {
      type uint32;
      description "The alert rate of http-flood detect.";
    }
  }
}
```

```

list https-flood {
  key "action";
  description "HTTPS flood detect.";

  leaf action {
    type https-flood-action;
    description "The action of https-flood detect.";
  }
}

```

```

}

leaf alert-rate {
  type uint32;
  description "The alert rate of https-flood detect.";
}

list dns-request-flood {
  key "action";
  description "DNS request flood detect.";

  leaf action {
    type dns-request-flood-action;
    description "The action of dns-request-flood detect.";
  }

  leaf alert-rate {
    type uint32;
    description "The alert rate of dns-request-flood detect.";
  }
}

list dns-reply-flood {
  key "action";
  description "DNS reply flood detect.";

  leaf action {
    type dns-reply-flood-action;
    description "The action of dns-reply-flood detect.";
  }

  leaf alert-rate {

```

```

        type uint32;
        description "The alert rate of dns-reply-flood detect.";
    }
}

list icmp-flood {
    key "action";
    description "ICMP flood detect.";

    leaf action {
        type icmp-flood-action;
        description "The action of icmp-flood detect.";
    }

    leaf alert-rate {

```

```

        type uint32;
        description "The alert rate of icmp-flood detect.";
    }
}

list sip-flood {
    key "action";
    description "SIP flood detect.";

    leaf action {
        type sip-flood-action;
        description "The action of sip-flood detect.";
    }

    leaf alert-rate {
        type uint32;
        description "The alert rate of sip-flood detect.";
    }
}

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

```

```

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

container baseline-learn {
    description "Alert rate baseline learning.";

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

    leaf start {
        type boolean;
        description "Enable baseline learning.";
    }
}

```

```

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

            enum once {
                description
                    "Indicate that baseline learning is performed once.";
            }
        }
        description "Indicate the baseline learning mode.";
    }

    leaf tolerance-value {
        type uint16;
        description

```



```
    "Init revision.";
    reference "xxx.";
}
```

```
typedef action-type {
    type enumeration {
        enum default-type {
            description "Default action type.";
        }
        enum alert {
            description "Alert.";
        }
        enum block {
            description "Block.";
        }
        enum allow {
            description "Allow.";
        }
    }
    description "The action type.";
}
```

```
typedef target-type {
    type enumeration {
        enum both {
            description "Both client and server.";
        }
        enum client {
            description "Client.";
        }
        enum server {
            description "Server.";
        }
    }
    description "The target type.";
}
```

```
typedef severity-type {
    type enumeration {
        enum high {
            description "High.";
        }
    }
}
```



```

    enum medium {
        description "Medium.";
    }
    enum low {
        description "Low.";
    }
    enum information {
        description "Information.";
    }
}
description "The severity filter type.";
}

typedef operating-system-type {
    type enumeration {
        enum android {
            description "Android OS.";
        }
        enum ios {
            description "IOS.";
        }
        enum unix-like {
            description "UNIX-like OS.";
        }
        enum windows {
            description "Windows OS.";
        }
        enum other {
            description "Other OS.";
        }
    }
    description "The operating system type.";
}

container ips {
    description "Intrusion prevention system.";
    container profiles {
        description "Profiles.";
        list profile {
            key "name";
            description "IPS Profile.";

            leaf name {

```

```

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

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

container signature-sets {
    description "Signature sets.";
    list signature-set {
        key "name";
        description "Signature set.";

        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 {
            description "Application.";
            leaf all-application {
                type boolean;
                mandatory true;
                description
                    "The all application filtering conditions of the
                    signature set.";
            }

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

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

```

Internet-Draft

Config. Advanced Sec. Func. in I2NSF

October 2018

```
}

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

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

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



```

key "id";
description "Exceptional signature.";

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

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

```

```

    }
  }
}
}
}
}
}

```

[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](#),

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.

Pan & Xia

Expires April 18, 2019

[Page 26]

Internet-Draft Config. Advanced Sec. Func. in I2NSF October 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

Wei Pan
Huawei

Email: william.panwei@huawei.com

Liang Xia
Huawei

Email: frank.xialiang@huawei.com