I2NSF
Internet-Draft
Intended status: Informational
Expires: November 3, 2017

D. Lopez
Telefonica I+D
E. Lopez
Curveball Networks
L. Dunbar
J. Strassner
Huawei
R. Kumar
Juniper Networks

July 2, 2017

Framework for Interface to Network Security Functions
draft-ietf-i2nsf-framework-06

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 to which the packet is associated.

Status of this Memo

Copyright Notice

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. This includes an analysis of the threats implied by the deployment of Network Security Functions (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 [I-D.ietf-i2nsf-problem-and-use-cases] call for standard interfaces for users of an I2NSF system (e.g., applications, overlay or cloud network management system, or enterprise network administrator or management system), to inform the I2NSF system which I2NSF functions should be applied to which traffic (or traffic patterns). The I2NSF system realizes this as a set of security rules for monitoring and controlling the behavior of different traffic. It also provides standard interfaces for users to monitor flow-based security functions hosted and managed by different administrative domains.

[I-D.ietf-i2nsf-problem-and-use-cases] also describes the motivation and the problem space for an Interface to Network Security Functions system.

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 [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.

Note: as this is an informational document, no RFC-2119 key words are used.

2.1. Acronyms

The following acronyms are used in this document:

IDS	Intrusion Detection System
IoT	Internet of Things
IPS	Intrusion Protection System
NSF	Network Security Function

2.2. Definitions

The following terms, which are used in this document, are defined in the I2NSF terminology document [I-D.ietf-i2nsf-terminology]:

Capability

Controller

Firewall

I2NSF Consumer

I2NSF NSF-Facing Interface

I2NSF Policy Rule

I2NSF Producer

I2NSF Registration Interface

I2NSF Registry

Interface

Interface Group

Intrusion Detection System

Intrusion Protection System

Network Security Function

Role

3. I2NSF Reference Model

Figure 1 shows a reference model (including major functional components and interfaces) for an I2NSF system. This figure is drawn from the point-of-view of the Controller; hence, this view does not assume any particular management architecture for either the NSFs or for how NSFs are managed (on the developer's side). In particular, the controller does not participate in NSF data plane activities.

Note that the term "Controller" is defined in [I-D.ietf-i2nsf-terminology].

```
       +--------------------------------------------------------+
       |  I2NSF User (e.g., Overlay Network Mgnt, Enterprise   |
       |  network Mgnt, another network domain's mgnt, etc.)    |
       +--------------------+-----------------------------------+
                            |
                            |  I2NSF Consumer-Facing Interface
                            |
                            |               I2NSF
               +------------+---------+ Registration  +-------------+
               | Network Operator Mgmt|  Interface    | Developer's |
               |      Controller      | < --------- > | Mgnt System |
               +----------------+-----+               +-------------+
                                |
                                | I2NSF NSF-Facing Interface
                                |
           +---------------+----+------------+---------------+
           |               |                 |               |
       +---+---+       +---+---+         +---+---+       +---+---+
       | NSF-1 |  ...  | NSF-m |         | NSF-1 |  ...  | NSF-m |  ...
       +-------+       +-------+         +-------+       +-------+

      Developer Mgnt System A              Developer Mgnt System B
```

Figure 1: I2NSF Reference Model

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

o Agnostic of network topology and NSF location in the network

o Agnostic of provider of the NSF (i.e., independent of the way that the provider makes an NSF available, as well as how the provider allows the NSF to be managed)

o Agnostic of any vendor-specific operational, administrative, and management implementation, hosting environment, and form-factor (physical or virtual)

o Agnostic to NSF control plane implementation (e.g., signaling capabilities)

o Agnostic to NSF data plane implementation (e.g., encapsulation capabilities)

3.1. I2NSF Consumer-Facing Interface

The I2NSF Consumer-Facing Interface is used to enable different users of a given I2NSF system to define, manage, and monitor security policies for specific flows within an administrative domain. The location and implementation of I2NSF policies are irrelevant to the consumer of I2NSF policies.

Some examples of I2NSF Consumers include:

o 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

o Enterprise network administrators and management systems that need to request their provider network to enforce specific I2NSF policies for particular flows

o An IoT management system sending requests to the underlay network to block flows that match a set of specific conditions.

3.2. I2NSF NSF-Facing Interface

The I2NSF NSF-Facing Interface (NSF-Facing Interface for short) is used to specify and monitor flow-based security policies enforced by one or more NSFs. Note that the controller does not need to use all features of a given NSF, nor does it need to use all available NSFs. Hence, this abstraction enables the different features from the set of NSFs that make up able given I2NSF system to be treated as building blocks, so that developers are free to use the security functions needed independent of vendor and technology.

Flow-based NSFs [I-D.ietf-i2nsf-problem-and-use-cases] inspect packets in the order that they are received. The Interface to flow-based NSFs can be grouped into the following types of Interface Groups:

1. NSF Operational and Administrative Interface: an Interface Group used by a controller to program the operational state of the NSF; this also includes administrative control functions. I2NSF Policy Rules represent one way to change this Interface Group in a consistent manner. Since applications and controllers need to dynamically control the behavior of traffic that they send and receive, much of the I2NSF effort is focused on this Interface Group.

2. Monitoring Interface: an Interface Group used by a controller to obtain monitoring information from one or more selected NSFs. Each interface in this Interface Group could be a query- or a report-based interface (as described above). This Interface Group includes logging and query functions between the NSF and external systems. The functionality of this Interface Group may also be defined by other protocols, such as SYSLOG and DOTS (DDoS Open Threat Signaling). This Interface Group does NOT change the operational state of the NSF.

3. Notification Interface: an Interface Group used by a controller to receive notification events (e.g., alarms) from NSFs. This requires the NSF to be registered. The controller may take an action based on the event; this should be specified by an I2NSF Policy Rule. This Interface Group does NOT change the operational state of the NSF.

This draft proposes that the flow-based paradigm is used to develop the NSF-Facing Interface. A common trait of flow-based NSFs is in the processing of packets based on the content (e.g., header/payload) and/or context (e.g., session state, authentication state) of the received packets.

3.3. I2NSF Registration Interface

NSFs provided by different vendors may 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 a dedicated interface for vendors to define the capabilities of (i.e., register) their NSFs. This Interface is called the I2NSF Registration Interface.

An NSF's capabilities can either be pre-configured or retrieved dynamically through the I2NSF Registration Interface. If a new function that is exposed to the consumer is added to an NSF, then the capabilities of that new function should be registered in the I2NSF Registry via the I2NSF Registration Interface, so that interested management and control entities may be made aware of them.

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 user can try to impersonate another user that can legitimately access external NSF services. This attack may lead to accessing the I2NSF Policy Rules and applications of the attacked user, and/or to generate network traffic outside the security functions with a falsified identity.

- An authorized user may misuse assigned privileges to alter the network traffic processing of other users in the NSF underlay or platform.

- A user may try to install malformed elements (e.g., I2NSF Policy Rules, or configuration files), trying to directly take the control of a NSF or the whole provider platform. For example, a user may exploit a vulnerability on one of the functions, or may try to intercept or modify the traffic of other users in the same provider platform.

- A malicious provider can modify the software (e.g., the operating system or the specific NSF implementation) to alter the behavior of one or more NSFs. This event has a high impact on all users accessing NSFs, since the provider has the highest level of privileges controlling the operation of the software.

- A user that has physical access to the provider platform can modify the behavior of the hardware/software components, or the components themselves. For example, the user 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 above threats may be mitigated by requiring the use of an AAA framework for all users to access the I2NSF environment. This could be further enhanced by requiring attestation to be used to detect changes to the I2NSF environment by authorized parties. Note that periodical attestation enables users to detect alterations in the NSFs and their supporting infrastructure, and raises the degree of physical control necessary to perform an untraceable malicious modification of the I2NSF 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 to not 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 I2NSF Interfaces as well as I2NSF Policy Rules, 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 I2NSF Interfaces to communicate with NSFs, as well as I2NSF Policy Rules to control NSF functionality, should not:

- o Narrowly define NSF categories, or their roles, when implemented within a network
- o Attempt to impose functional requirements or constraints, either directly or indirectly, upon NSF developers
- o Be a limited lowest common denominator approach, where interfaces can only support a limited set of standardized functions, without allowing for developer-specific functions
- o 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 Users and I2NSF Controller

As a general principle, in the I2NSF environment users directly interact with the controller. Given the role of the Security Controller, a mutual authentication of users 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 user and the 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 the user and the Controller, as described in [I-D.pastor-i2nsf-remote-attestation]. The only possible exception is when the required level of assurance is lower, (see Section 4.1 of [I-D.pastor-i2nsf-remote-attestation], in which case the user must be made aware of this circumstance.

[TBD: should we add the Remote Attestation to this section?]

6.2. Network Connecting the 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) and the Stream Control Transmission Protocol (SCTP) will need to be evaluated for applicability. Latency requirements for control message delivery must also be evaluated.

The network connection between the Controller and NSFs can rely either on:

- o 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 nature of a closed environment.

- o Open environments, where one or more NSFs can be hosted in one or more external administrative domains that are reached via secure external network connections. 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 implementations of identity management functions, as well as the AAA framework, are out of scope for I2NSF.

6.3. Interface to vNSFs

There are some unique characteristics in interfacing to virtual NSFs:

- o There could be multiple instantiations of one single NSF that has been distributed across a network. When different instantiations are visible to the 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). Therefore, it is recommended that Roles, in addition to the use of robust identities, be used to distinguish between different instantiations of the same vNSF.

- o When multiple instantiations of one single NSF appear as one single entity to the Controller, the Controller may need to either get assistance from other entities in the I2NSF Management System, and/or delegate the provisioning of the multiple instantiations of the (single) NSF to other entities in the I2NSF Management System. This is shown in Figure 2 below.

- o Policies to one vNSF may need to be retrieved and moved to another vNSF of the same type when user flows are moved from one vNSF to another.

- o Multiple vNSFs may share the same physical platform.

- o There may be scenarios where multiple vNSFs collectively perform the security policies needed.

```
                          +-------------------------+
                          |       Controller        |
                          +-------------------------+
                                   ^        ^
                                   |        |
                       +-----------+        +------------+
                       |                                 |
                       v                                 v
    + - - - - - - - - - - - - - - - +  + - - - - - - - - - - - - - - - +
    |  NSF-A  +--------------+      |  |  NSF-B  +--------------+      |
    |         |NSF Manager   |      |  |         |NSF Manager   |      |
    |         +--------------+      |  |         +--------------+      |
    | + - - - - - - - - - - - - - + |  | + - - - - - - - - - - - - - + |
    | |+---------+     +---------+| |  | |+---------+     +---------+| |
    | || NSF-A#1 | ... |  NSF-A#n|| |  | ||  NSF-B#1| ... |  NSF-B#m|| |
    | |+---------+     +---------+| |  | |+---------+     +---------+| |
    | |         NSF-A cluster     | |  | |          NSF-B cluster    | |
    | + - - - - - - - - - - - - - + |  | + - - - - - - - - - - - - - + |
    + - - - - - - - - - - - - - - - +  + - - - - - - - - - - - - - - - +
```

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 policy rules.

In this version of I2NSF, policy rules are limited to imperative paradigms. I2NSF is using an Event - Condition - Action (ECA) policy, where:

- o An Event clause is used to trigger the evaluation of the Condition clause of the Policy Rule.

- o A Condition clause is used to determine whether or not the set of Actions in the I2NSF Policy Rule can be executed or not.

- o An Action clause defines the type of operations that may be performed on this packet or flow.

Each of the above three clauses are defined to be Boolean clauses. This means that each is a logical statement that evaluates to either TRUE or FALSE.

The above concepts are described in detail in [I-D.draft-xibassnez-i2nsf-capability].

7.1. Customer-Facing Flow Security Policy Structure

This layer is for user'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 customer-facing 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 Customer-Facing Interface may need multiple network functions at various locations to achieve the enforcement. Some flow security policies from users may not be granted because of resource constraints. [I-D.xie-i2nsf-demo-outline-design] describes an implementation of translating a set of user 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 I2NSF 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 user identifier, or from specific ingress or egress points

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

- o Packet content values are based on one or more packet headers, data from the packet payload, bits in the packet, or data that are derived from the packet

- o 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:

- o Action ingress processing, such as pass, drop, rate limiting, mirroring, etc.

- o Action egress processing, such as invoke signaling, tunnel encapsulation, packet forwarding and/or transformation.

- o 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. Differences from ACL Data Models

Policy rules are very different from ACLs. An ACL is NOT a policy. Rather, policies are used to manage the construction and lifecycle of an ACL.

[I-D.ietf-netmod-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 [I-D.ietf-netmod-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 [I-D.ietf-netmod-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. This may also include metadata.

The I2NSF "actions" should extend the actions specified by [I-D.ietf-netmod-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 a capability discovery or inquiry mechanism over the I2NSF Customer-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 described in [RFC7297]. The Connectivity Provisioning Profile (CPP) template, even though currently covering only Connectivity requirements, includes security clauses such as isolation requirements and non-via nodes. Hence, could be extended as a basis for the negotiation procedure. Likewise, the companion Connectivity Provisioning Negotiation Protocol (CPNP) could be a candidate for the negotiation procedure.

"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.

[I.D.-draft-xibassnez-i2nsf-capability] describes the concepts of capabilities in detail.

9. Registration Considerations

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 more powerful technology, integration of platforms, and new threats. Basic types of flow-based NSFs include:

o 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.

o 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.

o 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 (Inter-Domain Routing) 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       | Layer 2 header fields:                    |
| Header        | Source/Destination/s-VID/c-VID/EtherType/.|
|               |                                           |
|---------------+-------------------------------------------+
| Layer 3       | Layer  header fields:                     |
|               |            protocol                       |
| IPv4 Header   |            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: Packet Content Matching Capability Index

Note:	DCCP:	Datagram Congestion Control Protocol
	PCP:	Port Control Protocol
	TRAM:	TURN Revised and Modernized, where TURN stands for Traversal Using Relays around NAT

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: Context Matching 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 Profile Index

10. Manageability Considerations

Management of NSFs includes:

- o Lifecycle management and resource management of NSFs
- o Configuration of devices, such as address configuration, device internal attributes configuration, etc.
- o Signaling
- o Policy rules provisioning

Currently, I2NSF only focuses 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 services. 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. Acknowledgements

This document includes significant contributions from Seetharama Rao Durbha (Cablelabs), Ramki Krishnan (Dell), Anil Lohiya (Juniper Networks), Joe Parrott (BT), and XiaoJun Zhuang (China Mobile).

Some of the results leading to this work have received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 611458.

14. References

14.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC5575] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J., and D. McPherson, "Dissemination of Flow Specification Rules", RFC 5575, DOI 10.17487/RFC5575, August 2009, <http://www.rfc-editor.org/info/rfc5575>.

[RFC7297] Boucadair, M., Jacquenet, C., and N. Wang, "IP Connectivity Provisioning Profile (CPP)", RFC 7297, DOI 10.17487/RFC7297, July 2014, <http://www.rfc-editor.org/info/rfc7297>.

14.2. Informative References

[I-D.ietf-i2nsf-problem-and-use-cases]
Hares, S., Dunbar, L., Lopez, D., Zarny, M., and C. Jacquenet, "I2NSF Problem Statement and Use cases",

draft-ietf-i2nsf-problem-and-use-cases-16 (work in progress), May 2017.

[I-D.ietf-netmod-acl-model]
Bogdanovic, D., Sreenivasa, K., Huang, L., and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-11 (work in progress), June, 2017.

[I-D.ietf-i2nsf-terminology]
Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-03 (work in progress), March 2017.

[I-D.draft-xibassnez-i2nsf-capability]
Xia, L., Strassner, J., Basile, C., and Lopez, D., "Information Model of NSFs Capabilities", draft-xibassnez-i2nsf-capability-01.txt (work in progress), March, 2017.

[I-D.pastor-i2nsf-remote-attestation]
Pastor, A., Lopez, D., and A. Shaw, "Remote Attestation Procedures for Network Security Functions (NSFs) through the I2NSF Security Controller", draft-pastor-i2nsf-nsf-remote-attestation-01 (work in progress), March 2017.

[I-D.xie-i2nsf-demo-outline-design]
Xie, Y., Xia, L., and J. Wu, "Interface to Network Security Functions Demo Outline Design", draft-xie-i2nsf-demo-outline-design-00 (work in progress), April 2015.

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

Authors' Addresses

Diego R. Lopez
Telefonica I+D
Editor Jose Manuel Lara, 9
Seville, 41013
Spain

Phone: +34 682 051 091
Email: diego.r.lopez@telefonica.com

Edward Lopez
Curveball Networks
Chantilly, Virgina
USA

Phone: +1 703 220 0988
Email: elopez@fortinet.com

Linda Dunbar
Huawei

Email: Linda.Dunbar@huawei.com

John Strassner
Huawei

Email: John.sc.Strassner@huawei.com

Rakesh Kumar
Juniper Networks

Email: rkkumar@juniper.net