Network Working Group
Internet Draft
Intended status: Informational
Expires: November 2016

E. Lopez
Fortinet
D. Lopez
Telefonica
L. Dunbar
J. Strassner
Huawei
X. Zhuang
China Mobile
J. Parrott
BT
R Krishnan
Dell
S. Durbha
CableLabs

May 2, 2016

Framework for Interface to Network Security Functions
draft-ietf-i2nsf-framework-00.txt

Status of this Memo

This Internet-Draft will expire on September 2, 2016.

Copyright Notice

Abstract

This document defines the framework for guiding the functionality provided by I2NSF. Network security functions (NSFs) are packet-processing engines that inspect and optionally modify packets traversing networks, either directly or in the context of sessions in which the packet is associated. This document provides an overview of how NSFs are used, and describes how NSF software interfaces are controlled and monitored using rulesets. The design of these software interfaces must prevent the creation of implied constraints on NSF capability and functionality.

Table of Contents

1. Introduction

This document describes the framework for the Interface to Network Security Functions (I2NSF), and defines a reference model (including major functional components) for I2NSF. It also describes how I2NSF facilitates Software-Defined Networking (SDN) and Network Function Virtualization (NVF) control, while avoiding potential constraints that could limit the internal functionality and capabilities of NSFs.

The I2NSF use cases ([I2NSF-ACCESS], [I2NSF-DC] and [I2NSF-Mobile]) call for standard interfaces for clients (e.g., applications, application controllers, or users), to inform the network what they are willing to receive. I2NSF realizes this as a set of security rules for monitoring and controlling the behavior of their specific traffic. It also provides standard interfaces for them to monitor the security functions hosted and managed by service providers.

[I2NSF-Problem] describes the motivation and the problem space for Interface to Network Security Functions.

2. Conventions used in this document

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 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

BSS: Business Support System

Controller: used interchangeably with Service Provider Security Controller or management system throughout this document.

FW: Firewall

IDS: Intrusion Detection System

IPS: Intrusion Protection System

NSF: Network Security Functions, defined by [I2NSF-Problem]

OSS: Operation Support System

vNSF: refers to NSF being instantiated on Virtual Machines.

3. Interfaces to Flow-based NSFs

The emergence of SDN and NFV have resulted in the need to create application programming interfaces (APIs) in support of dynamic requests from various applications or application controllers.

Flow-based NSFs [I2NSF-Problem] inspects packets in the order that they are received. The Interface to Flow-based NSFs can be generally grouped into three types:

1) Configuration - deals with the management and configuration of the NSF device itself, such as port address configurations. Configuration deals with attributes that are relatively static.

2) Signaling - which represents logging and query functions between the NSF and external systems. Signaling API functions may also be defined by other protocols, such as SYSLOG and DOTS.

3) Rules Provisioning - used to control the rules that govern how packets are treated by the NSFs. Due to the need of applications/controllers to dynamically control what traffic they need to receive, much of the I2NSF efforts towards interface development will be in this area.

This draft proposes that a rule provisioning interface to NSFs can be developed on a packet- or flow-based paradigm. A common trait of NSFs is in the processing of packets based on the content (header/payload) and/or context (session state, authentication state, etc) of the received packets.

An important concept underlying this framework is the fact that attackers do not have standards as to how to attack networks, so it is equally important not to constrain NSF developers to offering a limited set of security functions. In other words, the introduction of I2NSF standards should not make it easier for attackers to compromise the network. Therefore, in constructing standards for rules provisioning interfaces to NSFs, it is equally important to allow support for vendor-specific functions, as this enables the introduction of NSFs that evolve to meet new threats. Proposed standards for rules provisioning interfaces to NSFs SHOULD NOT:

- Narrowly define NSF categories, or their roles when implemented within a network

- Attempt to impose functional requirements or constraints, either directly or indirectly, upon NSF developers

- Be a limited lowest common denominator approach, where interfaces can only support a limited set of standardized functions, without allowing for vendor-specific functions

- Be seen as endorsing a best common practice for the implementation of NSFs

By using a packet/flow-based approach to the design of such provisioning interfaces, the goal is to create a workable interface to NSFs that aids in their integration within legacy, SDN, and/or NFV environments, while avoiding potential constraints which could limit their functional capabilities.

Even though security functions come in a variety of form factors and have different features, provisioning to flow-based NSFs can be standardized by using Event - Condition - Action (ECA) policy rulesets.

An Event, when used in the context of policy rules for a flow-based NSF, is used to determine whether the condition clause of the Policy Rule can be evaluated or not. Here are some examples of I2NSF Events:

- defining a clause, of the canonical form {variable, operator, value}, to represent an Event (e.g., time == 08:00)
- using an Event object as the variable or the value in the above clause (e.g., use one or more attributes from one or more Event objects in the comparison clause)
- using a Collection object to collect Events for aggregation, filtering, and/or correlation operations as part of the Event clause processing
- encoding the entire Event expression into an attribute

A Condition, when used in the context of policy rules for flow-based NSFs, is used to determine whether or not the set of Actions in that Policy Rule can be executed or not. A condition can be based on various combinations of the content (header/payload) and/or the context (session state, authentication state, etc) of the received packets:

- Packet content values are based on one or more packet headers, data from the packet payload, bits in the packet, or something derived from the packet;
- Context values are based on measured and inferred knowledge that define the state and environment in which a managed entity exists or has existed. In addition to state data, this includes data from sessions, direction of the traffic, time, and geo-location information. State refers to the behavior of a managed entity at a particular point in time. Hence, it may refer to situations in which multiple pieces of information that are not available at the same time must be analyzed. For example, tracking established TCP connections (connections that have gone through the initial three-way handshake).

Actions for flow-based NSFs include:

- Action ingress processing, such as pass, drop, mirroring, etc;
- Action egress processing, such as invoke signaling, tunnel encapsulation, packet forwarding and/or transformation;
- Applying a specific Functional Profile or signature - e.g., an IPS Profile, a signature file, an anti-virus file, or a URL filtering file. Many flow-based NSFs utilize profile and/or signature files to achieve more effective threat detection and prevention. It is not uncommon for a NSF to apply different profiles and/or signatures for different flows. Some profiles/signatures do not require any knowledge of past or future activities, while others are stateful, and may need to maintain state for a specific length of time.

The functional profile or signature file is one of the key properties that determine the effectiveness of the NSF, and is mostly vendor-specific today. The rulesets and software interfaces of I2NSF aim to standardize the form and function of profile and signature files while supporting vendor-specific functions of each.

4. Reference Models in Managing Flow-based NSFs

This document only focuses on the framework of rules provisioning for and monitoring of flow-based NSFs.

The following figure shows various interfaces for managing the provisioning & monitoring aspects of flow-based NSFs.

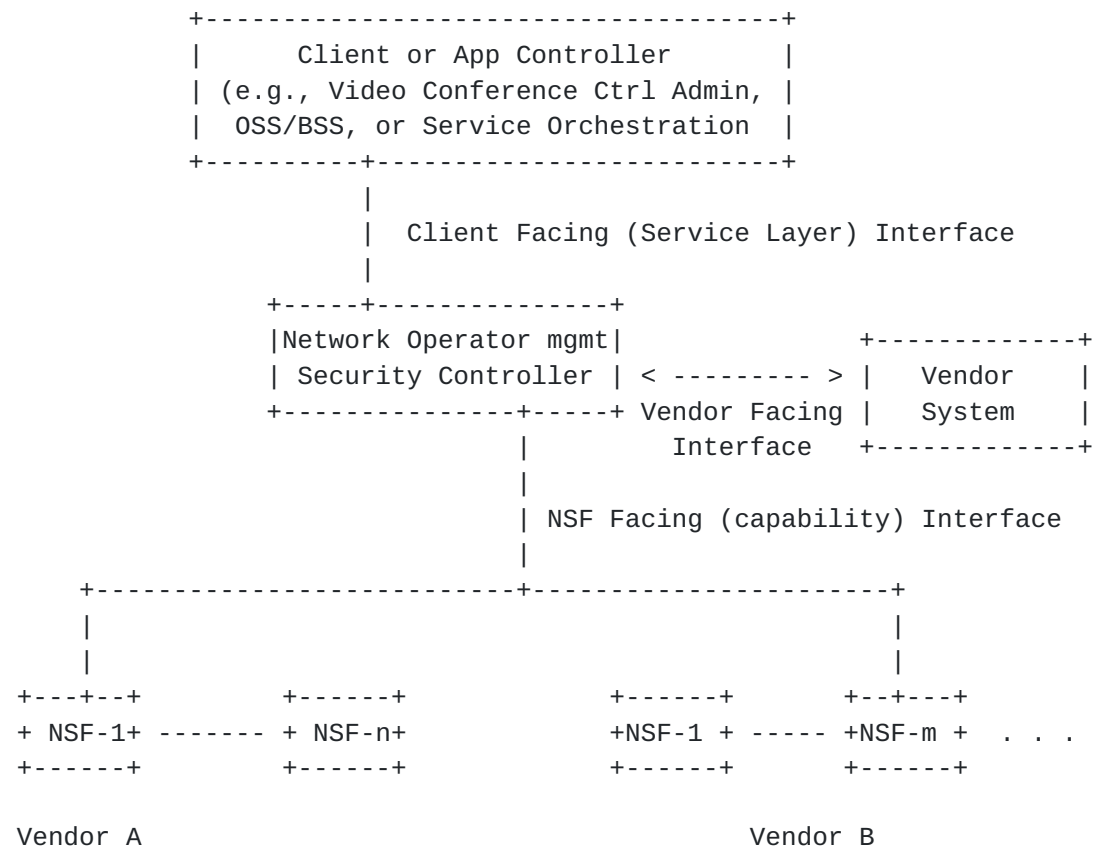

Figure 1: Multiple Interfaces

4.1. NSF Facing (Capability Layer) Interface

This is the interface between the Service Provider's management system (or Security Controller) and the set of NSFs that are selected to enforce the desired network security. This interface defines the features available for each NSF that the management system can choose to invoke for a particular packet or flow. Note that the management system does not need to use all features for a given NSF, nor does it need to use all available NSFs. Hence, this abstraction enables the same relative features from diverse NSFs from different vendors to be selected.

This interface is called the Capability Interface in the I2NSF context.

4.2. Client Facing (Service Layer) Interface

This interface is for clients or Application Controller to express and monitor security policies for their specific flows. The Client Facing interface is called the Service Layer Interface in the I2NSF context. The I2NSF Service Layer allows the client to define and monitor the client specific policies and their execution status.

A single client layer policy may need multiple NSFs (or multiple instantiations of the same NSF) to achieve the desired enforcement.

4.3. Vendor Facing Interface

NSFs provided by different vendors have different capabilities. In order to automate the process of utilizing multiple types of security functions provided by different vendors, it is necessary to have an interface for vendors to register their NSFs indicating the capabilities of their NSFs.

The Registration Interface can be defined statically or instantiated dynamically at runtime. If a new functionality that is exposed to the user is added to an NSF, the vendor MUST notify the network operator's management system or security controller of its updated functionality via the Registration Interface.

4.4. The Network Connecting the Security Controller and NSFs

Most likely the NSFs are not directly attached to the Security Controller; for example, NSFs can be distributed across the network. The network that connects the Security Controller with the NSFs can be the same network that carries the data traffic, or can be a dedicated network for management purposes only. In either case, packet loss could happen due to failure, congestion, or other reasons.

Therefore, the transport mechanism used to carry the control messages and monitoring information should provide reliable message delivery. Transport redundancy mechanisms such as Multipath TCP (MPTCP) [MPTCP] and the Stream Control Transmission Protocol (SCTP) [RFC3286] will need to be evaluated for applicability. Latency requirements for control message delivery must also be evaluated.

The network connection between the Security Controller and NSFs could be:

- Closed environments, where there is only one administrative domain. Less restrictive access control and simpler validation can be used inside the domain because of the protected environment.
- Open environments, where some NSFs (virtual or physical) can be hosted in external administrative domains or reached via secure external network domains. This requires more restrictive security control to be placed over the I2NSF interface. Not only must the information over the I2NSF interfaces use trusted channels, such as TLS, SASL (RFC4422), or the combination of the two, but also require proper authentication as described in [Remote-Attestation].

  Over the Open Environment, I2NSF needs to provide identity information, along with additional data that Authentication, Authorization, and Accounting (AAA) frameworks can use. This enables those frameworks to perform AAA functions on the I2NSF traffic.

4.5. Interface to vNSFs

Even though there is no difference between virtual network security functions (vNSF) and physical NSFs from the policy provisioning perspective, there are some unique characteristics in interfacing to the vNSFs:

- There could be multiple instantiations of one single NSF that has been distributed across a network. When different instantiations are visible to the Security Controller, different

policies may be applied to different instantiations of an individual NSF (e.g., to reflect the different roles that each vNSF is designated for).

- When multiple instantiations of one single NSF appear as one single entity to the Security Controller, the policy provisioning has to be sent to the NSF's sub-controller, which in turn disseminates the polices to the corresponding instantiations of the NSF, as shown in the Figure 2 below.
- Policies to one vNSF may need to be retrieved and moved to another vNSF of the same type when client flows are moved from one vNSF to another.
- Multiple vNSFs may share the same physical platform
- There may be scenarios where multiple vNSFs collectively perform the security policies needed.

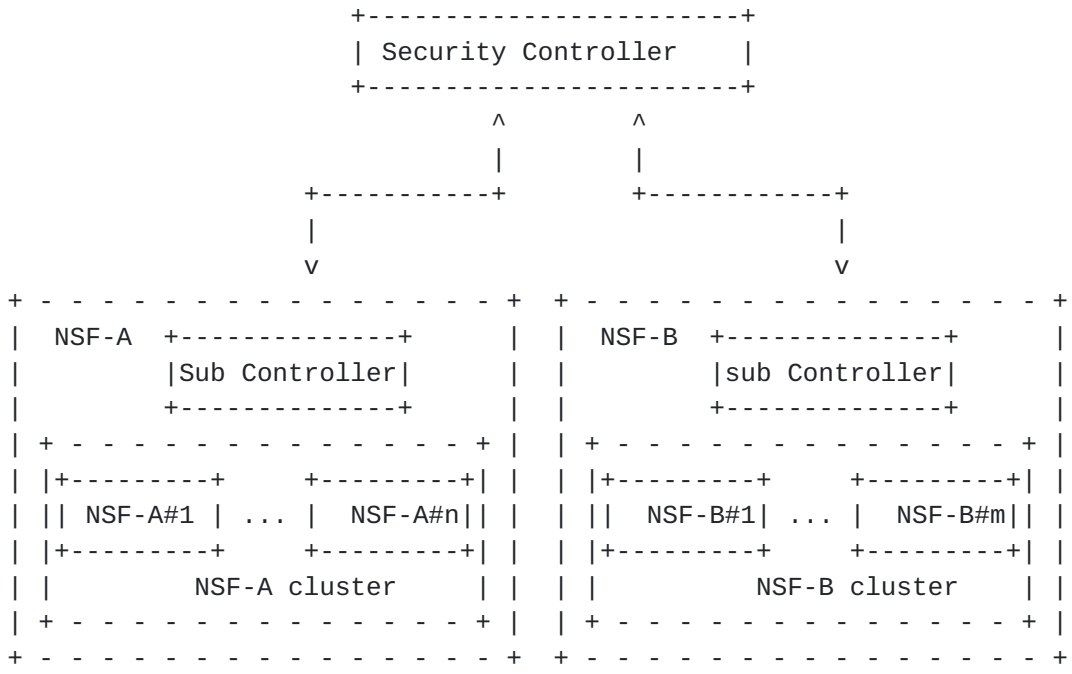

Figure 2: Cluster of NSF Instantiations Management

5. Flow-based NSF Capability Characterization

There are many types of flow-based NSFs. Firewall, IPS, and IDS are the commonly deployed flow-based NSFs. However, the differences

among them are definitely blurring, due to technological capacity increases, integration of platforms, and new threats. At their core:

. Firewall - A device or a function that analyzes packet headers and enforces policy based on protocol type, source address, destination address, source port, destination port, and/or other attributes of the packet header. Packets that do not match policy are rejected. Note that additional functions, such as logging and notification of a system administrator, could optionally be enforced as well.
. IDS (Intrusion Detection System) - A device or function that analyzes packets, both header and payload, looking for known events. When a known event is detected, a log message is generated detailing the event. Note that additional functions, such as notification of a system administrator, could optionally be enforced as well.
. IPS (Intrusion Prevention System) - A device or function that analyzes packets, both header and payload, looking for known events. When a known event is detected, the packet is rejected. Note that additional functions, such as logging and notification of a system administrator, could optionally be enforced as well.

To prevent constraints on NSF vendors' creativity and innovation, this document recommends the Flow-based NSF interfaces to be designed from the paradigm of processing packets in the network. Flow-based NSFs ultimately are packet-processing engines that inspect packets traversing networks, either directly or in the context of sessions in which the packet is associated.

Flow-based NSFs differ in the depth of packet header or payload they can inspect, the various session/context states they can maintain, and the specific profiles and the actions they can apply. An example of a session is "allowing outbound connection requests and only allowing return traffic from the external network".

Accordingly, the NSF capabilities are characterized by the level of packet processing and context that a NSF supports, the profiles and the actions that the NSF can apply. The term "context" includes anything that can influence the action(s) taken by the NSF, such as time of day, location, session state, and events.

Vendors can register their NSFs using Packet Content Match categories. The IDR Flow Specification [RFC5575] has specified 12 different packet header matching types. More packet content matching types have been proposed in the IDR WG. I2NSF should re-use the packet matching types being specified as much as possible. More matching types might be added for Flow-based NSFS. Tables 1-4 below list the applicable packet content categories that can be potentially used as packet matching types by Flow-based NSFs:

Packet Content Matching Capability Index	
Layer 2 Header	Layer 2 header fields: Source/Destination/s-VID/c-VID/EtherType/.
Layer 3	Layer header fields:
IPv4 Header	protocol
	dest port
	src port
	src address
	dest address
	dscp
	length
	flags
	ttl
IPv6 Header	
	addr
	protocol/nh
	src port
	dest port
	src address
	dest address
	length
	traffic class
	hop limit
	flow label
	dscp
TCP	Port
SCTP	syn
DCCP	ack
	fin
	rst
	? psh

		? urg
		? window
		sockstress
		Note: bitmap could be used to represent all the fields
UDP		
		flood abuse
		fragment abuse
		Port
HTTP layer		
		hash collision
		http - get flood
		http - post flood
		http - random/invalid url
		http - slowloris
		http - slow read
		http - r-u-dead-yet (rudy)
		http - malformed request
		http - xss
		https - ssl session exhaustion
IETF PCP	Configurable Ports	
IETF TRAM	profile	

Table 1: Subject Capability Index

context	matching Capability Index
Session	Session state, bidirectional state
Time	time span, time occurrence
Events	Event URL, variables
Location	Text string, GPS coords, URL
Connection	Internet (unsecured), Internet

| Type | (secured by VPN, etc.), Intranet, ... |

Direction	Inbound, Outbound
State	Authentication State
	Authorization State
	Accounting State
	Session State

Table 2: Object Capability Index

Action Capability Index	
Ingress port	SFC header termination,
	VxLAN header termination
Actions	Pass
	Deny
	Mirror
	Simple Statistics: Count (X min; Day;..)
	Client specified Functions: URL
Egress	Encap SFC, VxLAN, or other header

Table 3: Action Capability Index

Functional profile Index	
Profile types	Name, type, or
Signature	Flexible Profile/signature URL
	Command for Controller to enable/disable

Table 4: Function Capability Index

6. Structure of Rules for governing NSFs

6.1. Capability Layer Rules and Monitoring

The purpose of the Capability Layer is to define explicit rules for individual NSFs to treat packets, as well as methods to monitor the execution status of those functions.

[ACL-MODEL] has defined rules for the Access Control List supported by most routers/switches that forward packets based on packets' L2, L3, or sometimes L4 headers. The actions for Access Control Lists include Pass, Drop, or Redirect.

The functional profiles (or signatures) for NSFs are not present in [ACL-MODEL] because the functional profiles are unique to specific NSFs. For example, most vendors' IPS/IDS have their proprietary functions/profiles. One of the goals of I2NSF is to define a common envelop format for exchanging or sharing profiles among different organizations to achieve more effective protection against threats.

The "packet content matching" of the I2NSF policies should not only include the matching criteria specified by [ACL-MODEL] but also the L4-L7 fields depending on the NSFs selected.

Some Flow-based NSFs need matching criteria that include the context associated with the packets.

The I2NSF "actions" should extend the actions specified by [ACL-MODEL] to include applying statistics functions, threat profiles, or signature files that clients provide.

Policy consistency among multiple security function instances is very critical because security policies are no longer maintained by one central security device, but instead are enforced by multiple security functions instantiated at various locations.

6.2. Service Layer Policy

This layer is for clients or an Application Controller to express and monitor the needed security policies for their specific flows.

Some Customers may not have security skills. As such, they are not able to express requirements or security policies that are precise enough. These customers may instead express expectations or intent of the functionality desired by their security policies. Customers may also express guidelines such as which certain types of destinations are not allowed for certain groups. As a result, there could be different depths or layers of Service Layer policies. Here are some examples of more abstract service layer security Policies:

- Pass for Subscriber "xxx"
- enable basic parental control

- o enable "school protection control"
- o allow Internet traffic from 8:30 to 20:00
- o scan email for malware detection protect traffic to corporate network with integrity and confidentiality
- o remove tracking data from Facebook [website = *.facebook.com]
- o my son is allowed to access facebook from 18:30 to 20:00

One Service Layer Security Policy may need multiple security functions at various locations to achieve the enforcement. Service layer Security Policy may need to be updated by clients or Application controllers when clients' service requirements have been changed. Some service layer policies may not be granted because the carrier or Enterprises imposes additional constraints on what a client can have. [I2NSF-Demo] describes an implementation of translating a set of service layer policies to the Capability Layer instructions to NSFs.

I2NSF will first focus on simple service layer policies that are modeled as closely as possible on the Capability Layer. The I2NSF simple service layer should have similar structure as the I2NSF capability layer, but with more of a client-oriented expression for the packet content, context, and other parts of an ECA policy rule. This enables the client to construct an ECA policy rule without having to know its detailed structure or syntax.

There have been several industry initiatives to address network policies, such as OpenStack's Group-based Policy (GBP), IETF Policy Core Information Model-PCIM [RFC3060, RFC3460], and others. I2NSF will not work on general network service policies, but instead will define a standard interface for clients/applications to inform the Flow-based NSFs on the rules for treating traffic.

However, the notion of Groups (or roles), Target, Event, Context (or Conditions), and Action do cover what is needed for clients/applications to express the rules on how their flows can be treated by the Flow-Based NSFs in networks. The goal is to have a policy structure that can be mapped to the Capability layer's Event-Condition-Action paradigm.

The I2NSF simple service layer can have the following entities:

- I2NSF-Groups: This is a collection of users, applications, virtual networks, or traffic patterns to which a service

layer policy can be applied. An I2NSF-Group may be mapped to a client virtual Subnet (i.e. with private address prefix), a subnet with public address families, specific applications, destinations, or any combination of them with logical operators (Logical AND, OR, or NOT). An I2NSF-Group can have one or more Policy Rules applied to it.

- Target. This is used by the application client to identify the set of objects to be affected by the policy rules. A Target can be mapped to a physical/logical ingress port, a set of destinations, or a physical/logical egress port.
- Policy Rule. A Policy Rule consists of a set of Policy Events, Policy Conditions, and Policy Actions. Policy Rules are triggered by matching Events. If the Event portion of the Policy Rule evaluates to true, then the Condition portion is evaluated (otherwise, the Policy Rule terminates and no action is taken). If the Condition portion of the Policy Rule evaluates to true, then the set of Actions MAY be executed and applied to the traffic (otherwise, the Policy Rule terminates and no action is taken).
- Policy Event. This triggers a determination of whether the condition portion of a Policy Rule should be evaluated or not.
- Policy Condition. This determines when the Policy Actions contained in a Policy Rule are to be applied. It can be expressed as a direction, a list of L4 ports, time range, or a protocol, etc.
- Policy Action: This is the action applied to the traffic that matches the Conditions (and was triggered by the Events). An action may be a simple ACL action (i.e. allow, deny, mirroring), applying a well known statistics functions (e.g. X minutes count, Y hours court), applying client specified functions (with URL provided), or may refer to an ordered sequence of functions.

7. Capability Negotiation

When an NSF can't perform the desired provisioning (e.g., due to resource constraints), it MUST inform the controller.

The protocol needed for this security function/capability negotiation may be somewhat correlated to the dynamic service parameter negotiation procedure [RFC7297]. The Connectivity Provisioning Profile (CPP) template documented in RFC7297, even though currently covering only Connectivity requirements (but includes security clauses such as isolation requirements, non-via nodes, etc.), could be extended as a basis for the negotiation procedure. Likewise, the companion Connectivity Provisioning Negotiation Protocol (CPNP) could be a candidate to proceed with the negotiation procedure.

The "security as a service" would be a typical example of the kind of (CPP-based) negotiation procedures that could take place between a corporate customer and a service provider. However, more security specific parameters have to be considered.

8. Types of I2NSF clients

It is envisioned that I2NSF clients include:

- Application Controller:
    - For example, Video Conference Mgr/Controller needs to dynamically inform network to allow or deny flows (some of which are encrypted) based on specific fields in the packets for a certain time span. Otherwise, some flows can't go through the NSFs (e.g. FW/IPS/IDS) in the network because the payload is encrypted or packets' protocol codes are not recognized by those NSFs.
- Security Administrators
    - Enterprise users and applications

- Operator Management System dynamically updates, monitors and verifies the security policies to NSFs (by different vendors) in a network.
- Third party system

- Security functions send requests for more sophisticated functions upon detecting something suspicious, usually via a security controller.

9. Manageability Considerations

Management of NSFs usually includes:

- life cycle management and resource management of NSFs
- configuration of devices, such as address configuration, device internal attributes configuration, etc,
- signaling, and
- policy rules provisioning.

I2NSF will only focus on the policy rule provisioning part, i.e., the last bullet listed above.

10. Security Considerations

Having a secure access to control and monitor NSFs is crucial for hosted security service. Therefore, proper secure communication channels have to be carefully specified for carrying the controlling and monitoring information between the NSFs and their management entity (or entities).

11. IANA Considerations

This document requires no IANA actions. RFC Editor: Please remove this section before publication.

12. References

12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3060] Moore, B, et al, "Policy Core Information Model (PCIM)", RFC 3060, Feb 2001.

[RFC3460] Moore, B. "Policy Core Information Model (PCIM) Extensions", RFC3460, Jan 2003.

[RFC5575] Marques, P, et al, "Dissemination of Flow Specification Rules", RFC 5575, Aug 2009.

[RFC7297] Boucadair, M., "IP Connectivity Provisioning Profile", RFC7297, April 2014.

12.2. Informative References

[I2NSF-ACCESS] A. Pastor, et al, "Access Use Cases for an Open OAM Interface to Virtualized Security Services", <draft-pastor-i2nsf-access-usecases-00>, Oct 2014.

[I2NSF-DC] M. Zarny, et al, "I2NSF Data Center Use Cases", <draft-zarny-i2nsf-data-center-use-cases-00>, Oct 2014.

[I2NSF-MOBILE] M. Qi, et al, "Integrated Security with Access Network Use Case", <draft-qi-i2nsf-access-network-usecase-00>, Oct 2014

[I2NSF-Problem] L. Dunbar, et al "Interface to Network Security Functions Problem Statement", <draft-dunbar-i2nsf-problem-statement-01>, Jan 2015

[ACL-MODEL] D. Bogdanovic, et al, "Network Access Control List (ACL) YANG Data Model", <draft-ietf-net-acl-model-00>, Nov 2014.

[gs_NFV] ETSI NFV Group Specification, Network Functions Virtualizsation (NFV) Use Cases. ETSI GS NFV 001v1.1.1, 2013.

[NW-2011] J. Burke, "The Pros and Cons of a Cloud-Based Firewall", Network World, 11 November 2011

[SC-MobileNetwork] W. Haeffner, N. Leymann, "Network Based Services in Mobile Network", IETF87 Berlin, July 29, 2013.

[I2NSF-Demo] Y. Xie, et al, "Interface to Network Security Functions Demo Outline Design", <draft-xie-i2nsf-demo-outline-design-00>, April 2015.

[ITU-T-X1036] ITU-T Recommendation X.1036, "Framework for creation, storage, distribution and enforcement of policies for network security", Nov 2007.

13. Acknowledgments

Acknowledgements to xxx for his review and contributions.

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Edward Lopez
Fortinet
899 Kifer Road
Sunnyvale, CA 94086
Phone: +1 703 220 0988
Email: elopez@fortinet.com

Diego Lopez
Telefonica
Email: diego.r.lopez@telefonica.com

XiaoJun Zhuang
China Mobile
Email: zhuangxiaojun@chinamobile.com

Linda Dunbar
Huawei
Email: Linda.Dunbar@huawei.com

John Strassner
Huawei
John.sc.Strassner@huawei.com

Joe Parrott
BT
Email: joe.parrott@bt.com

Ramki Krishnan
Dell
Email: ramki_krishnan@dell.com

Seetharama Rao Durbha
CableLabs
Email: S.Durbha@cablelabs.com