Workgroup: Network Working Group
Internet-Draft:
draft-ietf-i2nsf-nsf-monitoring-data-model-09
Published: 24 August 2021
Intended Status: Standards Track
Expires: 25 February 2022
Authors: J. Jeong, Ed. P. Lingga
Sungkyunkwan University Sungkyunkwan University
S. Hares L. Xia H. Birkholz
Huawei Huawei Fraunhofer SIT

I2NSF NSF Monitoring Interface YANG Data Model

Abstract

This document proposes an information model and the corresponding YANG data model of an interface for monitoring Network Security Functions (NSFs) in the Interface to Network Security Functions (I2NSF) framework. If the monitoring of NSFs is performed with the NSF monitoring interface in a comprehensive way, it is possible to detect the indication of malicious activity, anomalous behavior, the potential sign of denial of service attacks, or system overload in a timely manner. This monitoring functionality is based on the monitoring information that is generated by NSFs. Thus, this document describes not only an information model for the NSF monitoring interface along with a YANG data diagram, but also the corresponding YANG data model.

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 25 February 2022.

Table of Contents

1. Introduction

According to [RFC8329], the interface provided by a Network Security Function (NSF) (e.g., Firewall, IPS, or Anti-DDoS function) to administrative entities (e.g., Security Controller) to enable remote management (i.e., configuring and monitoring) is referred to as an I2NSF Monitoring Interface. This interface enables the sharing of vital data from the NSFs (e.g., alarms, records, and counters) to the Security Controller through a variety of mechanisms (e.g., queries, notifications, and events). The monitoring of NSF plays an important role in an overall security framework, if it is done in a timely and comprehensive way. The monitoring information generated by an NSF can be a good, early indication of anomalous behavior or malicious activity, such as denial of service attacks (DoS).

This document defines a comprehensive information model of an NSF monitoring interface that provides visibility into an NSF for the NSF data collector (e.g., Security Controller). Note that an NSF data collector is defined as an entity to collect NSF monitoring data from an NSF, such as Security Controller. It specifies the information and illustrates the methods that enable an NSF to provide the information required in order to be monitored in a scalable and efficient way via the NSF Monitoring Interface. The information model for the NSF monitoring interface presented in this document is complementary for the security policy provisioning functionality of the NSF-Facing Interface specified in [I-D.ietf-i2nsf-nsf-facing-interface-dm].

This document also defines a YANG [RFC7950] data model for the NSF monitoring interface, which is derived from the information model for the NSF monitoring interface.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document uses the terminology described in [RFC8329].

This document follows the guidelines of [RFC8407], uses the common YANG types defined in [RFC6991], and adopts the Network Management Datastore Architecture (NMDA) [RFC8342]. The meaning of the symbols in tree diagrams is defined in [RFC8340].

3. Use Cases for NSF Monitoring Data

As mentioned earlier, monitoring plays a critical role in an overall security framework. The monitoring of the NSF provides very valuable information to an NSF data collector (e.g., Security Controller) in maintaining the provisioned security posture. Besides this, there are various other reasons to monitor the NSF as listed below:

* The security administrator with I2NSF User can configure a policy that is triggered on a specific event occurring in the NSF or the network [RFC8329] [I-D.ietf-i2nsf-consumer-facing-interface-dm]. If an NSF data collector detects the specified event, it configures additional security functions as defined by policies.

* The events triggered by an NSF as a result of security policy violation can be used by Security Information and Event Management (SIEM) to detect any suspicious activity in a larger correlation context.

* The information (i.e., events, records, and counters) from an NSF can be used to build advanced analytics, such as behavior and predictive models to improve security posture in large deployments.

* The NSF data collector can use events from the NSF for achieving high availability. It can take corrective actions such as restarting a failed NSF and horizontally scaling up the NSF.

* The information (i.e., events, records, and counters) from the NSF can aid in the root cause analysis of an operational issue, so it can improve debugging.

* The records from the NSF can be used to build historical data for operation and business reasons.

4. Classification of NSF Monitoring Data

In order to maintain a strong security posture, it is not only necessary to configure an NSF's security policies but also to continuously monitor the NSF by consuming acquirable and observable data. This enables security administrators to assess the state of the networks and in a timely fashion. It is not possible to block all the internal and external threats based on static security posture. A more practical approach is supported by enabling dynamic security measures, for which continuous visibility is required. This document defines a set of monitoring elements and their scopes that can be acquired from an NSF and can be used as NSF monitoring data. In essence, these types of monitoring data can be leveraged to support constant visibility on multiple levels of granularity and can be consumed by the corresponding functions.

Three basic domains about the monitoring data originating from a system entity [RFC4949], i.e., an NSF, are highlighted in this document.

*Retention and Emission

*Notifications, Events, and Records

*Unsolicited Poll and Solicited Push

As with I2NSF components, every generic system entity can include a set of capabilities that creates information about some context with monitoring data (i.e., monitoring information), composition, configuration, state or behavior of that system entity. This information is intended to be provided to other consumers of information and in the scope of this document, which deals with NSF monitoring data in an automated fashion.

4.1. Retention and Emission

A system entity (e.g., NSF) first retains I2NSF monitoring data inside its own system before emitting the information another I2NSF component (e.g., NSF Data Collector). The I2NSF monitoring information consist of I2NSF Event, I2NSF Record, and I2NSF Counter as follows:

I2NSF Event: I2NSF Event is defined as an important occurrence over time, that is, a change in the system being managed or a change in the environment of the system being managed. An I2NSF Event requires immediate attention and should be notified as soon as possible. When used in the context of an (imperative) I2NSF Policy Rule, an I2NSF Event is used to determine whether the Condition clause of that Policy Rule can be evaluated or not. The Alarm Management Framework in [RFC3877] defines an event as

something that happens which may be of interest. Examples for an event are a fault, a change in status, crossing a threshold, or an external input to the system. In the I2NSF domain, I2NSF events are created following the definition of an event in the Alarm Management Framework.

I2NSF Record: A record is defined as an item of information that is kept to be looked at and used in the future. Unlike I2NSF Event, records do not require immediate attention but may be useful for visibility and retroactive cyber forensic. Depending on the record format, there are different qualities in regard to structure and detail. Records are typically stored in log-files or databases on a system entity or NSF. Records in the form of log-files usually include less structures but potentially more detailed information in regard to the changes of a system entity's characteristics. In contrast, databases often use more strict schemas or data models, therefore enforcing a better structure. However, they inhibit storing information that does not match those models ("closed world assumption"). Records can be continuously processed by a system entity as an I2NSF Producer and emitted with a format tailored to a certain type of record. Typically, records are information generated by a system entity (e.g., NSF) that is based on operational and informational data, that is, various changes in system characteristics. The examples of records include as user activities, network/traffic status, and network activity. They are important for debugging, auditing and security forensic of a system entity or the network having the system entity.

I2NSF Counter: An I2NSF Counter is defined as a specific representation of continuous value changes of information elements that potentially occur in high frequency. Prominent examples are network interface counters for protocol data unit (PDU) amount, byte amount, drop counters, and error counters. Counters are useful in debugging and visibility into operational behavior of a system entity (e.g., NSF). When an NSF data collector asks for the value of a counter to it, a system entity emits

For the utilization of the storage space for accumulated NSF monitoring data, all of the information MUST provide the general information (e.g., timestamp) for purging existing records, which is discussed in Section 5. This document provides a YANG data model in Section 9 for the important I2NSF monitoring information that should be retained. All of the information in the data model is considered important and should be kept permanently as the information might be

useful in many circumstances in the future. The allowed cases for removing some monitoring information include the following:

* When the system storage is full to create a fresh record [RFC4949], the oldest record can be removed.

* The administrator deletes existing records manually after analyzing the information in them.

The I2NSF monitoring information retained on a system entity (e.g., NSF) may be delivered to a corresponding I2NSF User via an NSF data collector. The information consists of the aggregated records, typically in the form of log-files or databases. For the NSF Monitoring Interface to deliver the information to the NSF data collector, the NSF needs to accommodate standardized delivery protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. The NSF data collector can forward the information to the I2NSF User through one of standardized delivery protocols. The interface for this delivery is out of the scope of this document.

4.2. Notifications, Events, and Records

A specific task of I2NSF User is to process I2NSF Policy Rules. The rules of a policy are composed of three clauses: Event, Condition, and Action clauses. In consequence, an I2NSF Event is specified to trigger an I2NSF Policy Rule. Such an I2NSF Event is defined as any important occurrence over time in the system being managed, and/or in the environment of the system being managed, which aligns well with the generic definition of Event from [RFC3877].

Another role of the I2NSF Event is to trigger a notification for monitoring the status of an NSF. A notification is defined in [RFC3877] as an unsolicited transmission of management information. System alarm (called alarm) is defined as a warning related to service degradation in system hardware in Section 6.1. System event (called alert) is defined as a warning about any changes of configuration, any access violation, the information of sessions and traffic flows in Section 6.2. Both an alarm and an alert are I2NSF Events that can be delivered as a notification. The model illustrated in this document introduces a complementary type of information that can be a conveyed notification.

In I2NSF monitoring, a notification is used to deliver either an event and a record via the I2NSF Monitoring Interface. The difference between the event and record is the timing by which the notifications are emitted. An event is emitted as soon as it happens in order to notify an NSF Data Collector of the problem that needs immediate attention. A record is not emitted immediately to the NSF

Data Collector, and it can be emitted periodically to the NSF Data Collector every certain time interval.

It is important to note that an NSF Data Collector as a consumer (i.e., observer) of a notification assesses the importance of the notification rather than an NSF as a producer. The producer can include metadata in a notification that supports the observer in assessing its importance (e.g., severity).

4.3. Unsolicited Poll and Solicited Push

The freshness of the monitored information depends on the acquisition method. Ideally, an I2NSF User is accessing every relevant information about the I2NSF Component and is emitting I2NSF Events to an NSF data collector (e.g., Security Controller) in a timely manner. Publication of events via a pubsub/broker model, peer-2-peer meshes, or static defined channels are only a few examples on how a solicited push of I2NSF Events can be facilitated. The actual mechanism implemented by an I2NSF Component is out of the scope of this document.

Often, the corresponding management interfaces have to be queried in intervals or on demand if required by an I2NSF Policy rule. In some cases, the collection of information has to be conducted via a login mechanism provided by a system entity. Accessing records of information via this kind of unsolicited polls can introduce a significant latency in regard to the freshness of the monitored information. The actual definition of intervals implemented by an I2NSF Component is also out of scope of this document.

5. Basic Information Model for Monitoring Data

As explained in the above section, there is a wealth of data available from the NSF that can be monitored. Firstly, there must be some general information with each monitoring message sent from an NSF that helps a consumer to identify meta data with that message, which are listed as below:

*message: The extra detail to give the context of the information.

*vendor-name: The name of the NSF vendor.

*nsf-name: The name or IP address of the NSF generating the message. If the given nsf-name is not an IP address, the name can be an arbitrary string including FQDN (Fully Qualified Domain Name). The name MUST be unique for different NSFs to identify the NSF that generates the message.

*severity: It indicates the severity level. There are total four levels, i.e., critical, high, middle, and low.

*timestamp: Indicates the time when the message is generated. For the notification operations (i.e., System Alarms, System Events, NSF Events, System Logs, and NSF Logs), this is represented by the eventTime of NETCONF event notification [RFC5277] For other operations (i.e., System Counter and NSF Counter), the timestamp MUST be provided separately.

6. Extended Information Model for Monitoring Data

This section covers the additional information associated with the system messages. The extended information model is only for the structured data such as events, record, and counters. Any unstructured data is specified with the basic information model only.

Each information has characteristics as follows:

*Acquisition method: The method to obtain the message. It can be a "query" or a "subscription". A "query" is a request-based method to acquire the solicited information. A "subscription" is a subscribe-based method to acquire the unsolicited information.

*Emission type: The cause type for the message to be emitted. It can be "on-change" or "periodic". An "on-change" message is emitted when an important event happens in the NSF. A "periodic" message is emitted at a certain time interval. The time to periodically emit the message is configurable.

*Dampening type: The type of message dampening to stop the rapid transmission of messages. The dampening types are "on-repetition" and "no-dampening". The "on-repetition" type limits the transmitted "on-change" message to one message at a certain interval. This interval is defined as dampening-period in [RFC8641]. The dampening-period is configurable. The "no-dampening" type does not limit the transmission for the messages of the same type. In short, "on-repetition" means that the dampening is active and "no-dampening" is inactive. It is recommended to activate the dampening for an "on-change" type of message to reduce the number of messages generated.

6.1. System Alarms

System alarms have the following characteristics:

*acquisition-method: subscription

*emission-type: on-change

*dampening-type: on-repetition

6.1.1. Memory Alarm

The memory is the hardware to store information temporarily or for a short period, i.e., Random Access Memory (RAM). The memory-alarm is emitted when the RAM usage exceeds the threshold. The following information should be included in a Memory Alarm:

*event-name: memory-alarm.

*usage: specifies the size of memory used.

*threshold: The threshold triggering the alarm

*severity: The severity of the alarm such as critical, high, medium, and low.

*message: Simple information such as "The memory usage exceeded the threshold" or with extra information.

6.1.2. CPU Alarm

CPU is the Central Processing Unit that executes basic operations of the system. The cpu-alarm is emitted when the CPU usage exceeds the threshold. The following information should be included in a CPU Alarm:

*event-name: cpu-alarm.

*usage: Specifies the size of CPU used.

*threshold: The threshold triggering the event.

*severity: The severity of the alarm such as critical, high, medium, and low.

*message: Simple information such as "The CPU usage exceeded the threshold" or with extra information.

6.1.3. Disk Alarm

Disk is the hardware to store information for a long period, i.e., Hard Disk or Solid-State Drive. The disk-alarm is emitted when the Disk usage exceeds the threshold. The following information should be included in a Disk Alarm:

*event-name: disk-alarm.

*usage: Specifies the size of disk space used.

*threshold: The threshold triggering the event.

*severity: The severity of the alarm such as critical, high, medium, and low.

*message: Simple information such as "The disk usage exceeded the threshold" or with extra information.

6.1.4. Hardware Alarm

The hardware-alarm is emitted when a hardware, e.g., CPU, memory, disk, or interface, problem is detected. The following information should be included in a Hardware Alarm:

*event-name: hardware-alarm.

*component-name: It indicates the hardware component responsible for generating this alarm.

*severity: The severity of the alarm such as critical, high, medium, and low.

*message: Simple information such as "The hardware component has failed or degraded" or with extra information.

6.1.5. Interface Alarm

Interface is the network interface for connecting a device with the network. The interface-alarm is emitted when the state of the interface is changed. The following information should be included in an Interface Alarm:

*event-name: interface-alarm.

*interface-name: The name of the interface.

*interface-state: down, up (not congested), congested (up but congested).

*severity: The severity of the alarm such as critical, high, medium, and low.

*message: Simple information such as "The interface is 'interface-state'" or with extra information.

6.2. System Events

System events (as alerts) have the following characteristics:

*acquisition-method: subscription

*emission-type: on-change

*dampening-type: on-repetition

6.2.1. Access Violation

The access-violation system event is an event when a user tries to access (read or write) any information above their privilege. The following information should be included in this event:

*event-name: access-denied.

*user: Name of a user.

*group: Group(s) to which a user belongs. A user can belong to multiple groups.

*ip-address: The IP address of the user that triggered the event.

*authentication: The method to verify the valid user, i.e., pre-configured-key and certificate-authority.

*message: The message to give the context of the event, such as "Access is denied".

6.2.2. Configuration Change

A configuration change is a system event when a new configuration is added or an existing configuration is modified. The following information should be included in this event:

*event-name: config-change.

*user: Name of a user.

*group: Group(s) to which a user belongs. A user can belong to multiple groups.

*ip-address: The IP address of the user that triggered the event.

*authentication: The method to verify the valid user, i.e., pre-configured-key and certificate-authority.

*message: The message to give the context of the event, such as "Configuration is modified" or "New configuration is added".

6.2.3. Session Table Event

The following information should be included in a Session Table Event:

*event-name: session-table.

*current-session: The number of concurrent sessions.

*maximum-session: The maximum number of sessions that the session table can support.

*threshold: The threshold triggering the event.

*message: The message to give the context of the event, such as "The number of session table exceeded the threshold".

6.2.4. Traffic Flows

Traffic flows need to be monitored because they might be used for security attacks to the network. The following information should be included in this event:

*src-ip: The source IPv4 or IPv6 address of the traffic flow.

*dst-ip: The destination IPv4 or IPv6 address of the traffic flow.

*src-port: The source port of the traffic flow.

*dst-port: The destination port of the traffic flow.

*protocol: The protocol of the traffic flow.

*arrival-rate: Arrival rate of packets of the traffic flow.

6.3. NSF Events

NSF events have the following characteristics:

*acquisition-method: subscription

*emission-type: on-change

*dampening-type: on-repetition

6.3.1. DDoS Detection

The following information should be included in a DDoS Event:

*event-name: detection-ddos.

*attack-type: Any one of SYN flood, ACK flood, SYN-ACK flood, FIN/RST flood, TCP Connection flood, UDP flood, ICMP flood, HTTPS flood, HTTP flood, DNS query flood, DNS reply flood, SIP flood, SSL flood, and NTP amplification flood.

*attack-src-ip: The IP address of the source of the DDoS attack.

*attack-dst-ip: The network prefix with a network mask (for IPv4) or prefix length (for IPv6) of a victim under DDoS attack.

*dst-port: The port number that the attack traffic aims at.

*start-time: The time stamp indicating when the attack started.

*end-time: The time stamp indicating when the attack ended. If the attack is still undergoing when sending out the alarm, this field can be empty.

*attack-rate: The packets per second of attack traffic.

*attack-speed: the bits per second of attack traffic.

*rule-name: The name of the I2NSF Policy Rule being triggered. Note that rule-name is used to match a detected NSF event with a policy rule in [I-D.ietf-i2nsf-nsf-facing-interface-dm], and also that there is no rule-name in a system event.

6.3.2. Virus Event

The following information should be included in a Virus Event:

*event-name: detection-virus.

*virus: Type of the virus. e.g., trojan, worm, macro virus type.

*virus-name: Name of the virus.

*dst-ip: The destination IP address of the packet where the virus is found.

*src-ip: The source IP address of the packet where the virus is found.

*src-port: The source port of the packet where the virus is found.

*dst-port: The destination port of the packet where the virus is found.

*src-zone: The source geographical location (e.g., country and city) of the virus.

*dst-zone: The destination geographical location (e.g., country and city) of the virus.

*file-type: The type of the file where the virus is hided within.

*file-name: The name of the file where the virus is hided within.

*raw-info: The information describing the packet triggering the event.

*rule-name: The name of the rule being triggered.

6.3.3. Intrusion Event

The following information should be included in an Intrusion Event:

*event-name: The name of the event. e.g., detection-intrusion.

*attack-type: Attack type, e.g., brutal force and buffer overflow.

*src-ip: The source IP address of the flow.

*dst-ip: The destination IP address of the flow.

*src-port:The source port number of the flow.

*dst-port: The destination port number of the flow

*src-zone: The source geographical location (e.g., country and city) of the flow.

*dst-zone: The destination geographical location (e.g., country and city) of the flow.

*protocol: The employed transport layer protocol. e.g., TCP and UDP.

*app: The employed application layer protocol. e.g., HTTP and FTP.

*rule-name: The name of the I2NSF Policy Rule being triggered.

*raw-info: The information describing the flow triggering the event.

6.3.4. Web Attack Event

The following information should be included in a Web Attack Alarm:

*event-name: The name of event. e.g., detection-web-attack.

*attack-type: Concrete web attack type. e.g., SQL injection, command injection, XSS, CSRF.

*src-ip: The source IP address of the packet.

*dst-ip: The destination IP address of the packet.

*src-port: The source port number of the packet.

*dst-port: The destination port number of the packet.

*src-zone: The source geographical location (e.g., country and city) of the packet.

*dst-zone: The destination geographical location (e.g., country and city) of the packet.

*request-method: The method of requirement. For instance, "PUT" and "GET" in HTTP.

*req-uri: Requested URI.

*response-code: The HTTP Response code.

*req-user-agent: The HTTP request user agent header field.

*req-cookies: The HTTP Cookie previously sent by the server with Set-Cookie.

*req-host: The domain name of the requested host.

*uri-category: Matched URI category.

*filtering-type: URL filtering type. e.g., deny-list, allow-list, and unknown.

*rule-name: The name of the I2NSF Policy Rule being triggered.

6.3.5. VoIP/VoLTE Event

The following information should be included in a VoIP/VoLTE Event:

*source-voice-id: The detected source voice Call ID for VoIP and VoLTE that violates the policy.

*destination-voice-id: The destination voice Call ID for VoIP and VoLTE that violates the policy.

*user-agent: The user agent for VoIP and VoLTE that violates the policy.

*src-ip: The source IP address of the VoIP/VoLTE.

*dst-ip: The destination IP address of the VoIP/VoLTE.

*src-port: The source port number of the VoIP/VoLTE.

*dst-port: The destination port number of VoIP/VoLTE.

*src-zone: The source geographical location (e.g., country and city) of the VoIP/VoLTE.

*dst-zone: The destination geographical location (e.g., country and city) of the VoIP/VoLTE.

*rule-name: The name of the I2NSF Policy Rule being triggered.

6.4. System Logs

System log is a record that is used to monitor the activity of the user on the NSF and the status of the NSF. System logs have the following characteristics:

*acquisition-method: subscription

*emission-type: on-change or periodic

*dampening-type: on-repetition

6.4.1. Access Log

Access logs record administrators' login, logout, and operations on a device. By analyzing them, security vulnerabilities can be identified. The following information should be included in an operation report:

*username: The username that operates on the device.

*login-ip: IP address used by an administrator to log in.

*login-mode: Specifies the administrator logs in mode e.g. administrator, user, and guest.

*operation-type: The operation type that the administrator execute, e.g., login, logout, configuration, and other.

*input: The operation performed by a user after login. The operation is a command given by a user.

*output: The result after executing the input.

6.4.2. Resource Utilization Log

Running reports record the device system's running status, which is useful for device monitoring. The following information should be included in running report:

*system-status: The current system's running status.

*cpu-usage: Specifies the aggregated CPU usage.

*memory-usage: Specifies the memory usage.

*disk-id: Specifies the disk ID to identify the storage disk.

*disk-usage: Specifies the disk usage of disk-id.

*disk-left: Specifies the available disk space left of disk-id.

*session-number: Specifies total concurrent sessions.

*process-number: Specifies total number of systems processes.

*interface-id: Specifies the interface ID to identify the network interface.

*in-traffic-rate: The total inbound traffic rate in packets per second.

*out-traffic-rate: The total outbound traffic rate in packets per second.

*in-traffic-speed: The total inbound traffic speed in bits per second.

*out-traffic-speed: The total outbound traffic speed in bits per second.

6.4.3. User Activity Log

User activity logs provide visibility into users' online records (such as login time, online/lockout duration, and login IP addresses) and the actions that users perform. User activity reports are helpful to identify exceptions during a user's login and network access activities.

*user: Name of a user.

*group: Group to which a user belongs.

*login-ip-addr: Login IP address of a user.

*authentication: The method to verify the valid user, i.e., pre-configured-key and certificate-authority.

*online-duration: The duration of a user's activeness (stays in login) during a session.

*logout-duration: The duration of a user's inactiveness (not in login) from the last session.

*additional-info: Additional Information for login:

1. type: User activities. e.g., Successful User Login, Failed Login attempts, User Logout, Successful User Password Change, Failed User Password Change, User Lockout, and User Unlocking.

2. cause: Cause of a failed user activity.

6.5. NSF Logs

NSF logs have the folowing characteristics:

*acquisition-method: subscription

*emission-type: on-change

*dampening-type: on-repetition

6.5.1. Deep Packet Inspection Log

Deep Packet Inspection (DPI) Logs provide statistics on uploaded and downloaded files and data, sent and received emails, and alert and blocking records on websites. It is helpful to learn risky user behaviors and why access to some URLs is blocked or allowed with an alert record.

*attack-type: DPI action types. e.g., File Blocking, Data Filtering, and Application Behavior Control.

*src-user: User source who generates the policy.

*policy-name: Security policy name that traffic matches.

*action: Action defined in the file blocking rule, data filtering rule, or application behavior control rule that traffic matches.

6.6. System Counter

System counter has the following characteristics:

*acquisition-method: subscription or query

*emission-type: periodic

*dampening-type: none

6.6.1. Interface Counter

Interface counters provide visibility into traffic into and out of an NSF, and bandwidth usage. The statistics of the interface counters should be computed from the start of the service. When the service is reset, the computation of statistics per counter should restart from 0.

- *interface-name: Network interface name configured in NSF.
- *in-total-traffic-pkts: Total inbound packets.
- *out-total-traffic-pkts: Total outbound packets.
- *in-total-traffic-bytes: Total inbound bytes.
- *out-total-traffic-bytes: Total outbound bytes.
- *in-drop-traffic-pkts: Total inbound drop packets.
- *out-drop-traffic-pkts: Total outbound drop packets.
- *in-drop-traffic-bytes: Total inbound drop bytes.
- *out-drop-traffic-bytes: Total outbound drop bytes.
- *in-traffic-average-rate: Inbound traffic average rate in packets per second.
- *in-traffic-peak-rate: Inbound traffic peak rate in packets per second.
- *in-traffic-average-speed: Inbound traffic average speed in bits per second.
- *in-traffic-peak-speed: Inbound traffic peak speed in bits per second.
- *out-traffic-average-rate: Outbound traffic average rate in packets per second.
- *out-traffic-peak-rate: Outbound traffic peak rate in packets per second.
- *out-traffic-average-speed: Outbound traffic average speed in bits per second.
- *out-traffic-peak-speed: Outbound traffic peak speed in bits per second.

6.7. NSF Counters

NSF counters have the following characteristics:

*acquisition-method: subscription or query

*emission-type: periodic

*dampening-type: none

6.7.1. Firewall Counter

Firewall counters provide visibility into traffic signatures, bandwidth usage, and how the configured security and bandwidth policies have been applied.

*src-ip: Source IP address of traffic.

*src-user: User who generates the policy.

*dst-ip: Destination IP address of traffic.

*src-port: Source port of traffic.

*dst-port: Destination port of traffic.

*protocol: Protocol type of traffic.

*app: Application type of traffic.

*policy-id: Security policy id that traffic matches.

*policy-name: Security policy name that traffic matches.

*in-interface: Inbound interface of traffic.

*out-interface: Outbound interface of traffic.

*total-traffic: Total traffic volume.

*in-traffic-average-rate: Inbound traffic average rate in packets per second.

*in-traffic-peak-rate: Inbound traffic peak rate in packets per second.

*in-traffic-average-speed: Inbound traffic average speed in bits per second.

*in-traffic-peak-speed: Inbound traffic peak speed in bits per second.

*out-traffic-average-rate: Outbound traffic average rate in packets per second.

*out-traffic-peak-rate: Outbound traffic peak rate in packets per second.

*out-traffic-average-speed: Outbound traffic average speed in bits per second.

*out-traffic-peak-speed: Outbound traffic peak speed in bits per second.

6.7.2. Policy Hit Counter

Policy Hit Counters record the security policy that traffic matches and its hit count. It can check if policy configurations are correct.

*src-ip: Source IP address of traffic.

*src-user: User who generates the policy.

*dst-ip: Destination IP address of traffic.

*src-port: Source port of traffic.

*dst-port: Destination port of traffic.

*protocol: Protocol type of traffic.

*app: Application type of traffic.

*policy-id: Security policy id that traffic matches.

*policy-name: Security policy name that traffic matches.

*hit-times: The hit times that the security policy matches the specified traffic.

7. NSF Monitoring Management in I2NSF

A standard model for monitoring data is required for an administrator to check the monitoring data generated by an NSF. The administrator can check the monitoring data through the following process. When the NSF monitoring data that is under the standard format is generated, the NSF forwards it to an NSF data collector via the I2NSF NSF Monitoring Interface. The NSF data collector delivers it to I2NSF Consumer or Developer's Management System (DMS) so that the administrator can know the state of the I2NSF framework.

In order to communicate with other components, an I2NSF framework [RFC8329] requires the interfaces. The three main interfaces in I2NSF framework are used for sending monitoring data as follows:

* I2NSF Consumer-Facing Interface [I-D.ietf-i2nsf-consumer-facing-interface-dm]: When an I2NSF User makes a security policy and forwards it to the Security Controller via Consumer-Facing Interface, it can specify the threat-feed for threat prevention, the custom list, the malicious code scan group, and the event map group. They can be used as an event to be monitored by an NSF.

* I2NSF Registration Interface [I-D.ietf-i2nsf-registration-interface-dm]: The Network Functions Virtualization (NFV) architecture provides the lifecycle management of a Virtual Network Function (VNF) via the Ve-Vnfm interface. The role of Ve-Vnfm is to request VNF lifecycle management (e.g., the instantiation and de-instantiation of an NSF, and load balancing among NSFs), exchange configuration information, and exchange status information for a network service. In the I2NSF framework, the DMS manages data about resource states and network traffic for the lifecycle management of an NSF. Therefore, the generated monitoring data from NSFs are delivered from the NSF data collector to the DMS via either Registration Interface or a new interface (e.g., NSF Monitoring Interface). These data are delivered from the DMS to the VNF Manager in the Management and Orchestration (MANO) in the NFV system [I-D.ietf-i2nsf-applicability].

* I2NSF NSF Monitoring Interface [RFC8329]: After a high-level security policy from I2NSF User is translated by security policy translator [I-D.yang-i2nsf-security-policy-translation] in the Security Controller, the translated security policy (i.e., low-level policy) is applied to an NSF via NSF-Facing Interface. The monitoring interface data model for an NSF specifies the list of events that can trigger Event-Condition-Action (ECA) policies via NSF Monitoring Interface.

8. Tree Structure

The tree structure of the NSF monitoring YANG module is provided below:

```
module: ietf-i2nsf-nsf-monitoring
  +--ro i2nsf-counters
  |  +--ro system-interface* [interface-name]
  |  |  +--ro acquisition-method?          identityref
  |  |  +--ro emission-type?               identityref
  |  |  +--ro dampening-type?              identityref
  |  |  +--ro interface-name               string
  |  |  +--ro in-total-traffic-pkts?       yang:counter32
  |  |  +--ro out-total-traffic-pkts?      yang:counter32
  |  |  +--ro in-total-traffic-bytes?      uint64
  |  |  +--ro out-total-traffic-bytes?     uint64
  |  |  +--ro in-drop-traffic-pkts?        yang:counter32
  |  |  +--ro out-drop-traffic-pkts?       yang:counter32
  |  |  +--ro in-drop-traffic-bytes?       uint64
  |  |  +--ro out-drop-traffic-bytes?      uint64
  |  |  +--ro total-traffic?               yang:counter32
  |  |  +--ro in-traffic-average-rate?     uint32
  |  |  +--ro in-traffic-peak-rate?        uint32
  |  |  +--ro in-traffic-average-speed?    uint32
  |  |  +--ro in-traffic-peak-speed?       uint32
  |  |  +--ro out-traffic-average-rate?    uint32
  |  |  +--ro out-traffic-peak-rate?       uint32
  |  |  +--ro out-traffic-average-speed?   uint32
  |  |  +--ro out-traffic-peak-speed?      uint32
  |  |  +--ro message?                     string
  |  |  +--ro vendor-name?                 string
  |  |  +--ro nsf-name?                    union
  |  |  +--ro severity?                    severity
  |  |  +--ro timestamp?                   yang:date-and-time
  |  +--ro nsf-firewall* [policy-name]
  |  |  +--ro acquisition-method?          identityref
  |  |  +--ro emission-type?               identityref
  |  |  +--ro dampening-type?              identityref
  |  |  +--ro policy-name
  |  |       -> /nsfi:i2nsf-security-policy/system-policy-name
  |  |  +--ro src-user?                    string
  |  |  +--ro total-traffic?               yang:counter32
  |  |  +--ro in-traffic-average-rate?     uint32
  |  |  +--ro in-traffic-peak-rate?        uint32
  |  |  +--ro in-traffic-average-speed?    uint32
  |  |  +--ro in-traffic-peak-speed?       uint32
  |  |  +--ro out-traffic-average-rate?    uint32
  |  |  +--ro out-traffic-peak-rate?       uint32
  |  |  +--ro out-traffic-average-speed?   uint32
  |  |  +--ro out-traffic-peak-speed?      uint32
  |  |  +--ro message?                     string
  |  |  +--ro vendor-name?                 string
  |  |  +--ro nsf-name?                    union
```

```
|  |  +--ro severity?                    severity
|  |  +--ro timestamp?                   yang:date-and-time
|  +--ro nsf-policy-hits* [policy-name]
|     +--ro acquisition-method?   identityref
|     +--ro emission-type?        identityref
|     +--ro dampening-type?       identityref
|     +--ro policy-name
          -> /nsfi:i2nsf-security-policy/system-policy-name
|     +--ro src-user?             string
|     +--ro message?              string
|     +--ro vendor-name?          string
|     +--ro nsf-name?             union
|     +--ro severity?             severity
|     +--ro hit-times?            yang:counter32
|     +--ro timestamp?            yang:date-and-time
+--rw i2nsf-monitoring-configuration
   +--rw i2nsf-system-detection-alarm
   |  +--rw enabled?        boolean
   |  +--rw system-alarm* [alarm-type]
   |     +--rw alarm-type          enumeration
   |     +--rw threshold?          uint8
   |     +--rw dampening-period?   uint32
   +--rw i2nsf-system-detection-event
   |  +--rw enabled?            boolean
   |  +--rw dampening-period?   uint32
   +--rw i2nsf-traffic-flows
   |  +--rw dampening-period?   uint32
   |  +--rw enabled?            boolean
   +--rw i2nsf-nsf-detection-ddos {i2nsf-nsf-detection-ddos}?
   |  +--rw enabled?            boolean
   |  +--rw dampening-period?   uint32
   +--rw i2nsf-nsf-detection-session-table-configuration
   |  +--rw enabled?            boolean
   |  +--rw dampening-period?   uint32
   +--rw i2nsf-nsf-detection-intrusion
                 {i2nsf-nsf-detection-intrusion}?
   |  +--rw enabled?            boolean
   |  +--rw dampening-period?   uint32
   +--rw i2nsf-nsf-detection-web-attack
               {i2nsf-nsf-detection-web-attack}?
   |  +--rw enabled?            boolean
   |  +--rw dampening-period?   uint32
   +--rw i2nsf-nsf-system-access-log
   |  +--rw enabled?            boolean
   |  +--rw dampening-period?   uint32
   +--rw i2nsf-system-res-util-log
   |  +--rw enabled?            boolean
   |  +--rw dampening-period?   uint32
   +--rw i2nsf-system-user-activity-log
```

```
   |  +--rw enabled?            boolean
   |  +--rw dampening-period?   uint32
   +--rw i2nsf-nsf-log-dpi {i2nsf-nsf-log-dpi}?
   |  +--rw enabled?            boolean
   |  +--rw dampening-period?   uint32
   +--rw i2nsf-counter
      +--rw period?   uint16

notifications:
  +---n i2nsf-event
  |  +--ro (sub-event-type)?
  |     +--:(i2nsf-system-detection-alarm)
  |     |  +--ro i2nsf-system-detection-alarm
  |     |     +--ro alarm-category?       identityref
  |     |     +--ro component-name?       string
  |     |     +--ro interface-name?       string
  |     |     +--ro interface-state?      enumeration
  |     |     +--ro acquisition-method?   identityref
  |     |     +--ro emission-type?        identityref
  |     |     +--ro dampening-type?       identityref
  |     |     +--ro usage?                uint8
  |     |     +--ro threshold?            uint8
  |     |     +--ro message?              string
  |     |     +--ro vendor-name?          string
  |     |     +--ro nsf-name?             union
  |     |     +--ro severity?             severity
  |     +--:(i2nsf-system-detection-event)
  |     |  +--ro i2nsf-system-detection-event
  |     |     +--ro event-category?       identityref
  |     |     +--ro acquisition-method?   identityref
  |     |     +--ro emission-type?        identityref
  |     |     +--ro dampening-type?       identityref
  |     |     +--ro user                  string
  |     |     +--ro group*                string
  |     |     +--ro ip-address            inet:ip-address
  |     |     +--ro authentication?       identityref
  |     |     +--ro message?              string
  |     |     +--ro vendor-name?          string
  |     |     +--ro nsf-name?             union
  |     |     +--ro severity?             severity
  |     +--:(i2nsf-traffic-flows)
  |     |  +--ro i2nsf-traffic-flows
  |     |     +--ro src-ip?               inet:ip-address
  |     |     +--ro dst-ip?               inet:ip-address
  |     |     +--ro protocol?             identityref
  |     |     +--ro src-port?             inet:port-number
  |     |     +--ro dst-port?             inet:port-number
  |     |     +--ro arrival-rate?         uint32
  |     |     +--ro acquisition-method?   identityref
```

```
|     |     +--ro emission-type?        identityref
|     |     +--ro dampening-type?       identityref
|     |     +--ro message?              string
|     |     +--ro vendor-name?          string
|     |     +--ro nsf-name?             union
|     |     +--ro severity?             severity
|     +--:(i2nsf-nsf-detection-session-table)
|        +--ro i2nsf-nsf-detection-session-table
|           +--ro current-session?   uint32
|           +--ro maximum-session?   uint32
|           +--ro threshold?         uint32
|           +--ro message?           string
|           +--ro vendor-name?       string
|           +--ro nsf-name?          union
|           +--ro severity?          severity
+---n i2nsf-log
|  +--ro (sub-logs-type)?
|     +--:(i2nsf-nsf-system-access-log)
|     |  +--ro i2nsf-nsf-system-access-log
|     |     +--ro login-ip              inet:ip-address
|     |     +--ro username?             string
|     |     +--ro login-role?           login-role
|     |     +--ro operation-type?       operation-type
|     |     +--ro input?                string
|     |     +--ro output?               string
|     |     +--ro acquisition-method?   identityref
|     |     +--ro emission-type?        identityref
|     |     +--ro dampening-type?       identityref
|     |     +--ro message?              string
|     |     +--ro vendor-name?          string
|     |     +--ro nsf-name?             union
|     |     +--ro severity?             severity
|     +--:(i2nsf-system-res-util-log)
|     |  +--ro i2nsf-system-res-util-log
|     |     +--ro system-status?        enumeration
|     |     +--ro cpu-usage?            uint8
|     |     +--ro memory-usage?         uint8
|     |     +--ro disk* [disk-id]
|     |     |  +--ro disk-id       string
|     |     |  +--ro disk-usage?   uint8
|     |     |  +--ro disk-left?    uint8
|     |     +--ro session-num?          uint32
|     |     +--ro process-num?          uint32
|     |     +--ro interface* [interface-id]
|     |     |  +--ro interface-id         string
|     |     |  +--ro in-traffic-rate?     uint32
|     |     |  +--ro out-traffic-rate?    uint32
|     |     |  +--ro in-traffic-speed?    uint32
|     |     |  +--ro out-traffic-speed?   uint32
```

```
|     |     +--ro acquisition-method?   identityref
|     |     +--ro emission-type?        identityref
|     |     +--ro dampening-type?       identityref
|     |     +--ro message?              string
|     |     +--ro vendor-name?          string
|     |     +--ro nsf-name?             union
|     |     +--ro severity?             severity
|     +--:(i2nsf-system-user-activity-log)
|        +--ro i2nsf-system-user-activity-log
|           +--ro acquisition-method?   identityref
|           +--ro emission-type?        identityref
|           +--ro dampening-type?       identityref
|           +--ro user                  string
|           +--ro group*                string
|           +--ro ip-address            inet:ip-address
|           +--ro authentication?       identityref
|           +--ro message?              string
|           +--ro vendor-name?          string
|           +--ro nsf-name?             union
|           +--ro severity?             severity
|           +--ro online-duration?      uint32
|           +--ro logout-duration?      uint32
|           +--ro additional-info?      enumeration
+---n i2nsf-nsf-event
   +--ro (sub-event-type)?
      +--:(i2nsf-nsf-detection-ddos) {i2nsf-nsf-detection-ddos}?
      |  +--ro i2nsf-nsf-detection-ddos
      |     +--ro attack-type?          identityref
      |     +--ro start-time            yang:date-and-time
      |     +--ro end-time              yang:date-and-time
      |     +--ro attack-src-ip*        inet:ip-address
      |     +--ro attack-dst-ip*        inet:ip-prefix
      |     +--ro attack-src-port*      inet:port-number
      |     +--ro attack-dst-port*      inet:port-number
      |     +--ro rule-name
                  -> /nsfi:i2nsf-security-policy/rules/rule-name
      |     +--ro raw-info?             string
      |     +--ro attack-rate?          uint32
      |     +--ro attack-speed?         uint32
      |     +--ro action*               log-action
      |     +--ro acquisition-method?   identityref
      |     +--ro emission-type?        identityref
      |     +--ro dampening-type?       identityref
      |     +--ro message?              string
      |     +--ro vendor-name?          string
      |     +--ro nsf-name?             union
      |     +--ro severity?             severity
      +--:(i2nsf-nsf-detection-virus)
                  {i2nsf-nsf-detection-virus}?
```

```
|  +--ro i2nsf-nsf-detection-virus
|     +--ro dst-ip?               inet:ip-address
|     +--ro dst-port?             inet:port-number
|     +--ro rule-name
          -> /nsfi:i2nsf-security-policy/rules/rule-name
|     +--ro raw-info?             string
|     +--ro src-ip?               inet:ip-address
|     +--ro src-port?             inet:port-number
|     +--ro src-zone?             string
|     +--ro dst-zone?             string
|     +--ro virus?                identityref
|     +--ro virus-name?           string
|     +--ro file-type?            string
|     +--ro file-name?            string
|     +--ro os?                   string
|     +--ro action*               log-action
|     +--ro acquisition-method?   identityref
|     +--ro emission-type?        identityref
|     +--ro dampening-type?       identityref
|     +--ro message?              string
|     +--ro vendor-name?          string
|     +--ro nsf-name?             union
|     +--ro severity?             severity
+--:(i2nsf-nsf-detection-intrusion)
            {i2nsf-nsf-detection-intrusion}?
|  +--ro i2nsf-nsf-detection-intrusion
|     +--ro dst-ip?               inet:ip-address
|     +--ro dst-port?             inet:port-number
|     +--ro rule-name
        -> /nsfi:i2nsf-security-policy/rules/rule-name
|     +--ro raw-info?             string
|     +--ro src-ip?               inet:ip-address
|     +--ro src-port?             inet:port-number
|     +--ro src-zone?             string
|     +--ro dst-zone?             string
|     +--ro protocol?             identityref
|     +--ro app?                  identityref
|     +--ro attack-type?          identityref
|     +--ro action*               log-action
|     +--ro attack-rate?          uint32
|     +--ro attack-speed?         uint32
|     +--ro acquisition-method?   identityref
|     +--ro emission-type?        identityref
|     +--ro dampening-type?       identityref
|     +--ro message?              string
|     +--ro vendor-name?          string
|     +--ro nsf-name?             union
|     +--ro severity?             severity
+--:(i2nsf-nsf-detection-web-attack)
```

```
          {i2nsf-nsf-detection-web-attack}?
|  +--ro i2nsf-nsf-detection-web-attack
|     +--ro dst-ip?               inet:ip-address
|     +--ro dst-port?             inet:port-number
|     +--ro rule-name
          -> /nsfi:i2nsf-security-policy/rules/rule-name
|     +--ro raw-info?             string
|     +--ro src-ip?               inet:ip-address
|     +--ro src-port?             inet:port-number
|     +--ro src-zone?             string
|     +--ro dst-zone?             string
|     +--ro attack-type?          identityref
|     +--ro request-method?       identityref
|     +--ro req-uri?              string
|     +--ro filtering-type*       identityref
|     +--ro req-user-agent?       string
|     +--ro req-cookie?           string
|     +--ro req-host?             string
|     +--ro response-code?        string
|     +--ro acquisition-method?   identityref
|     +--ro emission-type?        identityref
|     +--ro dampening-type?       identityref
|     +--ro action*               log-action
|     +--ro message?              string
|     +--ro vendor-name?          string
|     +--ro nsf-name?             union
|     +--ro severity?             severity
+--:(i2nsf-nsf-detection-voip-volte)
           {i2nsf-nsf-detection-voip-volte}?
|  +--ro i2nsf-nsf-detection-voip-volte
|     +--ro dst-ip?                 inet:ip-address
|     +--ro dst-port?               inet:port-number
|     +--ro rule-name
          -> /nsfi:i2nsf-security-policy/rules/rule-name
|     +--ro raw-info?               string
|     +--ro src-ip?                 inet:ip-address
|     +--ro src-port?               inet:port-number
|     +--ro src-zone?               string
|     +--ro dst-zone?               string
|     +--ro source-voice-id*        string
|     +--ro destination-voice-id*   string
|     +--ro user-agent*             string
+--:(i2nsf-nsf-log-dpi) {i2nsf-nsf-log-dpi}?
   +--ro i2nsf-nsf-log-dpi
      +--ro attack-type?          dpi-type
      +--ro acquisition-method?   identityref
      +--ro emission-type?        identityref
      +--ro dampening-type?       identityref
      +--ro policy-name
```

```
  -> /nsfi:i2nsf-security-policy/system-policy-name
+--ro src-user?             string
+--ro message?              string
+--ro vendor-name?          string
+--ro nsf-name?             union
+--ro severity?             severity
```

Figure 1: Information Model for NSF Monitoring

9. YANG Data Model

This section describes a YANG module of I2NSF NSF Monitoring. The data model provided in this document uses identities to be used to get information of the monitored of an NSF's monitoring data. Every identity used in the document gives information or status about the current situation of an NSF. This YANG module imports from [RFC6991], and makes references to [RFC0768][RFC0791] [RFC0792] [RFC0793] [RFC0959][RFC4443] [RFC8200][RFC8641] [IANA-HTTP-Status-Code] [IANA-Media-Types].

```
<CODE BEGINS> file "ietf-i2nsf-nsf-monitoring@2021-08-24.yang"


module ietf-i2nsf-nsf-monitoring {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-i2nsf-nsf-monitoring";
  prefix
    nsfmi;
  import ietf-inet-types{
    prefix inet;
    reference
      "Section 4 of RFC 6991";
  }
  import ietf-yang-types {
    prefix yang;
    reference
      "Section 3 of RFC 6991";
  }
  import ietf-i2nsf-policy-rule-for-nsf {
    prefix nsfi;
    reference
      "Section 4.1 of draft-ietf-i2nsf-nsf-facing-interface-dm-13";
  }
  organization
    "IETF I2NSF (Interface to Network Security Functions)
     Working Group";
  contact
    "WG Web: <http://tools.ietf.org/wg/i2nsf>
     WG List: <mailto:i2nsf@ietf.org>

     Editor: Jaehoon Paul Jeong
     <mailto:pauljeong@skku.edu>

     Editor: Patrick Lingga
     <mailto:patricklink@skku.edu>";

  description
    "This module is a YANG module for I2NSF NSF Monitoring.

     The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
     'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
     'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this
     document are to be interpreted as described in BCP 14
     (RFC 2119) (RFC 8174) when, and only when, they appear
     in all capitals, as shown here.

     Copyright (c) 2021 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.
```

```
   Redistribution and use in source and binary forms, with or
   without modification, is permitted pursuant to, and subject to
   the license terms contained in, the Simplified BSD License set
   forth in Section 4.c of the IETF Trust's Legal Provisions
   Relating to IETF Documents
   (https://trustee.ietf.org/license-info).

   This version of this YANG module is part of RFC XXXX
   (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
   for full legal notices.";

revision "2021-08-24" {
  description "Latest revision";
  reference
    "RFC XXXX: I2NSF NSF Monitoring Interface YANG Data Model";

  // RFC Ed.: replace XXXX with an actual RFC number and remove
  // this note.
}

/*
 * Typedefs
 */

typedef severity {
  type enumeration {
    enum critical {
      description
        "The 'critical' severity level indicates that
         an immediate corrective action is required.
         A 'critical' severity is reported when a service
         becomes totally out of service and must be restored.";
    }
    enum high {
      description
        "The 'high' severity level indicates that
         an urgent corrective action is required.
         A 'high' severity is reported when there is
         a severe degradation in the capability of the
         service and its full capability must be restored.";
    }
    enum middle {
      description
        "The 'middle' severity level indicates the
         existence of a non-service-affecting fault
         condition and corrective action should be done
         to prevent a more serious fault. The 'middle'
         severity is reported when the detected problem
         is not degrading the capability of the service, but
```

```
          some service degradation might happen if not
          prevented.";
      }
      enum low {
        description
          "The 'low' severity level indicates the detection
          of a potential fault before any effect is observed.
          The 'low' severity is reported when an action should
          be done before a fault happen.";
      }
    }
    description
      "An indicator representing severity levels. The severity
       levels starting from the highest are critical, high, middle,
       and low.";
}

typedef log-action {
    type enumeration {
      enum allow {
        description
          "If action is allowed";
      }
      enum alert {
        description
          "If action is alert";
      }
      enum block {
        description
          "If action is block";
      }
      enum discard {
        description
          "If action is discarded";
      }
      enum declare {
        description
          "If action is declared";
      }
      enum block-ip {
        description
          "If action is block-ip";
      }
      enum block-service{
        description
          "If action is block-service";
      }
    }
    description
```

```
    "The type representing action for logging.";
}

typedef dpi-type{
  type enumeration {
    enum file-blocking{
      description
        "DPI for preventing the specified file types from flowing
         in the network.";
    }
    enum data-filtering{
      description
        "DPI for preventing sensitive information (e.g., Credit
         Card Number or Social Security Numbers) leaving a
         protected network.";
    }
    enum application-behavior-control{
      description
        "DPI for filtering packet based on the application or
         network behavior analysis to identify malicious or
         unusual activity.";
    }
  }
  description
    "The type of Deep Packet Inspection (DPI).
     The defined types are file-blocking, data-filtering, and
     application-behavior-control.";
}

typedef operation-type{
  type enumeration {
    enum login {
      description
        "The operation type is Login.";
    }
    enum logout {
      description
        "The operation type is Logout.";
    }
    enum configuration {
      description
        "The operation type is Configuration. The configuration
         operation includes the command for writing a new
         configuration and modifying an existing configuration.";
    }
    enum other {
      description
        "The operation type is Other operation. This other
         includes all operations done by a user except login,
```

```
          logout, and configuration.";
      }
    }
    description
      "The type of operation done by a user during a session.
       The user operation is not considering their privileges.";
}

typedef login-role {
  type enumeration {
    enum administrator {
      description
        "Administrator (i.e., Super User) login role.
         Non-restricted role.";
    }
    enum user {
      description
        "User login role. Semi-restricted role, some data and
         configurations are available but confidential or important
         data and configuration are restricted.";
    }
    enum guest {
      description
        "Guest login role. Restricted role, only few read data are
         available and write configurations are restricted.";
    }
  }
  description
    "The role of a user after login.";
}

/*
 * Identity
 */

identity characteristics {
  description
    "Base identity for monitoring information
     characteristics";
}
identity acquisition-method {
  base characteristics;
  description
    "The type of acquisition-method. It can be multiple
     types at once.";
}
identity subscription {
  base acquisition-method;
  description
```

```
    "The acquisition-method type is subscription.";
}
identity query {
  base acquisition-method;
  description
    "The acquisition-method type is query.";
}
identity emission-type {
  base characteristics;
  description
    "The type of emission-type.";
}
identity periodic {
  base emission-type;
  description
    "The emission-type type is periodic.";
}
identity on-change {
  base emission-type;
  description
    "The emission-type type is on-change.";
}
identity dampening-type {
  base characteristics;
  description
    "The type of message dampening to stop the rapid transmission
     of messages. The dampening types are on-repetition and
     no-dampening";
}
identity no-dampening {
  base dampening-type;
  description
    "The dampening-type is no-dampening. No-dampening type does
     not limit the transmission for the messages of the same
     type.";
}
identity on-repetition {
  base dampening-type;
  description
    "The dampening-type is on-repetition. On-repetition type limits
     the transmitted on-change message to one message at a certain
     interval.";
}

identity authentication-mode {
  description
    "The authentication mode for a user to connect to the NSF,
     e.g., pre-configured-key and certificate-authority";
}
```

```
identity pre-configured-key {
  base authentication-mode;
  description
    "The pre-configured-key is an authentication using a key
     authentication.";
}
identity certificate-authority {
  base authentication-mode;
  description
    "The certificate-authority (CA) is an authentication using a
     digital certificate.";
}

identity event {
  description
    "Base identity for I2NSF events.";
}

identity system-event {
  base event;
  description
    "Identity for system event";
}

identity system-alarm {
  base event;
  description
    "Base identity for detectable system alarm types";
}

identity memory-alarm {
  base system-alarm;
  description
    "A memory alarm is alerted.";
}
identity cpu-alarm {
  base system-alarm;
  description
    "A CPU alarm is alerted.";
}
identity disk-alarm {
  base system-alarm;
  description
    "A disk alarm is alerted.";
}
identity hardware-alarm {
  base system-alarm;
  description
    "A hardware alarm (i.e., hardware failure) is alerted.";
```

```
}
identity interface-alarm {
  base system-alarm;
  description
    "An interface alarm is alerted.";
}

identity access-violation {
  base system-event;
  description
    "The access-violation system event is an event when a user
     tries to access (read or write) any information above their
     privilege.";
}
identity configuration-change {
  base system-event;
  description
    "The configuration-change system event is an event when a user
     adds a new configuration or modify an existing configuration
     (write configuration).";
}

identity attack-type {
  description
    "The root ID of attack-based notification
     in the notification taxonomy";
}
identity nsf-attack-type {
  base attack-type;
  description
    "This ID is intended to be used
     in the context of NSF event.";
}

identity virus-type {
  base nsf-attack-type;
  description
    "The type of virus. It can be multiple types at once.
     This attack type is associated with a detected
     system-log virus-attack.";
}
identity trojan {
  base virus-type;
  description
    "The virus type is a trojan. Trojan is able to disguise the
     intent of the files or programs to misleads the users.";
}
identity worm {
  base virus-type;
```

```
  description
    "The virus type is a worm. Worm can self-replicate and
     spread through the network automatically.";
}
identity macro {
  base virus-type;
  description
    "The virus type is a macro virus. Macro causes a series of
     threats automatically after the program is executed.";
}
identity boot-sector {
  base virus-type;
  description
    "The virus type is a boot sector virus. Boot sector is a virus
     that infects the core of the computer, affecting the startup
     process.";
}
identity polymorphic {
  base virus-type;
  description
    "The virus type is a polymorphic virus. Polymorphic can
     modify its version when it replicates, making it hard to
     detect.";
}
identity overwrite {
  base virus-type;
  description
    "The virus type is an overwrite virus. Overwrite can remove
     existing software and replace it with malicious code by
     overwriting it.";
}
identity resident {
  base virus-type;
  description
    "The virus-type is a resident virus. Resident saves itself in
     the computer's memory and infects other files and software.";
}
identity non-resident {
  base virus-type;
  description
    "The virus-type is a non-resident virus. Non-resident attaches
     directly to an executable file and enters the device when
     executed.";
}
identity multipartite {
  base virus-type;
  description
    "The virus-type is a multipartite virus. Multipartite attacks
     both the boot sector and executables files of a computer.";
```

```
}
identity spacefiller {
  base virus-type;
  description
    "The virus-type is a spacefiller virus. Spacefiller fills empty
     spaces of a file or software with malicious code.";
}

identity intrusion-attack-type {
  base nsf-attack-type;
  description
    "The attack type is associated with a detected
     system-log intrusion.";
}
identity brute-force {
  base intrusion-attack-type;
  description
    "The intrusion type is brute-force.";
}
identity buffer-overflow {
  base intrusion-attack-type;
  description
    "The intrusion type is buffer-overflow.";
}
identity web-attack-type {
  base nsf-attack-type;
  description
    "The attack type is associated with a detected
     system-log web-attack.";
}
identity command-injection {
  base web-attack-type;
  description
    "The detected web attack type is command injection.";
}
identity xss {
  base web-attack-type;
  description
    "The detected web attack type is XSS.";
}
identity csrf {
  base web-attack-type;
  description
    "The detected web attack type is CSRF.";
}

identity ddos-type {
  base nsf-attack-type;
  description
```

```
    "Base identity for detectable flood types";
}
identity syn-flood {
  base ddos-type;
  description
    "A SYN flood is detected.";
}
identity ack-flood {
  base ddos-type;
  description
    "An ACK flood is detected.";
}
identity syn-ack-flood {
  base ddos-type;
  description
    "A SYN-ACK flood is detected.";
}
identity fin-rst-flood {
  base ddos-type;
  description
    "A FIN-RST flood is detected.";
}
identity tcp-con-flood {
  base ddos-type;
  description
    "A TCP connection flood is detected.";
}
identity udp-flood {
  base ddos-type;
  description
    "A UDP flood is detected.";
}
identity icmpv4-flood {
  base ddos-type;
  description
    "An ICMPv4 flood is detected.";
}
identity icmpv6-flood {
  base ddos-type;
  description
    "An ICMPv6 flood is detected.";
}
identity http-flood {
  base ddos-type;
  description
    "An HTTP flood is detected.";
}
identity https-flood {
  base ddos-type;
```

```
  description
    "An HTTPS flood is detected.";
}
identity dns-query-flood {
  base ddos-type;
  description
    "A Domain Name System (DNS) query flood is detected.";
}
identity dns-reply-flood {
  base ddos-type;
  description
   "A Domain Name System (DNS) reply flood is detected.";
}
identity sip-flood {
  base ddos-type;
  description
    "A Session Initiation Protocol (SIP) flood is detected.";
}
identity ssl-flood {
  base ddos-type;
  description
    "An Secure Sockets Layer (SSL) flood is detected";
}
identity ntp-amp-flood {
  base ddos-type;
  description
    "A Network Time Protocol (NTP) amplification is detected";
}

identity request-method {
  description
    "A set of request types in HTTP (if applicable).";
}
identity put {
  base request-method;
  description
    "The detected request type is PUT.";
  reference
    "RFC 7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics
     and Content - Request Method PUT";
}
identity post {
  base request-method;
  description
    "The detected request type is POST.";
  reference
    "RFC 7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics
     and Content - Request Method POST";
}
```

```
identity get {
  base request-method;
  description
    "The detected request type is GET.";
  reference
    "RFC 7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics
     and Content - Request Method GET";
}
identity head {
  base request-method;
  description
    "The detected request type is HEAD.";
  reference
    "RFC 7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics
     and Content - Request Method HEAD";
}
identity delete {
  base request-method;
  description
    "The detected request type is DELETE.";
  reference
    "RFC 7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics
     and Content - Request Method DELETE";
}
identity connect {
  base request-method;
  description
    "The detected request type is CONNECT.";
  reference
    "RFC 7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics
     and Content - Request Method CONNECT";
}
identity options {
  base request-method;
  description
    "The detected request type is OPTIONS.";
  reference
    "RFC 7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics
     and Content - Request Method OPTIONS";
}
identity trace {
  base request-method;
  description
    "The detected request type is TRACE.";
  reference
    "RFC 7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics
     and Content - Request Method TRACE";
}
```

```
identity filter-type {
  description
    "The type of filter used to detect an attack,
     for example, a web-attack.  It can be applicable to
     more than web-attacks.";
}
identity allow-list {
  base filter-type;
  description
    "The applied filter type is an allow list. This filter blocks
     all connection except the specified list.";
}
identity deny-list {
  base filter-type;
  description
    "The applied filter type is a deny list. This filter opens all
     connection except the specified list.";
}
identity unknown-filter {
  base filter-type;
  description
    "The applied filter is unknown.";
}

identity protocol {
  description
    "An identity used to enable type choices in leaves
     and leaflists with respect to protocol metadata. This is used
     to identify the type of protocol that goes through the NSF.";
}
identity ip {
  base protocol;
  description
    "General IP protocol type.";
  reference
    "RFC 791: Internet Protocol
     RFC 8200: Internet Protocol, Version 6 (IPv6)";
}
identity ipv4 {
  base ip;
  description
    "IPv4 protocol type.";
  reference
    "RFC 791: Internet Protocol";
}
identity ipv6 {
  base ip;
  description
    "IPv6 protocol type.";
```

```
  reference
    "RFC 8200: Internet Protocol, Version 6 (IPv6)";
}
identity icmp {
  base protocol;
  description
    "Base identity for ICMPv4 and ICMPv6 condition capability";
  reference
    "RFC 792: Internet Control Message Protocol
     RFC 4443: Internet Control Message Protocol (ICMPv6)
     for the Internet Protocol Version 6 (IPv6) Specification
     - ICMPv6";
}
identity icmpv4 {
  base icmp;
  description
    "ICMPv4 protocol type.";
  reference
    "RFC 791: Internet Protocol
     RFC 792: Internet Control Message Protocol";
}
identity icmpv6 {
  base icmp;
  description
    "ICMPv6 protocol type.";
  reference
    "RFC 8200: Internet Protocol, Version 6 (IPv6)
     RFC 4443: Internet Control Message Protocol (ICMPv6)
     for the Internet Protocol Version 6 (IPv6)
     Specification";
}
identity transport-protocol {
  base protocol;
  description
    "Base identity for Layer 4 protocol condition capabilities,
     e.g., TCP, UDP, SCTP, DCCP, and ICMP";
}
identity tcp {
  base transport-protocol;
  description
    "TCP protocol type.";
  reference
    "RFC 793: Transmission Control Protocol";
}
identity udp {
  base transport-protocol;
  description
    "UDP protocol type.";
  reference
```

```
    "RFC 768: User Datagram Protocol";
}
identity sctp {
  base transport-protocol;
  description
    "Identity for SCTP condition capabilities";
  reference
    "RFC 4960: Stream Control Transmission Protocol";
}
identity dccp {
  base transport-protocol;
  description
    "Identity for DCCP condition capabilities";
  reference
    "RFC 4340: Datagram Congestion Control Protocol";
}
identity application-protocol {
  base protocol;
  description
    "Base identity for Application protocol, e.g., HTTP, FTP";
}
identity http {
  base application-protocol;
  description
    "HTTP protocol type.";
  reference
    "RFC7230: Hypertext Transfer Protocol (HTTP/1.1): Message
     Syntax and Routing
     RFC7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics
     and Content";
}
identity https {
  base application-protocol;
  description
    "HTTPS protocol type.";
  reference
    "RFC7230: Hypertext Transfer Protocol (HTTP/1.1): Message
     Syntax and Routing
     RFC7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics
     and Content";
}
identity ftp {
  base application-protocol;
  description
    "FTP protocol type.";
  reference
    "RFC 959: File Transfer Protocol";
}
identity ssh {
```

```
  base application-protocol;
  description
    "SSH protocol type.";
  reference
    "RFC 959: File Transfer Protocol";
}
identity telnet {
  base application-protocol;
  description
    "The identity for telnet.";
  reference
    "RFC 854: Telnet Protocol";
}
identity smtp {
  base application-protocol;
  description
    "The identity for smtp.";
  reference
    "RFC 5321: Simple Mail Transfer Protocol (SMTP)";
}
identity sftp {
  base application-protocol;
  description
    "The identity for sftp.";
  reference
    "RFC 913: Simple File Transfer Protocol (SFTP)";
}
identity pop3 {
  base application-protocol;
  description
    "The identity for pop3.";
  reference
    "RFC 1081: Post Office Protocol -Version 3 (POP3)";
}

/*
 * Grouping
 */

grouping timestamp {
  description
    "Grouping for identifying the time of the message.";
  leaf timestamp {
    type yang:date-and-time;
    description
      "Specify the time of a message being delivered.";
  }
}
```

```
grouping common-monitoring-data {
  description
    "A set of common monitoring data that is needed
    as the basic information.";
  leaf message {
    type string;
    description
      "This is a freetext annotation for
       monitoring a notification's content.";
  }
  leaf vendor-name {
    type string;
    description
      "The name of the NSF vendor. The string is unrestricted to
       identify the provider or vendor of the NSF.";
  }
  leaf nsf-name {
    type union {
      type string;
      type inet:ip-address;
    }
    description
      "The name or IP address of the NSF generating the message.
       If the given nsf-name is not IP address, the name can be an
       arbitrary string including FQDN (Fully Qualified Domain
       Name). The name MUST be unique for different NSF to
       identify the NSF that generates the message.";
  }
  leaf severity {
    type severity;
    description
      "The severity of the alarm such as critical, high,
       middle, and low.";
  }
}
grouping characteristics {
  description
    "A set of characteristics of a notification.";
  leaf acquisition-method {
    type identityref {
      base acquisition-method;
    }
    description
      "The acquisition-method for characteristics";
  }
  leaf emission-type {
    type identityref {
      base emission-type;
    }
```

```
   description
      "The emission-type for characteristics";
  }
  leaf dampening-type {
    type identityref {
      base dampening-type;
    }
    description
      "The dampening-type for characteristics";
  }
}
grouping i2nsf-system-alarm-type-content {
  description
    "A set of contents for alarm type notification.";
  leaf usage {
    type uint8 {
      range "0..100";
    }
    units "percent";
    description
      "Specifies the used percentage";
  }
  leaf threshold {
    type uint8 {
      range "0..100";
    }
    units "percent";
    description
      "The threshold percentage triggering the alarm or
       the event";
  }
}
grouping i2nsf-system-event-type-content {
  description
    "System event metadata associated with system events
     caused by user activity.";
  leaf user {
    type string;
    mandatory true;
    description
      "The name of a user";
  }
  leaf-list group {
    type string;
    description
      "The group(s) to which a user belongs.";
  }
  leaf ip-address {
    type inet:ip-address;
```

```
    mandatory true;
    description
      "The IPv4 (or IPv6) address of a user that trigger the
       event.";
  }
  leaf authentication {
    type identityref {
      base authentication-mode;
    }
    description
      "The authentication-mode of a user.";
  }
}
grouping i2nsf-nsf-event-type-content {
  description
    "A set of common IPv4 (or IPv6)-related NSF event
     content elements";
  leaf dst-ip {
    type inet:ip-address;
    description
      "The destination IPv4 (IPv6) address of the packet";
  }
  leaf dst-port {
    type inet:port-number;
    description
      "The destination port of the packet";
  }
  leaf rule-name {
    type leafref {
      path
        "/nsfi:i2nsf-security-policy"
       +"/nsfi:rules/nsfi:rule-name";
    }
    mandatory true;
    description
      "The name of the I2NSF Policy Rule being triggered";
  }
  leaf raw-info {
    type string;
    description
      "The information describing the packet
       triggering the event.";
  }
}
grouping i2nsf-nsf-event-type-content-extend {
  description
    "A set of extended common IPv4 (or IPv6)-related NSF
     event content elements";
  uses i2nsf-nsf-event-type-content;
```

```
  leaf src-ip {
    type inet:ip-address;
    description
      "The source IPv4 (or IPv6) address of the packet";
  }
  leaf src-port {
    type inet:port-number;
    description
      "The source port of the packet";
  }
  leaf src-zone {
    type string {
      length "1..100";
      pattern "[0-9a-zA-Z ]*";
    }
    description
      "The source geographical location (e.g., country and city) of
       the packet.";
  }
  leaf dst-zone {
    type string {
      length "1..100";
      pattern "[0-9a-zA-Z ]*";
    }
    description
      "The destination geographical location (e.g., country and
       city) of the packet.";
  }
}
grouping log-action {
  description
    "A grouping for logging action.";
  leaf-list action {
    type log-action;
    description
      "Action type: allow, alert, block, discard, declare,
       block-ip, block-service";
  }
}
grouping attack-rates {
  description
    "A set of traffic rates for monitoring attack traffic
     data";
  leaf attack-rate {
    type uint32;
    units "pps";
    description
      "The average packets per second (pps) rate of attack
       traffic";
```

```
  }
  leaf attack-speed {
    type uint32;
    units "bps";
    description
      "The average bits per second (bps) speed of attack traffic";
  }
}
grouping traffic-rates {
  description
    "A set of traffic rates for statistics data";
  leaf total-traffic {
    type yang:counter32;
    units "packets";
    description
      "The total number of traffic packets (in and out) in the
       NSF.";
  }
  leaf in-traffic-average-rate {
    type uint32;
    units "pps";
    description
      "Inbound traffic average rate in packets per second (pps).
       The average is calculated from the start of the NSF service
       until the generation of this record.";
  }
  leaf in-traffic-peak-rate {
    type uint32;
    units "pps";
    description
      "Inbound traffic peak rate in packets per second (pps).";
  }
  leaf in-traffic-average-speed {
    type uint32;
    units "bps";
    description
      "Inbound traffic average speed in bits per second (bps).
       The average is calculated from the start of the NSF service
       until the generation of this record.";
  }
  leaf in-traffic-peak-speed {
    type uint32;
    units "bps";
    description
      "Inbound traffic peak speed in bits per second (bps).";
  }
  leaf out-traffic-average-rate {
    type uint32;
    units "pps";
```

```
    description
      "Outbound traffic average rate in packets per second (pps).
       The average is calculated from the start of the NSF service
       until the generation of this record.";
  }
  leaf out-traffic-peak-rate {
    type uint32;
    units "pps";
    description
     "Outbound traffic peak rate in packets per Second (pps).";
  }
  leaf out-traffic-average-speed {
    type uint32;
    units "bps";
    description
      "Outbound traffic average speed in bits per second (bps).
       The average is calculated from the start of the NSF service
       until the generation of this record.";
  }
  leaf out-traffic-peak-speed {
    type uint32;
    units "bps";
    description
      "Outbound traffic peak speed in bits per second (bps).";
  }
}
grouping i2nsf-system-counter-type-content{
  description
    "A set of counters for an interface traffic data.";
  leaf interface-name {
    type string;
    description
      "Network interface name configured in an NSF";
  }
  leaf in-total-traffic-pkts {
    type yang:counter32;
    description
      "Total inbound packets";
  }
  leaf out-total-traffic-pkts {
    type yang:counter32;
    description
      "Total outbound packets";
  }
  leaf in-total-traffic-bytes {
    type uint64;
    units "bytes";
    description
      "Total inbound bytes";
```

```
  }
  leaf out-total-traffic-bytes {
    type uint64;
    units "bytes";
    description
      "Total outbound bytes";
  }
  leaf in-drop-traffic-pkts {
    type yang:counter32;
    description
      "Total inbound drop packets";
  }
  leaf out-drop-traffic-pkts {
    type yang:counter32;
    description
      "Total outbound drop packets";
  }
  leaf in-drop-traffic-bytes {
    type uint64;
    units "bytes";
    description
      "Total inbound drop bytes";
  }
  leaf out-drop-traffic-bytes {
    type uint64;
    units "bytes";
    description
      "Total outbound drop bytes";
  }
  uses traffic-rates;
}

grouping i2nsf-nsf-counters-type-content{
  description
    "A set of contents of a policy in an NSF.";
  leaf policy-name {
    type leafref {
      path
        "/nsfi:i2nsf-security-policy"
       +"/nsfi:system-policy-name";
    }
    mandatory true;
    description
      "The name of the policy being triggered";
  }
  leaf src-user{
    type string;
    description
      "The I2NSF User's name who generates the policy.";
```

```
  }
}

grouping enable-notification {
  description
    "A grouping for enabling or disabling notification";
  leaf enabled {
    type boolean;
    default "true";
    description
      "Enables or Disables the notification.
       If 'true', then the notification is enabled.
       If 'false, then the notification is disabled.";
  }
}

grouping dampening {
  description
    "A grouping for dampening period of notification.";
  leaf dampening-period {
    type uint32;
    units "centiseconds";
    default "0";
    description
      "Specifies the minimum interval between the assembly of
       successive update records for a single receiver of a
       subscription. Whenever subscribed objects change and
       a dampening-period interval (which may be zero) has
       elapsed since the previous update record creation for
       a receiver, any subscribed objects and properties
       that have changed since the previous update record
       will have their current values marshalled and placed
       in a new update record. But if the subscribed objects change
       when the dampening-period is active, it should update the
       record without sending the notification until the dampening-
       period is finished. If multiple changes happen during the
       active dampening-period, it should update the record with
       the latest data. And at the end of the dampening-period, it
       should send the record as a notification with the latest
       updated record and restart the countdown.";
    reference
      "RFC 8641:  Subscription to YANG Notifications for
       Datastore Updates - Section 5.";
  }
}

/*
 * Feature Nodes
 */
```

```
feature i2nsf-nsf-detection-ddos {
  description
    "This feature means it supports I2NSF nsf-detection-ddos
     notification";
}
feature i2nsf-nsf-detection-virus {
  description
    "This feature means it supports I2NSF nsf-detection-virus
     notification";
}
feature i2nsf-nsf-detection-intrusion {
  description
    "This feature means it supports I2NSF nsf-detection-intrusion
     notification";
}
feature i2nsf-nsf-detection-web-attack {
  description
    "This feature means it supports I2NSF nsf-detection-web-attack
     notification";
}
feature i2nsf-nsf-detection-voip-volte {
  description
    "This feature means it supports I2NSF nsf-detection-voip-volte
     notification";
}
feature i2nsf-nsf-log-dpi {
  description
    "This feature means it supports I2NSF nsf-log-dpi
     notification";
}

/*
 * Notification nodes
 */

notification i2nsf-event {
  description
    "Notification for I2NSF Event.";
  choice sub-event-type {
    description
      "This choice must be augmented with cases for each allowed
       sub-event. Only 1 sub-event will be instantiated in each
       i2nsf-event message. Each case is expected to define one
       container with all the sub-event fields.";
    case i2nsf-system-detection-alarm {
      container i2nsf-system-detection-alarm{
        description
          "This notification is sent, when a system alarm
```

```
       is detected.";
    leaf alarm-category {
      type identityref {
       base system-alarm;
      }
      description
        "The alarm category for
         system-detection-alarm notification";
    }
    leaf component-name {
      type string;
      description
        "The hardware component responsible for generating
         the message. Applicable for Hardware Failure
         Alarm.";
    }
    leaf interface-name {
      type string;
      description
        "The interface name responsible for generating
         the message. Applicable for Network Interface
         Failure Alarm.";
    }
    leaf interface-state {
      type enumeration {
        enum down {
          description
            "The interface state is down.";
        }
        enum up {
          description
            "The interface state is up and not congested.";
        }
        enum congested {
          description
            "The interface state is up but congested.";
        }
      }
      description
        "The state of the interface (i.e., up, down,
         congested). Applicable for Network Interface Failure
         Alarm.";
    }
    uses characteristics;
    uses i2nsf-system-alarm-type-content;
    uses common-monitoring-data;
  }
}
```

```
case i2nsf-system-detection-event {
  container i2nsf-system-detection-event {
    description
      "This notification is sent when a security-sensitive
       authentication action fails.";
    leaf event-category {
      type identityref {
        base system-event;
      }
      description
        "The event category for system-detection-event";
    }
    uses characteristics;
    uses i2nsf-system-event-type-content;
    uses common-monitoring-data;
  }
}

case i2nsf-traffic-flows {
  container i2nsf-traffic-flows {
    description
      "This notification is sent to inform about the traffic
       flows.";
    leaf src-ip {
      type inet:ip-address;
      description
        "The source IPv4 (or IPv6) address of the flow";
    }
    leaf dst-ip {
      type inet:ip-address;
      description
        "The destination IPv4 (or IPv6) address of the flow";
    }
    leaf protocol {
      type identityref {
        base protocol;
      }
      description
        "The protocol type for nsf-detection-intrusion
         notification";
    }
    leaf src-port {
      type inet:port-number;
      description
        "The source port of the flow";
    }
    leaf dst-port {
      type inet:port-number;
      description
```

```
            "The destination port of the flow";
        }
        leaf arrival-rate {
          type uint32;
          units "pps";
          description
            "The average arrival rate of the flow in packets per
             second. The average is calculated from the start of
             the NSF service until the generation of this
             record.";
        }
        uses characteristics;
        uses common-monitoring-data;
      }
    }

    case i2nsf-nsf-detection-session-table {
      container i2nsf-nsf-detection-session-table {
        description
          "This notification is sent, when a session table
           event is detected.";
        leaf current-session {
          type uint32;
          description
            "The number of concurrent sessions";
        }
        leaf maximum-session {
          type uint32;
          description
            "The maximum number of sessions that the session
             table can support";
        }
        leaf threshold {
          type uint32;
          description
            "The threshold triggering the event";
        }
        uses common-monitoring-data;
      }
    }
  }
}

notification i2nsf-log {
  description
    "Notification for I2NSF log. The notification is generated
     from the logs of the NSF.";
  choice sub-logs-type {
    description
```

```
  "This choice must be augmented with cases for each allowed
   sub-logs. Only 1 sub-event will be instantiated in each
   i2nsf-logs message. Each case is expected to define one
   container with all the sub-logs fields.";
case i2nsf-nsf-system-access-log {
  container i2nsf-nsf-system-access-log {
    description
      "The notification is sent, if there is a new system
       log entry about a system access event.";
    leaf login-ip {
      type inet:ip-address;
      mandatory true;
      description
        "Login IP address of a user";
    }
    leaf username {
      type string;
      description
        "The login username that maintains the device";
    }
    leaf login-role {
      type login-role;
      description
        "Specifies the user log-in role, i.e., administrator,
         user, or guest.";
    }
    leaf operation-type {
      type operation-type;
      description
        "The operation type that the user executes";
    }
    leaf input {
      type string;
      description
        "The operation performed by a user after login. The
         operation is a command given by a user.";
    }
    leaf output {
      type string;
      description
        "The result in text format after executing the
         input.";
    }
    uses characteristics;
    uses common-monitoring-data;
  }
}

case i2nsf-system-res-util-log {
```

```
container i2nsf-system-res-util-log {
  description
    "This notification is sent, if there is a new log
     entry representing resource utilization updates.";
  leaf system-status {
    type enumeration {
      enum running {
        description
          "The system is active and running the security
           service.";
      }
      enum waiting {
        description
          "The system is active but waiting for an event to
           provide the security service.";
      }
      enum inactive {
        description
          "The system is inactive and not running the
           security service.";
      }
    }
    description
      "The current system's running status";
  }
  leaf cpu-usage {
    type uint8;
    units "percent";
    description
      "Specifies the relative percentage of CPU usage with
       respect to platform resources";
  }
  leaf memory-usage {
    type uint8;
    units "percent";
    description
      "Specifies the percentage of memory usage.";
  }
  list disk {
    key disk-id;
    description
      "Disk is the hardware to store information for a
       long period, i.e., Hard Disk or Solid-State Drive.";
    leaf disk-id {
      type string;
      description
        "The ID of the storage disk. It is a free form
         identifier to identify the storage disk.";
    }
```

```
  leaf disk-usage {
    type uint8;
    units "percent";
    description
      "Specifies the percentage of disk usage";
  }
  leaf disk-left {
    type uint8;
    units "percent";
    description
      "Specifies the percentage of disk left";
  }
}
leaf session-num {
  type uint32;
  description
    "The total number of sessions";
}
leaf process-num {
  type uint32;
  description
    "The total number of processes";
}
list interface {
  key interface-id;
  description
    "The network interface for connecting a device
     with the network.";
  leaf interface-id {
    type string;
    description
      "The ID of the network interface. It is a free form
       identifier to identify the network interface.";
  }
  leaf in-traffic-rate {
    type uint32;
    units "pps";
    description
      "The total inbound traffic rate in packets per
       second";
  }
  leaf out-traffic-rate {
    type uint32;
    units "pps";
    description
       "The total outbound traffic rate in packets per
        second";
  }
  leaf in-traffic-speed {
```

```
        type uint32;
        units "bps";
        description
          "The total inbound traffic speed in bits per second";
      }
      leaf out-traffic-speed {
        type uint32;
        units "bps";
        description
          "The total outbound traffic speed in bits per
           second";
      }
    }
    uses characteristics;
    uses common-monitoring-data;
  }
}

case i2nsf-system-user-activity-log {
  container i2nsf-system-user-activity-log {
    description
      "This notification is sent, if there is a new user
       activity log entry.";
    uses characteristics;
    uses i2nsf-system-event-type-content;
    uses common-monitoring-data;
    leaf online-duration {
      type uint32;
      units "seconds";
      description
        "The duration of a user's activeness (stays in login)
         during a session.";

    }
    leaf logout-duration {
      type uint32;
      units "seconds";
      description
        "The duration of a user's inactiveness (not in login)
         from the last session.";
    }
    leaf additional-info {
      type enumeration {
        enum successful-login {
          description
            "The user has succeeded in login.";
        }
        enum failed-login {
          description
```

```
                "The user has failed in login (e.g., wrong
                 password)";
              }
              enum logout {
                description
                  "The user has succeeded in logout";
              }
              enum successful-password-changed {
                description
                  "The password has been changed successfully";
              }
              enum failed-password-changed{
                description
                  "The attempt to change password has failed";
              }
              enum lock {
                description
                  "The user has been locked. A locked user cannot
                   login.";
              }
              enum unlock {
                description
                  "The user has been unlocked.";
              }
            }
            description
              "User activities, e.g., Successful User Login,
               Failed Login attempts, User Logout, Successful User
               Password Change, Failed User Password Change, User
               Lockout, User Unlocking, and Unknown.";
          }
        }
      }
    }
  }

notification i2nsf-nsf-event {
  description
    "Notification for I2NSF NSF Event. This notification is
     used for a specific NSF that supported such feature.";
  choice sub-event-type {
    description
    "This choice must be augmented with cases for each allowed
     sub-event. Only 1 sub-event will be instantiated in each
     i2nsf-event message. Each case is expected to define one
     container with all the sub-event fields.";
    case i2nsf-nsf-detection-ddos {
      if-feature "i2nsf-nsf-detection-ddos";
      container i2nsf-nsf-detection-ddos {
```

```
description
  "This notification is sent, when a specific flood type
   is detected.";
leaf attack-type {
  type identityref {
    base ddos-type;
  }
  description
    "Any one of Syn flood, ACK flood, SYN-ACK flood,
     FIN/RST flood, TCP Connection flood, UDP flood,
     ICMP (i.e., ICMPv4 or ICMPv6) flood, HTTP flood,
     HTTPS flood, DNS query flood, DNS reply flood, SIP
     flood, etc.";
}
leaf start-time {
  type yang:date-and-time;
  mandatory true;
  description
    "The time stamp indicating when the attack started";
}
leaf end-time {
  type yang:date-and-time;
  mandatory true;
  description
    "The time stamp indicating when the attack ended";
}
leaf-list attack-src-ip {
  type inet:ip-address;
  description
    "The source IPv4 (or IPv6) addresses of attack
     traffic. It can hold multiple IPv4 (or IPv6)
     addresses.";
}
leaf-list attack-dst-ip {
  type inet:ip-prefix;
  description
    "The destination IPv4 (or IPv6) addresses of attack
     traffic. It can hold multiple IPv4 (or IPv6)
     addresses.";
}
leaf-list attack-src-port {
  type inet:port-number;
  description
    "The source ports of the DDoS attack";
}
leaf-list attack-dst-port {
  type inet:port-number;
  description
    "The destination ports of the DDoS attack";
```

```
    }
    leaf rule-name {
      type leafref {
        path
          "/nsfi:i2nsf-security-policy"
         +"/nsfi:rules/nsfi:rule-name";
      }
      mandatory true;
      description
        "The name of the I2NSF Policy Rule being triggered";
    }
    leaf raw-info {
      type string;
      description
        "The information describing the packet
         triggering the event.";
    }
    uses attack-rates;
    uses log-action;
    uses characteristics;
    uses common-monitoring-data;
  }
}
case i2nsf-nsf-detection-virus {
  if-feature "i2nsf-nsf-detection-virus";
  container i2nsf-nsf-detection-virus {
    description
      "This notification is sent, when a virus is detected.";
    uses i2nsf-nsf-event-type-content-extend;
    leaf virus {
      type identityref {
        base virus-type;
      }
      description
        "The virus type for nsf-detection-virus notification";
    }
    leaf virus-name {
      type string;
      description
        "The name of the detected virus";
    }
    leaf file-type {
      type string;
      description
        "The type of file virus code is found in (if
         applicable).";
      reference
        "IANA Website: Media Types";
    }
```

```
    leaf file-name {
      type string;
      description
        "The name of file virus code is found in (if
         applicable).";
    }
    leaf os {
      type string;
      description
        "The operating system of the device.";
    }
    uses log-action;
    uses characteristics;
    uses common-monitoring-data;
  }
}
case i2nsf-nsf-detection-intrusion {
  if-feature "i2nsf-nsf-detection-intrusion";
  container i2nsf-nsf-detection-intrusion {
    description
      "This notification is sent, when an intrusion event
       is detected.";
    uses i2nsf-nsf-event-type-content-extend;
    leaf protocol {
      type identityref {
        base transport-protocol;
      }
      description
        "The transport protocol type for
         nsf-detection-intrusion notification";
    }
    leaf app {
      type identityref {
        base application-protocol;
      }
      description
        "The employed application layer protocol";
    }
    leaf attack-type {
      type identityref {
        base intrusion-attack-type;
      }
      description
        "The sub attack type for intrusion attack";
    }
    uses log-action;
    uses attack-rates;
    uses characteristics;
    uses common-monitoring-data;
```

```
  }
}
case i2nsf-nsf-detection-web-attack {
  if-feature "i2nsf-nsf-detection-web-attack";
  container i2nsf-nsf-detection-web-attack {
    description
      "This notification is sent, when an attack event is
       detected.";
    uses i2nsf-nsf-event-type-content-extend;
    leaf attack-type {
      type identityref {
        base web-attack-type;
      }
      description
        "Concrete web attack type, e.g., SQL injection,
         command injection, XSS, and CSRF.";
    }
    leaf request-method {
      type identityref {
        base request-method;
      }
      description
        "The HTTP request method, e.g., PUT or GET.";
      reference
        "RFC 7231: Hypertext Transfer Protocol (HTTP/1.1):
         Semantics and Content - Request Methods";
    }
    leaf req-uri {
      type string;
      description
        "The Requested URI";
    }
    leaf-list filtering-type {
      type identityref {
        base filter-type;
      }
      description
        "URL filtering type, e.g., deny-list, allow-list,
         and Unknown";
    }
    leaf req-user-agent {
      type string;
      description
        "The request user agent";
    }
    leaf req-cookie {
      type string;
      description
        "The HTTP Cookie previously sent by the server with
```

```
          Set-Cookie";
      }
      leaf req-host {
        type string;
        description
          "The domain name of the requested host";
      }
      leaf response-code {
        type string;
        description
          "The HTTP Response code";
        reference
          "IANA Website: Hypertext Transfer Protocol (HTTP)
           Status Code Registry";
      }
      uses characteristics;
      uses log-action;
      uses common-monitoring-data;
    }
  }
  case i2nsf-nsf-detection-voip-volte{
    if-feature "i2nsf-nsf-detection-voip-volte";
    container i2nsf-nsf-detection-voip-volte {
      description
        "This notification is sent, when a VoIP/VoLTE violation
         is detected.";
      uses i2nsf-nsf-event-type-content-extend;
      leaf-list source-voice-id {
        type string;
        description
          "The detected source voice ID for VoIP and VoLTE that
           violates the security policy.";
      }
      leaf-list destination-voice-id {
        type string;
        description
          "The detected destination voice ID for VoIP and VoLTE
           that violates the security policy.";
      }
      leaf-list user-agent {
        type string;
        description
          "The detected user-agent for VoIP and VoLTE that violates
           the security policy.";
      }
    }
  }
  case i2nsf-nsf-log-dpi {
    if-feature "i2nsf-nsf-log-dpi";
```

```
      container i2nsf-nsf-log-dpi {
        description
          "This notification is sent, if there is a new DPI
           event in the NSF log.";
        leaf attack-type {
          type dpi-type;
          description
            "The type of the DPI";
        }
        uses characteristics;
        uses i2nsf-nsf-counters-type-content;
        uses common-monitoring-data;
      }
    }
  }
}
/*
 * Data nodes
 */
container i2nsf-counters {
  config false;
  description
    "This is probably better covered by an import as this
     will not be notifications.  Counters are not very
     suitable as telemetry, maybe via periodic
     subscriptions, which would still violate the principle
     of least surprise.";
  list system-interface {
    key interface-name;
    description
      "Interface counters provide the visibility of traffic into
       and out of an NSF, and bandwidth usage.";
    uses characteristics;
    uses i2nsf-system-counter-type-content;
    uses common-monitoring-data;
    uses timestamp;
  }
  list nsf-firewall {
    key policy-name;
    description
      "Firewall counters provide the visibility of traffic
       signatures, bandwidth usage, and how the configured security
       and bandwidth policies have been applied.";
    uses characteristics;
    uses i2nsf-nsf-counters-type-content;
    uses traffic-rates;
    uses common-monitoring-data;
    uses timestamp;
  }
```

```
  list nsf-policy-hits {
    key policy-name;
    description
      "Policy Hit Counters record the number of hits that traffic
       packets match a security policy. It can check if policy
       configurations are correct or not.";
    uses characteristics;
    uses i2nsf-nsf-counters-type-content;
    uses common-monitoring-data;
    leaf hit-times {
      type yang:counter32;
      description
        "The number of times a policy is hit";
    }
    uses timestamp;
  }
}

container i2nsf-monitoring-configuration {
  description
    "The container for configuring I2NSF monitoring.";
  container i2nsf-system-detection-alarm {
    description
      "The container for configuring I2NSF system-detection-alarm
       notification";
    uses enable-notification;
    list system-alarm {
      key alarm-type;
      description
        "Configuration for system alarm (i.e., CPU, Memory,
         and Disk Usage)";
      leaf alarm-type {
        type enumeration {
          enum CPU {
            description
              "To configure the CPU usage threshold to trigger the
               CPU-USAGE-ALARM";
          }
          enum Memory {
            description
              "To configure the Memory usage threshold to trigger
               the MEM-USAGE-ALARM";
          }
          enum Disk {
            description
              "To configure the Disk (storage) usage threshold to
               trigger the DISK-USAGE-ALARM";
          }
        }
```

```
      description
        "Type of alarm to be configured";
    }
    leaf threshold {
      type uint8 {
        range "1..100";
      }
      units "percent";
      description
        "The configuration for threshold percentage to trigger
         the alarm. The alarm will be triggered if the usage
         is exceeded the threshold.";
    }
    uses dampening;
  }
}
container i2nsf-system-detection-event {
  description
    "The container for configuring I2NSF system-detection-event
     notification";
  uses enable-notification;
  uses dampening;
}
container i2nsf-traffic-flows {
  description
    "The container for configuring I2NSF traffic-flows
     notification";
  uses dampening;
  uses enable-notification;
}
container i2nsf-nsf-detection-ddos {
  if-feature "i2nsf-nsf-detection-ddos";
  description
    "The container for configuring I2NSF nsf-detection-ddos
     notification";
  uses enable-notification;
  uses dampening;
}
container i2nsf-nsf-detection-session-table-configuration {
  description
    "The container for configuring I2NSF nsf-detection-session-
     table notification";
  uses enable-notification;
  uses dampening;
}
container i2nsf-nsf-detection-intrusion {
  if-feature "i2nsf-nsf-detection-intrusion";
  description
    "The container for configuring I2NSF nsf-detection-intrusion
```

```
     notification";
  uses enable-notification;
  uses dampening;
}
container i2nsf-nsf-detection-web-attack {
  if-feature "i2nsf-nsf-detection-web-attack";
  description
    "The container for configuring I2NSF nsf-detection-web-attack
     notification";
  uses enable-notification;
  uses dampening;
}
container i2nsf-nsf-system-access-log {
  description
    "The container for configuring I2NSF system-access-log
     notification";
  uses enable-notification;
  uses dampening;
}
container i2nsf-system-res-util-log {
  description
    "The container for configuring I2NSF system-res-util-log
     notification";
  uses enable-notification;
  uses dampening;
}
container i2nsf-system-user-activity-log {
  description
    "The container for configuring I2NSF system-user-activity-log
     notification";
  uses enable-notification;
  uses dampening;
}
container i2nsf-nsf-log-dpi {
  if-feature "i2nsf-nsf-log-dpi";
  description
    "The container for configuring I2NSF nsf-log-dpi
     notification";
  uses enable-notification;
  uses dampening;
}
container i2nsf-counter {
  description
    "This is used to configure the counters
     for monitoring an NSF";
  leaf period {
    type uint16;
    units "minutes";
    default 0;
```

```
        description
          "The configuration for the period interval of reporting
           the counter. If 0, then the counter period is disabled.
           If value is not 0, then the counter will be reported
           following the period value.";
      }
    }
  }
}

<CODE ENDS>
```

Figure 2: Data Model of Monitoring

10. I2NSF Event Stream

This section discusses the NETCONF event stream for I2NSF NSF Monitoring subscription. The YANG module in this document supports "ietf-subscribed-notifications" YANG module [RFC8639] for subscription. The reserved event stream name for this document is "I2NSF-Monitoring". The NETCONF Server (e.g., an NSF) MUST support "I2NSF-Monitoring" event stream for an NSF data collector (e.g., Security Controller). The "I2NSF-Monitoring" event stream contains all I2NSF events described in this document. The following example shows the capabilities of the event streams of an NSF (e.g., "NETCONF" and "I2NSF-Monitoring" event streams) by the subscription of an NSF data collector; note that this example XML file is delivered by an NSF to an NSF data collector:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="1"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <netconf xmlns="urn:ietf:params:xml:ns:netmod:notification">
      <streams>
        <stream>
          <name>NETCONF</name>
          <description>Default NETCONF Event Stream</description>
          <replaySupport>false</replaySupport>
        </stream>
        <stream>
          <name>I2NSF-Monitoring</name>
          <description>I2NSF Monitoring Event Stream</description>
          <replaySupport>true</replaySupport>
          <replayLogCreationTime>
            2021-04-29T09:37:39+00:00
          </replayLogCreationTime>
        </stream>
      </streams>
    </netconf>
  </data>
</rpc-reply>
```

Figure 3: Example of NETCONF Server supporting I2NSF-Monitoring Event Stream

11. XML Examples for I2NSF NSF Monitoring

This section shows the XML examples of I2NSF NSF Monitoring data delivered via Monitoring Interface from an NSF.

11.1. I2NSF System Detection Alarm

The following example shows an alarm triggered by Memory Usage of the server; note that this example XML file is delivered by an NSF to an NSF data collector:

```
<?xml version="1.0" encoding="UTF-8"?>
<notification
 xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2021-04-29T07:43:52.181088+00:00</eventTime>
  <i2nsf-event
    xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-nsf-monitoring">
    <i2nsf-system-detection-alarm>
      <alarm-category
        xmlns:nsfmi="urn:ietf:params:xml:ns:yang:\
                    ietf-i2nsf-nsf-monitoring">
        nsfmi:memory-alarm
      </alarm-category>
      <acquisition-method
        xmlns:nsfmi="urn:ietf:params:xml:ns:yang:\
                      ietf-i2nsf-nsf-monitoring">
        nsfmi:subscription
      </acquisition-method>
      <emission-type
        xmlns:nsfmi="urn:ietf:params:xml:ns:yang:\
                      ietf-i2nsf-nsf-monitoring">
        nsfmi:on-change
      </emission-type>
      <dampening-type
        xmlns:nsfmi="urn:ietf:params:xml:ns:yang:\
                      ietf-i2nsf-nsf-monitoring">
        nsfmi:on-repetition
      </dampening-type>
      <usage>91</usage>
      <threshold>90</threshold>
      <message>Memory Usage Exceeded the Threshold</message>
      <nsf-name>time_based_firewall</nsf-name>
      <severity>high</severity>
    </i2nsf-system-detection-alarm>
  </i2nsf-event>
</notification>
```

Figure 4: Example of I2NSF System Detection Alarm triggered by Memory Usage

The XML data above shows:

1. The NSF that sends the information is named "time_based_firewall".

2. The memory usage of the NSF triggered the alarm.

3. The monitoring information is received by subscription method.

4. The monitoring information is emitted "on-change".

5. The monitoring information is dampened "on-repetition".

6. The memory usage of the NSF is 91 percent.

7. The memory threshold to trigger the alarm is 90 percent.

8. The severity level of the notification is high.

11.2. I2NSF Interface Counters

To get the I2NSF system interface counters information by query, NETCONF Client (e.g., NSF data collector) needs to initiate GET connection with NETCONF Server (e.g., NSF). The following XML file can be used to get the state data and filter the information.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <get>
    <filter
      xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-nsf-monitoring">
      <i2nsf-counters>
        <system-interface/>
      </i2nsf-counters>
    </filter>
  </get>
</rpc>
```

Figure 5: XML Example for NETCONF GET with System Interface Filter

The following XML file shows the reply from the NETCONF Server (e.g., NSF):

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="1"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <i2nsf-counters
      xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-nsf-monitoring">
      <system-interface>
        <interface-name>ens3</interface-name>
        <acquisition-method
          xmlns:nsfmi="urn:ietf:params:xml:ns:yang:\
                       ietf-i2nsf-nsf-monitoring">
          nsfmi:query
        </acquisition-method>
        <in-total-traffic-bytes>549050</in-total-traffic-bytes>
        <out-total-traffic-bytes>814956</out-total-traffic-bytes>
        <in-drop-traffic-bytes>0</in-drop-traffic-bytes>
        <out-drop-traffic-bytes>5078</out-drop-traffic-bytes>
        <nsf-name>time_based_firewall</nsf-name>
      </system-interface>
      <system-interface>
        <interface-name>lo</interface-name>
        <acquisition-method
          xmlns:nsfmi="urn:ietf:params:xml:ns:yang:\
                       ietf-i2nsf-nsf-monitoring">
          nsfmi:query
        </acquisition-method>
        <in-total-traffic-bytes>48487</in-total-traffic-bytes>
        <out-total-traffic-bytes>48487</out-total-traffic-bytes>
        <in-drop-traffic-bytes>0</in-drop-traffic-bytes>
        <out-drop-traffic-bytes>0</out-drop-traffic-bytes>
        <nsf-name>time_based_firewall</nsf-name>
      </system-interface>
    </i2nsf-counters>
  </data>
</rpc-reply>
```

Figure 6: Example of I2NSF System Interface Counters XML Information

12. IANA Considerations

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

URI: urn:ietf:params:xml:ns:yang:ietf-i2nsf-nsf-monitoring
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

This document requests IANA to register the following YANG module in the "YANG Module Names" registry [RFC7950][RFC8525]:

```
name: ietf-i2nsf-nsf-monitoring
namespace: urn:ietf:params:xml:ns:yang:ietf-i2nsf-nsf-monitoring
prefix: nsfmi
reference: RFC XXXX

// RFC Ed.: replace XXXX with an actual RFC number and remove
// this note.
```

13. Security Considerations

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

The NETCONF access control model [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

All data nodes defined in the YANG module which can be created, modified and deleted (i.e., config true, which is the default) are considered sensitive as they all could potentially impact security monitoring and mitigation activities. Write operations (e.g., edit-config) applied to these data nodes without proper protection could result in missed alarms or incorrect alarms information being returned to the NSF data collector. There are threats that need to be considered and mitigated:

Compromised NSF with valid credentials: It can send falsified information to the NSF data collector to mislead detection or mitigation activities; and/or to hide activity. Currently, there is no in-framework mechanism to mitigate this and an issue for all monitoring infrastructures. It is important to keep the enclosure of confidential information to unauthorized persons to mitigate the possibility of compromising the NSF with this information.

Compromised NSF data collector with valid credentials: It has visibility to all collected security alarms; entire detection and mitigation infrastructure may be suspect. It is important to keep the enclosure of confidential information to unauthorized persons

to mitigate the possibility of compromising the NSF with this information.

Impersonating NSF: It is a system trying to send false information while imitating an NSF; client authentication would help the NSF data collector to identify this invalid NSF in the "push" model (NSF-to-collector), while the "pull" model (collector-to-NSF) should already be addressed with the authentication.

Impersonating NSF data collector: It is a rogue NSF data collector with which a legitimate NSF is tricked into communicating; for "push" model (NSF-to-collector), it is important to have valid credentials, without it it should not work; for "pull" model (collector-to-NSF), mutual authentication should be used to mitigate the threat.

In addition, to defend against the DDoS attack caused by a lot of NSFs sending massive notifications to the NSF data collector, the rate limiting or similar mechanisms should be considered in both an NSF and NSF data collector, whether in advance or just in the process of DDoS attack.

All of the readable data nodes in this YANG module may be considered vulnerable in some network environments. Some data also may contain private information that is highly sensitive to the user, such as the IP address of a user in the container "i2nsf-system-user-activity-log" and the container "i2nsf-system-detection-event". It is important to control read access (e.g., via get, get-config, or notification) to the data nodes. If access control is not properly configured, it can expose system internals to those who should not have access to this information.

14. Acknowledgments

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea MSIT (Ministry of Science and ICT) (R-20160222-002755, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning). This work was supported in part by the IITP (2020-0-00395, Standard Development of Blockchain based Network Management Automation Technology). This work was supported in part by the MSIT under the Information Technology Research Center (ITRC) support program (IITP-2021-2017-0-01633) supervised by the IITP.

15. Contributors

This document is made by the group effort of I2NSF working group. Many people actively contributed to this document. The authors sincerely appreciate their contributions.

The following are co-authors of this document:

Chaehong Chung Department of Electronic, Electrical and Computer Engineering Sungkyunkwan University 2066 Seo-ro Jangan-gu Suwon, Gyeonggi-do 16419 Republic of Korea EMail: darkhong@skku.edu

Jinyong (Tim) Kim Department of Electronic, Electrical and Computer Engineering Sungkyunkwan University 2066 Seo-ro Jangan-gu Suwon, Gyeonggi-do 16419 Republic of Korea EMail: timkim@skku.edu

Dongjin Hong Department of Electronic, Electrical and Computer Engineering Sungkyunkwan University 2066 Seo-ro Jangan-gu Suwon, Gyeonggi-do 16419 Republic of Korea EMail: dong.jin@skku.edu

Dacheng Zhang Huawei EMail: dacheng.zhang@huawei.com

Yi Wu Aliababa Group EMail: anren.wy@alibaba-inc.com

Rakesh Kumar Juniper Networks 1133 Innovation Way Sunnyvale, CA 94089 USA EMail: rkkumar@juniper.net

Anil Lohiya Juniper Networks EMail: alohiya@juniper.net

16. References

16.1. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <https://www.rfc-editor.org/info/rfc768>.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <https://www.rfc-editor.org/info/rfc792>.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <https://www.rfc-editor.org/info/rfc793>.

[RFC0959] Postel, J. and J. Reynolds, "File Transfer Protocol", STD 9, RFC 959, DOI 10.17487/RFC0959, October 1985, <https://www.rfc-editor.org/info/rfc959>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/

RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

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

[RFC3877] Chisholm, S. and D. Romascanu, "Alarm Management Information Base (MIB)", RFC 3877, DOI 10.17487/RFC3877, September 2004, <https://www.rfc-editor.org/info/rfc3877>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <https://www.rfc-editor.org/info/rfc4443>.

[RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, DOI 10.17487/RFC5277, July 2008, <https://www.rfc-editor.org/info/rfc5277>.

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

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

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/

RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", RFC 8525, DOI 10.17487/RFC8525, March 2019, <https://www.rfc-editor.org/info/rfc8525>.

[RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <https://www.rfc-editor.org/info/rfc8639>.

[RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <https://www.rfc-editor.org/info/rfc8641>.

16.2. Informative References

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <https://www.rfc-editor.org/info/rfc4949>.

[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>.

[I-D.ietf-i2nsf-consumer-facing-interface-dm] Jeong, J. (., Chung, C., Ahn, T., Kumar, R., and S. Hares, "I2NSF Consumer-Facing Interface YANG Data Model", Work in Progress, Internet-Draft, draft-ietf-i2nsf-consumer-facing-interface-dm-13, 8 March 2021, <https://www.ietf.org/archive/id/draft-ietf-i2nsf-consumer-facing-interface-dm-13.txt>.

[I-D.ietf-i2nsf-nsf-facing-interface-dm] Kim, J. (., Jeong, J. (., Park, J., Hares, S., and Q. Lin, "I2NSF Network Security Function-Facing Interface YANG Data Model", Work in Progress, Internet-Draft, draft-ietf-i2nsf-nsf-facing-interface-dm-12, 8 March 2021, <https://www.ietf.org/archive/id/draft-ietf-i2nsf-nsf-facing-interface-dm-12.txt>.

[I-D.ietf-i2nsf-registration-interface-dm] Hyun, S., Jeong, J. P., Roh, T., Wi, S., and J. Park, "I2NSF Registration Interface YANG Data Model", Work in Progress, Internet-Draft, draft-ietf-i2nsf-registration-interface-dm-10, 21 February 2021, <https://www.ietf.org/archive/id/draft-ietf-i2nsf-registration-interface-dm-10.txt>.

[I-D.ietf-i2nsf-applicability] Jeong, J. P., Hyun, S., Ahn, T., Hares, S., and D. R. Lopez, "Applicability of Interfaces to Network Security Functions to Network-Based Security Services", Work in Progress, Internet-Draft, draft-ietf-i2nsf-applicability-18, 16 September 2019, <https://www.ietf.org/archive/id/draft-ietf-i2nsf-applicability-18.txt>.

[I-D.yang-i2nsf-security-policy-translation] Jeong, J. (., Lingga, P., Yang, J., and C. Chung, "Security Policy Translation in Interface to Network Security Functions", Work in Progress, Internet-Draft, draft-yang-i2nsf-security-policy-translation-08, 22 February 2021, <https://www.ietf.org/archive/id/draft-yang-i2nsf-security-policy-translation-08.txt>.

[IANA-HTTP-Status-Code] Internet Assigned Numbers Authority (IANA), "Hypertext Transfer Protocol (HTTP) Status Code

Registry", September 2018, <https://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml>.

[IANA-Media-Types] Internet Assigned Numbers Authority (IANA), "Media Types", August 2021, <https://www.iana.org/assignments/media-types/media-types.xhtml>.

Appendix A. Changes from draft-ietf-i2nsf-nsf-monitoring-data-model-08

The following changes are made from draft-ietf-i2nsf-nsf-monitoring-data-model-08:

* This version is revised following Tom Petch's, Martin Bjorklund's, and Roman Danyliw's Comments.

* This version is revised to synchronize with other I2NSF documents.

Authors' Addresses

Jaehoon (Paul) Jeong (editor)
Department of Computer Science and Engineering
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon
Gyeonggi-Do
16419
Republic of Korea

Phone: +82 31 299 4957
Email: pauljeong@skku.edu
URI: http://iotlab.skku.edu/people-jaehoon-jeong.php

Patrick Lingga
Department of Electrical and Computer Engineering
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon
Gyeonggi-Do
16419
Republic of Korea

Phone: +82 31 299 4957
Email: patricklink@skku.edu

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
United States of America

Phone: +1-734-604-0332
Email: shares@ndzh.com

Liang (Frank) Xia
Huawei
101 Software Avenue, Yuhuatai District
Nanjing
Jiangsu,
China

Email: Frank.xialiang@huawei.com

Henk Birkholz
Fraunhofer Institute for Secure Information Technology
Rheinstrasse 75
64295 Darmstadt
Germany

Email: henk.birkholz@sit.fraunhofer.de