

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: May 4, 2017

H. Kim  
H. Ko  
D. Daghmehchi  
J. Jeong  
Sungkyunkwan University  
T. Ahn  
Korea Telecom  
October 31, 2016

**I2NSF Data Model of Consumer-Facing Interface for Security Management  
draft-kim-i2nsf-consumer-facing-interface-dm-00**

**Abstract**

This document describes a data model for security management that is based on Interface to Network Security Functions (I2NSF) by using Network Functions Virtualization (NFV). This document proposes a security management architecture based on I2NSF framework. Note that the I2NSF framework consists of I2NSF User, Security Management System (i.e., Security Controller and Developer's Management System), and NSF instances in the lowest layer of the framework. I2NSF User consists of Application Logic, Policy Updater, and Event Collector. Security Controller consists of Security Policy Manager and NSF Capability Manager. This document explains a data model to perform the missions for a security service (i.e., VoIP-VoLTE) in I2NSF security management system.

**Status of This Memo**

This Internet-Draft is submitted to IETF in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/1id-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on May 4, 2017.

## Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">3</a>
<a href="#">2.</a>	Objectives . . . . .	<a href="#">4</a>
<a href="#">3.</a>	Requirements Language . . . . .	<a href="#">4</a>
<a href="#">4.</a>	Terminology . . . . .	<a href="#">4</a>
<a href="#">5.</a>	Architecture of Security Management . . . . .	<a href="#">5</a>
<a href="#">5.1.</a>	I2NSF User . . . . .	<a href="#">6</a>
<a href="#">5.2.</a>	Security Management System . . . . .	<a href="#">7</a>
<a href="#">5.3.</a>	NSF Instances . . . . .	<a href="#">7</a>
<a href="#">6.</a>	Use Case: VoIP-VoLTE Security Service . . . . .	<a href="#">7</a>
<a href="#">6.1.</a>	Security Management for VoIP-VoLTE Security Service . . . . .	<a href="#">8</a>
<a href="#">6.2.</a>	Data Model for VoIP-VoLTE Security Service . . . . .	<a href="#">8</a>
<a href="#">7.</a>	Security Considerations . . . . .	<a href="#">10</a>
<a href="#">8.</a>	Acknowledgements . . . . .	<a href="#">10</a>
<a href="#">9.</a>	References . . . . .	<a href="#">11</a>
<a href="#">9.1.</a>	Normative References . . . . .	<a href="#">11</a>
<a href="#">9.2.</a>	Informative References . . . . .	<a href="#">11</a>



## **1. Introduction**

Basically, information model and data model are used to defining the managed objects in the network management. Despite of some overlapped details, they have different characters in the view of network management. Generally, the main purpose of information model is to model managed objects at a conceptual level, with no dependent of any specific implementations or protocols. To make a clear overall design, the information model should hide all protocol and implementation details defining relationships between managed objects. Based on this, the information models can be implemented in different ways and mapped on different protocols. They are neutral to protocols. In general, information models can be defined in an informal way, using natural languages such as English. Furthermore, it seems advisable to use object-oriented techniques to describe an information model.

Data models are defined at a lower level of abstraction and provide many details. They provide details about the implementation and protocols' specification, e.g., rules that explain how to map managed objects onto lower-level protocol constructs. Since conceptual models can be implemented in different ways, multiple data models can be derived by a single information model.

The impressive role of the network functions virtualization (NFV) in the network management leads to a rapid advent of NFV in this industry. As practical applications, network security functions (NSFs), such as firewall, intrusion detection system (IDS) and intrusion protection system (IPS), can also be provided as virtual network functions (VNF). By virtual technology, these VNFs might be automatically provisioned and dynamically migrated based on real-time security requirements. This document presents an information model to implement security functions based on NFV.

This document proposes a data modeling in an architecture for security management [[i2nsf-security-management](#)], which is based on I2NSF framework [[i2nsf-framework](#)]. This I2NSF framework contains I2NSF User, Security Management System (i.e., Security Controller and Developer's Management System), and NSFs in the NSF instance layer. The security management architecture has more detailed structures for core components in the I2NSF framework. I2NSF User includes Application Logic, Policy Updater, and Event Collector. Security Controller contains Security Policy Manager and NSF Capability Manager.

Application Logic generates a high-level policy and Policy Updater sends it to Security Policy Manager via Consumer-Facing Interface. Security Policy Manager maps the high-level policy into several low-



level policies in Security Controller. After mapping, the low-level policies are distributed to NSF(s) via NSF-Facing Interface so that they can be enforced in them. When an event occurs for NSF to change a low-level policy, NSF sends the event to Security Controller via NSF-Facing Interface. Security Controller then forwards it to Event Collector via Consumer-Facing Interface. Next, Event Collector sends it to Application Logic. Application Logic then updates the current policies in accordance with the event.

This document proposes a data model for security services in the security management architecture in [[i2nsf-security-management](#)] so that the security management architecture can support flexible and effective security policies.

## **2. Objectives**

The two main objectives for security management architecture in this document are as follows.

- o High-level security management: To propose the design of a generic security management architecture to support the enforcement of flexible and effective security policies in NSFs.
- o Automatic update of security policies: To provide the reflection of the updated low-level security policies for new security attacks on the corresponding high-level security policies.

## **3. Requirements Language**

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#) [[RFC3444](#)].

## **4. Terminology**

This document uses the terminology described in [[i2nsf-framework](#)][[i2nsf-security-management](#)]. In addition, the following terms are defined below:

- o Application Logic: It is a component in the security management architecture which generates high-level security policies to block or mitigate security attacks.
- o Policy Updater: It is a component which forwards a high-level security policy to Security Controller. The high-level policy is received from Application Logic.



- o Security Policy Manager: It maps a high-level security policy received from Policy Updater into low-level security policies, and vice versa.
- o NSF Capability Manager: It is a component which stores the NSF capability registered by Developer's Management System via Registration Interface and shares it with Security Policy Manager to generate the corresponding low-level security policies.
- o Event Collector: It is a component which receives an event from Security Controller, which should be reflected by updating (or generating) a high-level policy in Application Logic.

## **5. Architecture of Security Management**

Generally, Data models are often represented in formal data definition languages that are specific to the management protocol being used. Based on NFV, the structure of the proposed model is based on VNFs to provide a flexible and effective security policies. Figure 1 illustrates the structure of the suggested model. The architecture is designed based on three layers: I2NSF user, security management system, and NSF instances. The high level security policies are defined and distributed in the I2NSF user layer. Translating the high level security policies relevant to NSF capability and delivering them to NSF interfaces are performed in the security management system.





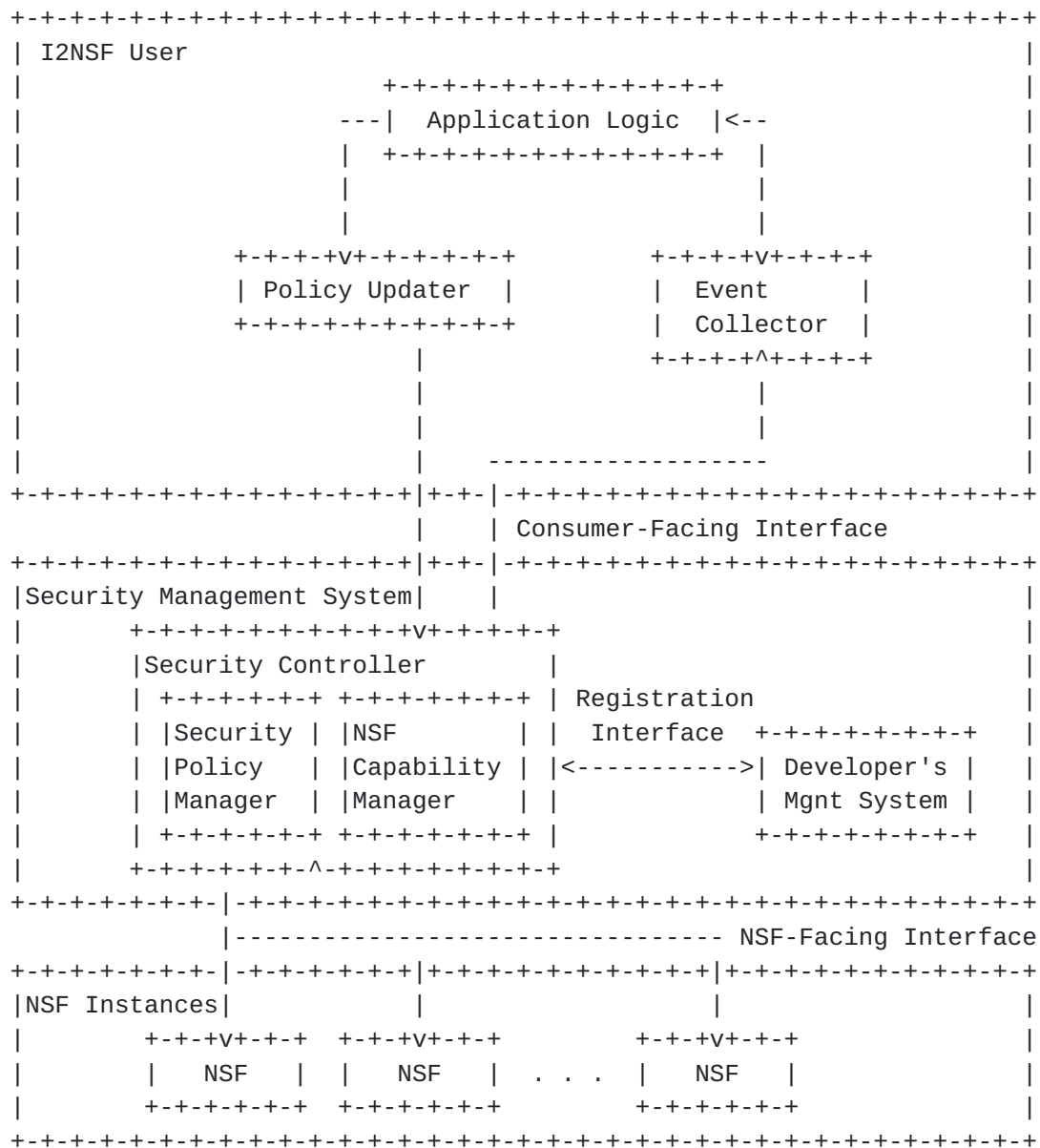


Figure 1: Security Management Architecture in I2NSF

### 5.1. I2NSF User

I2NSF User layer consists of three components; Application logic, Policy Updater, and Event Collector. The Application logic is a component which generates high level security policies. To this end, it receives the event for updating (or generating) a high level policy from Event collector and updates (or generates) a high level policy based on the collected events. After that, the high level policy is sent to Policy updater in order to distribute it to Security controller(s). In order to update (or generate) a high level policy, Event collector receives the events sent by Security



controller and sends them to Application logic. Based on these feedbacks, the Application logic can update (or generate) high level security policies.

### **5.2. Security Management System**

In the security management system layer, the Security policy manager receives a high level policy from Policy updater via Consumer-Facing Interface and maps this policy into several low level policies. These low level policies are relevant to a given NSF capability that is registered into NSF capability manager. Moreover, Security policy manager delivers those policies to NSF(s) via NSF-Facing Interface.

To generate low level policies relevant to a given NSF capability, the NSF capability manager stores an NSF's capability registered by Developer's management system and shares it with Security policy manager. Whenever a new NSF is registered, NSF capability manager requests Developer's management system to register the NSF's capability into the management table of NSF capability manager via Registration Interface. Developer's management system is another part of security management system to registers a new NSF's capability into NSF capability manager.

### **5.3. NSF Instances**

All NSFs are located at this layer. After mapping the high level policies to low level policies, the Security policy manager delivers those policies to NSF(s) through NSF-Facing Interface.

## **6. Use Case: VoIP-VoLTE Security Service**

As a use case for implementation, VoIP-VoLTE security management is considered to develop a data model. Based on this, the VoIP-VoLTE security manager acts as Application logic for VoIP-VoLTE security services and defines the security conditions. Based on VoIP-VoLTE security management, the list of illegal devices information is stored in VoIP-VoLTE database and can be updated either manually or automatically by VoIP-VoLTE security manager. To define the policies, information of dangerous domain blacklists (e.g., IP addresses and source ports), time management (e.g., access time and expire time), user-agent (e.g., priority levels), and Session Initiation Protocol (SIP) URIs of an SIP device that are suspicious of illegal call or authentication is published by VoIP-VoLTE security manager. Accordingly, the list of illegal devices, which is automatically (or manually) updated, is stored in VoIP-VoLTE database. The VoIP-VoLTE security manager periodically loads this list to generate a new high level security policy (e.g., the blocking list of illegal devices using IP address, source ports, etc) to



prevent the delivery of packets from/to those newly added VoIP-VoLTE attackers.

When the NSF detects an anomalous message or call delivered from a domain, the information of the domain such as an IP address, user-agents and expire time values is sent by an NSF to Security controller via NSF-Facing Interface. Security controller delivers it to Event collector. Event collector forwards the detected domain information to VoIP-VoLTE security manager, and then VoIP-VoLTE security manager updates the VoIP-VoLTE database.

### **6.1. Security Management for VoIP-VoLTE Security Service**

VoIP-VoLTE security management maintains and publishes the blacklists of IP addresses, source ports, expire time, user-agents, and Session Initiation Protocol (SIP) URIs of SIP device that are suspicious of illegal call and authentication. In our generic security management architecture, VoIP-VoLTE Security Manager is plays the role of Application Logic for VoIP-VoLTE security services in Figure 1.

Based on VoIP-VoLTE security management, the list of illegal devices information can be updated either manually or automatically by VoIP-VoLTE Security Manager as Application Logic. Also, VoIP-VoLTE Security Manager periodically generates a new high-level security policy to prevent the delivery of packets from/to those newly added VoIP-VoLTE attackers and enforce the low-level security policies in NSF. It sends the new high-level security policy to Policy Updater, which forwards it to Security Controller.

When the NSF detects an anomalous message or call delivered from a domain, the domain information such as an IP address, user-agents and expire time values is sent by an NSF to Security Controller via NSF Facing Interface. Security Controller delivers it to Event Collector. Event Collector forwards the detected domain information to VoIP-VoLTE Security Manager, and then VoIP-VoLTE Security Manager updates the VoIP-VoLTE database.

### **6.2. Data Model for VoIP-VoLTE Security Service**

To implement the model, three parameters have been considered to define the high level policies; blacklisting countries, time interval specification, and caller's priority levels. If the administrator sets a new high-level security policy, a data model parser in I2NSF User interprets the policy and generates an XML file in accordance with YANG data model. In order to enable interaction between I2NSF User and Security management system, a communication channel based on RESTCONF is implemented. Basically, the data model is defined based on the security policy requirements to detect the suspicious calls in



VoIP-VoLTE services. Figure 2 shows a part of this data model.

```

+--: (policy)
  +--rw policy-lifecycle *(policy-lifecycle-id)
  |   +--rw expiration-event
  |   |   +--rw enabled boolean
  |   |   +--rw event-id uint 16
  |   +--rw expiration-time
  |       +--rw enabled boolean
  |       +--rw time data-and-time
  +--rw policy-rule *[policy-rule-id]
  |   +--rw policy-name string
  |   +--rw policy-rule-id uint 16
  |   +--rw service
  |   |   +--voip-handling boolean
  |   |   +--volet-handling boolean
  |   +--rw condition *[condition-id]
  |       +--rw caller
  |       |   +--rw caller-id uint 16
  |       |   +--rw caller-location
  |       |       +--rw country string
  |       |       +--rw city string
  |       +--rw callee
  |       |   +--rw callee-id uint 16
  |       |   +--rw callee-location
  |       |       +--rw country string
  |       |       +--rw city string
  |       +--rw valid-time-interval
  |           +--rw start-time data-and-time
  |           +--rw end-time data-and-time
  +--rw action
      +--rw (action-type)?
      +--: (ingress-action)
      |   +--rw permit? boolean
      |   +--rw mirror? boolean
      |   +--rw log? boolean
      +--: (egress-type)
          +--rw redirection? boolean

```

Figure 2: Data Model for VoIP-VoLTE Security Service

The data model consists of policy life cycle management, policy rule, and action. The policy life cycle field specifies an expiration time and/or a set of expiration events to determine the life-time of the policy itself. The policy rule field represents the specific information about a high-level policy such as service types, conditions and valid time interval. The action field specifies which actions should be taken. For example, call traffic from a





blacklisted caller location at an unusual time of day (included in the valid-time-interval) could be blocked and sequentially forwarded to a pre-defined host for Deep Packet Inspection (DPI) when both permit and mirror are assigned true.

To translate a high level policy into a set of low level policies, the security management system is implemented. After translating the high-level security policy, Security management system generates low-level security policies to specify the actions network traffic from and/or to those IP addresses. The data model parser generates an XML \_le for a low-level security policy and delivers it to proper NSF instances. Security management system also interprets security events generated by NSF into a high-level log message in a YANG data model and delivers it to I2NSF Users in the opposite direction.

In this case, we select a firewall application as an NSF instance to determine whether a VoIP-VoLTE call is suspicious or not by checking the caller's and callee's locations and call time. When a call has suspicious behavior patterns, its network traffic could be effectively blocked by the firewall application according to the low-level security policy. The results for the firewall application would be delivered in a YANG data model to the Security management system through the RESTCONF protocol. Multiple NSF instances can be considered depending on specific situations. For example, additionally DPI can be used for analyzing the network traffic from suspicious callers.

## **7. Security Considerations**

The security management architecture is derived from the I2NSF framework [[i2nsf-framework](#)], so the security considerations of the I2NSF framework should be included in this document. Especially, proper secure communication channels should be used for the delivery of control or management messages amongst the components in the proposed architecture.

## **8. Acknowledgements**

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No.R-20160222-002755, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning). This document has greatly benefited from inputs by Sanghak Oh, Eunsoo Kim, Soyoung Kim, and Se-Hui Lee.

## **9. References**



### **9.1. Normative References**

- [RFC3444] Pras, A., "On the Difference between Information Models and Data Models", [RFC 3444](#), January 2003.

### **9.2. Informative References**

- [i2nsf-framework] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", [draft-ietf-i2nsf-framework-04](#) (work in progress), October 2016.
- [i2nsf-security-management] Kim, H., Ko, H., Oh, S., Jeong, J., and S. Lee, "An Architecture for Security Management in I2NSF Framework", [draft-kim-i2nsf-security-management-architecture-03](#) (work in progress), October 2016.

### **Authors' Addresses**

Hyoungshick Kim  
Department of Software  
Sungkyunkwan University  
2066 Seobu-Ro, Jangan-Gu  
Suwon, Gyeonggi-Do 16419  
Republic of Korea

Phone: +82 31 299 4324  
Fax: +82 31 290 7996  
EMail: [hyoung@skku.edu](mailto:hyoung@skku.edu)  
URI: <http://seclab.skku.edu/people/hyoungshick-kim/>

Hoon Ko  
Department of Computer Science and Engineering  
Sungkyunkwan University  
2066 Seobu-Ro, Jangan-Gu  
Suwon, Gyeonggi-Do 16419  
Republic of Korea

Phone: +82-31-299-4104  
EMail: [skoh21@skku.edu](mailto:skoh21@skku.edu)



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

Phone: +82-31-299-4104  
EMail: mdaghmechi@skku.edu

Jaehoon Paul Jeong  
Department of Software  
Sungkyunkwan University  
2066 Seobu-Ro, Jangan-Gu  
Suwon, Gyeonggi-Do 16419  
Republic of Korea

Phone: +82 31 299 4957  
Fax: +82 31 290 7996  
EMail: pauljeong@skku.edu  
URI: <http://iotlab.skku.edu/people-jaehoon-jeong.php>

Tae-Jin Ahn  
Korea Telecom  
70 Yuseong-Ro, Yuseong-Gu  
Daejeon 305-811  
Republic of Korea

Phone: +82 42 870 8409  
EMail: taejin.ahn@kt.com

