Network Working Group
Internet Draft
Intended status: Informational
Expires: January 2017

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

August 17, 2016

Framework for Interface to Network Security Functions
draft-ietf-i2nsf-framework-03.txt

Status of this Memo

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 February 17, 2009.

Copyright Notice

Abstract

This document describes the framework for the Interface to Network Security Functions (I2NSF), and defines a reference model (including major functional components) for 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.

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, including an analysis of the threats implied by the deployment of NSFs that are externally provided. It also describes how I2NSF facilitates Software-Defined Networking (SDN) and Network Function Virtualization (NFV) 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, overlay or cloud network management system, or enterprise network administrator or management system), 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 flows. It also provides standard interfaces for them to monitor the

flow based security functions hosted and managed by different administrative domains.

[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

3. I2NSF Reference Model

The following figure shows a reference model (including major functional components) for I2NSF and the interfaces among those components.

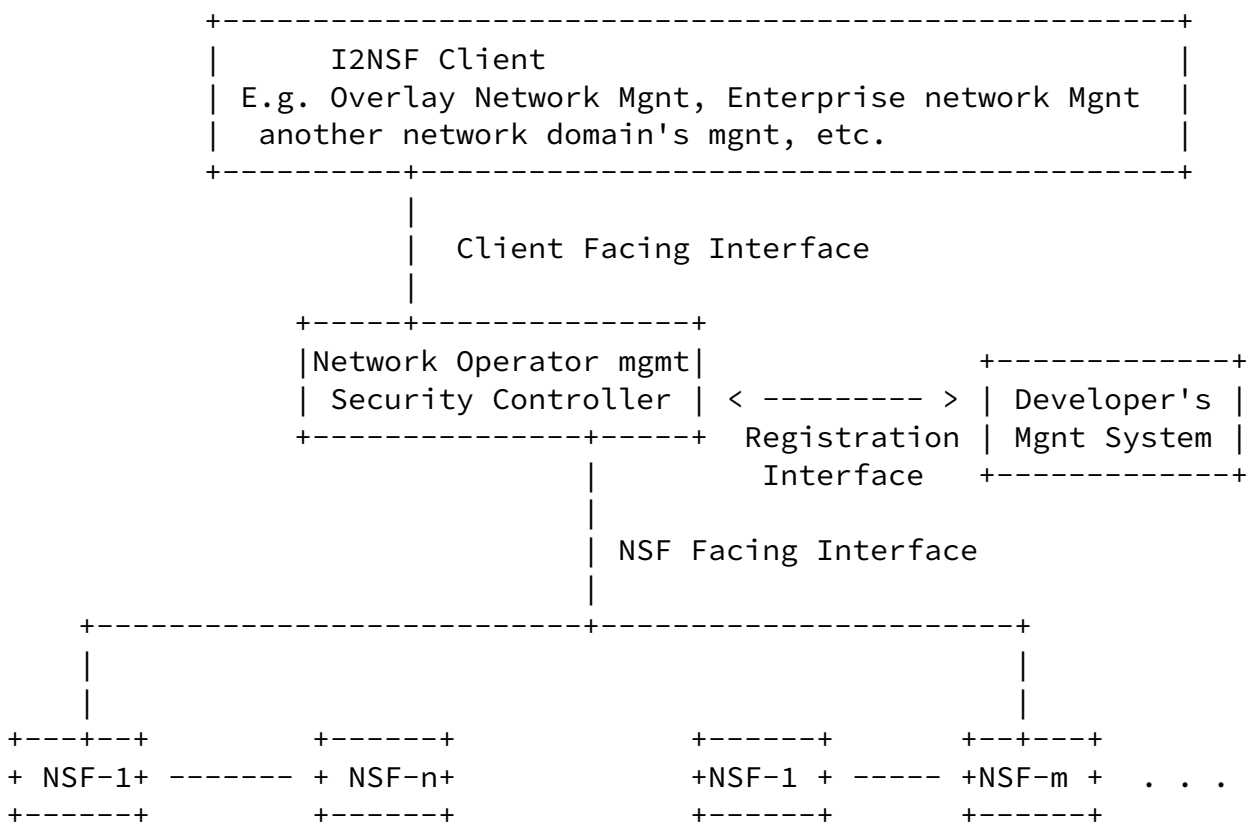

Figure 1: I2NSF Reference Model

Note that security controller focuses on the management of NSFs, and it does not participate in the NSF data plane activity. The diagram has to be interpreted from the point of view of the security controller and does not assume any particular management architecture for the NSFs or on the developer's side.

When defining controller interfaces, this framework adheres to the following principles:

- Agnostic of network topology and NSF location in the network
- Agnostic of provider, implementation and form-factor (physical, virtual)
- Agnostic to how NSF is implemented and its hosting environment
- Agnostic to how NSF becomes operational
- Agnostic to NSF control plane implementation (if there is one)
- Agnostic to NSF data plane implementation such as encapsulation capabilities.

3.1. Client Facing Interface

The Client Facing Interface, which is often loosely called the north bound interface to the controller, is for clients to express and monitor security policies for clients' specific flows through an administrative domain.

In today's world, where everything is connected, preventing unwanted traffic has become a key challenge. More and more networks, including various types of Internet of Things (IoT) networks, information-centric networks (ICN), content delivery networks (CDN), and cloud networks, are some form of overlay networks with their paths (or links) among nodes being provided by other networks (a.k.a. underlay networks). The overlay networks' own security solutions cannot prevent various attacks from saturating the access links to the overlay network nodes, which may cause overlay nodes' CPU/links too overloaded to handle their own legitimate traffic.

Very much like traditional networks placing firewall or intrusion prevention system (IPS) on the wire to enforce traffic rules, Interface to Network Security Functions (I2NSF) can be used by overlay networks to request certain flow-based security rules to be enforced by underlay networks. With this mechanism, unwanted traffic, including DDoS attacks, can be eliminated from occupying the physical links and ports to the overlay network nodes, thereby avoiding excessive or problematic overlay node CPU/storage/port utilization. The same approach can be used by enterprise networks to request their specific flow security policies to be enforced by the provider network that interconnect their users. Clients may not care where a specific security policy is implemented.

Here are some examples of I2NSF clients:

- A videoconference network manager that needs to dynamically inform the underlay network to allow, rate-limit or deny flows (some of which are encrypted) based on specific fields in the packets for a certain time span.
- Enterprise network administrators and management systems that need to request their provider network to enforce some rules to their specific flows.

- An IoT management system sending requests to the underlay network to block flows that match their specific conditions.

3.2. NSFs Facing Interface

The NSFs Facing Interface, which is often loosely called the south bound interface to the controller, specifies and monitors a number of flow based security policies to individual NSFs. Note that the controller 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 developers to be selected.

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) Registration Interface - Interface to discover NSF/vNSF capability so that controller can determine whether a NSF is capable of enforcing a given policy. This could be either a query interface (controller queries from a NSF for a specific functionality) or a report interface where each NSF sends its supported capabilities such as feature, scale, performance. The operational state of the NSF is not changed by this interface.
2) Capability Interface - Interface used by controller to program a specific NSF to enforce a security policy. This changes the operational state of the NSF if successful. Due to the need of applications/controllers to dynamically control what traffic they need to receive, much of the I2NSF efforts would be focused towards this interface.
3) Monitoring Interface - Interface to get monitoring information from a NSF. This could be a query or report interface. This includes logging and query functions between the NSF and external systems. The functions for this interface may also be defined by other protocols, such as SYSLOG and DOTS.
4) Notification Interface - Interface used for notification (event/alarm) from a NSF to the controller (if registered). The controller may take an action based on the event. This is a report and registry interface. This does not change the operational state of the NSF.

This draft proposes that a capability interface to NSFs can be developed on a flow-based paradigm. A common trait of flow based 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.

3.3. Registration Interface

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

NSF's capabilities can be either pre-configured or retrieved dynamically on a need basis through the registration interface. If a new functionality that is exposed to the user is added to an NSF, then those capabilities must be notified to security controller via the Registration Interface.

4. Threats Associated with Externally Provided NSFs

While associated with a much higher flexibility, and in many cases a necessary approach given the deployment conditions, the usage of externally provided NSFs implies several additional concerns in security. The most relevant threats associated with a security platform of this nature are:

- An unknown/unauthorized client can try to impersonate another client that can legitimately access external NSF services. This attack may lead to accessing the policies and applications of the attacked client or to generate network traffic outside the security functions with a falsified identity.
- An authorized client may misuse assigned privileges to alter the network traffic processing of other clients in the NSF underlay or platform. This can become especially serious when such a client has higher (or even administration) privileges granted by the provider (the direct NSF provider, the ISP or the underlay network operator).

- o A client may try to install malformed elements (policy or configuration), trying to directly take the control of a NSF or the whole provider platform, for example by exploiting a vulnerability on one of the functions, or may try to intercept or modify the traffic of other clients in the same provider platform.
- o A malicious provider can modify the software providing the functions (the operating system or the specific NSF implementations) to alter the behavior of the latter. This event has a high impact on all clients accessing NSFs as the provider has the highest level of privilege on the software in execution.
- o A client that has physical access to the provider platform can modify the behavior of the hardware/software components, or the components themselves. Furthermore, it can access a serial console (most devices offer this interface for maintenance reasons) to access the NSF software with the same level of privilege of the provider.

The authentication between the client and the NSF environment and, what is more important, the attestation of the elements in the NSF environment by clients could address these threats to an acceptable level of risk. Periodical attestation enables clients to detect alterations in the NSFs and their supporting infrastructure able, and raises the degree of physical control necessary to perform an untraceable malicious modification of the environment.

5. Avoiding NSF Ossification

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 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 developer-specific functions

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

To prevent constraints on NSF developers' 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. 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.

6. The Network Connecting I2NSF Components

6.1. Network connecting I2NSF Clients and I2NSF Controller

Editor's note: should we add the Remote Attestation to this section?

As a general principle, in the I2NSF environment clients directly interact with the controller. Given the role of the Security Controller, a mutual authentication of clients and the Security Controller maybe required. I2NSF does not mandate a specific authentication scheme; it is up to the users to choose available authentication scheme based on their needs.

Upon successful authentication, a trusted connection between the client and the Security Controller (or an endpoint designated by it) SHALL be established. All traffic to and from the NSF environment will flow through this connection. The connection is intended not only to be secure, but trusted in the sense that it

SHOULD be bound to the mutual authentication between client and Security Controller, as described in [Remote-Attestation], with the only possible exception of the application of the lowest levels of assurance, in which case the client MUST be made aware of this circumstance.

6.2. Network Connecting the Security Controller and NSFs

Most likely the NSFs are not directly attached to the I2NSF Controller; for example, NSFs can be distributed across the network. The network that connects the I2NSF 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 can rely either on:

- 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 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. The information over the I2NSF interfaces SHALL be exchanged used trusted channels as described in the previous section.

  When running in an open environment, I2NSF needs to rely on interfaces to properly verify peer identities e.g. through an

AAA framework. The implementation of identity management functions is out of scope for I2NSF.

6.3. 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 Manager, 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.

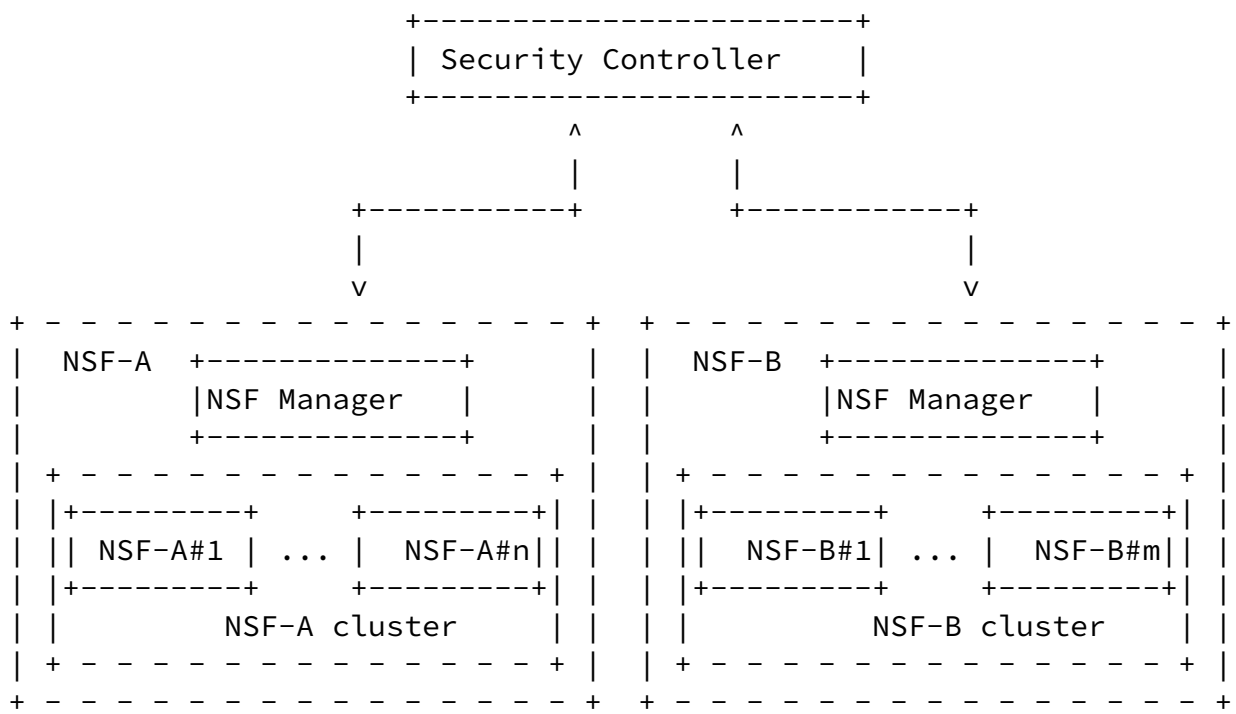

Figure 2: Cluster of NSF Instantiations Management

7. I2NSF Flow Security Policy Structure

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.

Event is used to determine whether the condition clause of the Policy Rule can be evaluated or not.

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.

Action can be simple permit/deny/rate-limiting, applying specify profile, or establishing specific secure tunnels, etc.

7.1. Client Facing Flow Security Policy structure

This layer is for client's network management system to express and monitor the needed flow 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 security Policies that can be developed based on the I2NSF defined client interfaces:

- Pass for Subscriber "xxx"
- Enable basic parental control
- Enable "school protection control"
- Allow Internet traffic from 8:30 to 20:00
- Scan email for malware detection protect traffic to corporate network with integrity and confidentiality
- Remove tracking data from Facebook [website = *.facebook.com]
- My son is allowed to access Facebook from 18:30 to 20:00

One flow policy over Client Facing Interface may need multiple network functions at various locations to achieve the enforcement. Some flow Security policies from clients may not be granted because of resource constraints. [I2NSF-Demo] describes an implementation of translating a set of client policies to the flow policies to individual NSFs.

I2NSF will first focus on simple client policies that can be modeled as closely as possible to the flow security policies to individual NSFs. The I2NSF simple client flow policies should have similar structure as the policies to NSFs, 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 actual tags or addresses in the packets.

For example, when used in the context of policy rules over the

Client Facing Interface:

- An Event can be "the client has passed AAA process";
- A Condition can be matching client Identifier, or from specific ingress or egress points; and
- An action can be establishing a IPSec tunnel.

7.2. NSF Facing Flow Security Policy structure

The NSF Facing Interface is to pass explicit rules to individual NSFs to treat packets, as well as methods to monitor the execution status of those functions.

Here are some examples of Events over the NSF facing interface:

- time == 08:00,
- a NSF state change from standby to active

Here are some examples of Conditions over the NSF facing interface

- 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 to individual flow-based NSFs include:

- Action ingress processing, such as pass, drop, rate limiting, 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 NSF-specific today. The rulesets and software interfaces of I2NSF aim to specify the format to pass profile and signature files while supporting specific functionalities of each.

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.

7.3. Difference from ACL data model

[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 IPS/IDS implementations 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.

8. Capability Negotiation

It is very possible that the underlay network (or provider network) does not have the capability or resource to enforce the flow security policies requested by the overlay network (or enterprise network). Therefore, it is very important to have capability discovery or inquiry mechanism over the I2NSF Client Facing Interface for the clients to discover if the needed flow polices can be supported or not.

When an NSF cannot 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.

9. Registration consideration

9.1. 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.

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".

9.2. Registration Categories

Developers 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 SCTP DCCP	Port syn 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 Type	Internet (unsecured), Internet (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 Signature	Name, type, or Flexible Profile/signature URL Command for Controller to enable/disable

Table 4: Function Capability Index

10. 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.

11. 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).

12. IANA Considerations

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

13. References

13.1. 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.

13.2. 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.

14. 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

Rakesh Kumar
Juniper Networks
Email: rkkumar@juniper.net

Anil Lohiya
Juniper Networks
Email: alohiya@juniper.net