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

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

October 15, 2015

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

Status of this Memo

Copyright Notice

Abstract

This document serves as the framework for detailed work items for I2NSF. In the design of interfaces to allow for the provisioning of network security functions (NSFs), a critical consideration is to prevent the creation of implied constraints.

This document makes the recommendation that such interfaces be designed from the paradigm of processing packets and flows on the network. NSFs ultimately are packet-processing engines that inspect packets traversing networks, either directly or in context to sessions to which the packet is associated.

Table of Contents

1. Introduction

This document describes the framework for Interface to Network Security Functions (I2NSF) and defines a reference model along with functional components for I2NSF. It also describes how I2NSF facilitates Software-defined network (SDN) and Network Function Virtualization (NVF) control, while avoiding potential constraints which could limit NSFs internal functions.

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 network what they are willing to receive, when and how their specific data should be delivered. And provide the standard interface 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 has 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] inspect and treat packets in the order as 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, supported protocols, and/or addresses configurations. Configuration deals with attributes that are don't change very much.

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

3) Rules Provisioning - used to control the rules that govern how packets are treated by the NSFs. To enable applications, application controllers or clients to dynamically control what/when/how traffic they want 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-based paradigm. While there are many classifications of existing and emerging NSFs, a common trait shared by them is in the processing of packets based on the content (header/payload) and context (session state, authentication state, etc) of received packets.

An important concept 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. Therefore, in constructing standards for rules provisioning interfaces to NSFs, it is equally important to allow support for vendor-specific functions, to allow 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-based approach to the design of such provisioning interfaces, the goal is to create a workable interface to NSFs which aid in their integration within SDN/NFV environments, while avoiding potential constraints which could limit their functional capabilities.

Even though security functions come in variety of form factors and have different features, provisioning to Flow-based NSFs can be categorized by

- Subject - Match values based on packet data Packet header or Packet payload, which can be one or more header fields or bits in the packets, or the various combination of them;
- Object - Match values based on context, e.g. State, direction of the traffic, time, geo-location, etc.,
- Action- Egress processing, such as Invoke signaling; Packet forwarding and/or transformation; Possibility for SDN/NFV integration, and
- Functional Profile - E.g. IPS:<Profile>, signature file, Anti-virus file, URL filtering file, etc. Integrated and one-pass checks on the content of packets.

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.

4. Reference Models in Managing NSFs

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

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

```
             +-------------------------------------------+
             |       Client or App Gateway               |
             |       (e.g. Video conference Ctrl         |
             | Admin, OSS/BSS, or Service Orchestration)|
             +-------+-----------------------------------+
                         |
                         |  Client Facing (service layer) Interface
                   +-----+--------------+
                   |Service Provider mgmt|              +-------------+
                   | Security Controller | < -------- > | Vendor      |
                   +---------------------+ Vendor Facing|  Sys        |
                               |          Interface  +-------------+
                               |
                               | NSF Facing (capability) Interface
                               |
        +-------------------------------------------------+
        |                                                 |
        |                                                 |
   +------+         +------+             +------+      +------+
   + NSF-1+ ------- + NSF-n+             +NSF-1 + ----- +NSF-m +  . . .
   +------+         +------+             +------+      +------+

   Vendor A                                      Vendor B
```

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 NSFs that are selected to enforce the desired network security. This interface is called 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 Server Layer Interface in the I2NSF context. The I2NSF Service Layer also allows clients to monitor the client specific policies and execution status.

A single client layer policy may need multiple NSFs or NSF instantiations collectively together to achieve the enforcement.

4.3. Vendor Facing Interface

When service providers have multiple types of security functions provided by different vendors, it is necessary to have an interface for vendors to register their NSFs indicating their NSFs capabilities.

The Registration Interface can be static or dynamic. When NSFs are upgraded, vendors need to notify the service provider management system or controller of the updated capabilities.

4.4. The network connecting the Security Controller and NSFs

Most likely, the NSFs are not directly attached to the Security Controller; it is especially true when NSFs are distributed across the network. The network that connects the Security Controller with the NSFs can be the same network that carry the data traffic, or can be a dedicated network for management purpose only. 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 connection between Security Controller and NSFs could be:

- Closed environments where there is only one administrative domain. More permissive access controls and lighter validation is needed 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 external network domains. Then more restrictive security controls are required over the I2NSF interface. The information over the I2NSF interfaces must use trusted channels, such as TLS, SASL, or the combination of the two.

Over the Open Environment, I2NSF needs to provide the identity frameworks and federations models for authentication and Authorization.

4.5. Interface to vNSFs

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

- There could be multiple instantiations of one single NSF being distributed across network. When different instantiations are visible to the Security Controller, different policies may be applied to different instantiations of one single NSF.
- 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 disseminate 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 move 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.

```
                          +-------------------------+
                          | Security Controller     |
                          +-------------------------+
                                   ^        ^
                                   |        |
                       +-----------+        +------------+
                       |                                 |
                       v                                 v
    + - - - - - - - - - - - - - - - - +  + - - - - - - - - - - - - - - - - +
    |  NSF-A  +--------------+        |  |  NSF-B  +--------------+        |
    |         |Sub Controller|        |  |         |sub Controller|        |
    |         +--------------+        |  |         +--------------+        |
    | + - - - - - - - - - - - - - - + |  | + - - - - - - - - - - - - - - + |
    | |+---------+     +---------+| |  | |+---------+     +---------+| |
    | || 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

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 somewhat as technological capacity increases, platforms are integrated, and the threat landscape shifts. 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, and/or destination port. Packets that do not match policy are rejected.
- IDS (Intrusion Detection System) - A device or function that analyzes whole packets, both header and payload, looking for known events. When a known event is detected a log message is generated detailing the event.
- IPS (Intrusion Prevention System) - A device or function that analyzes whole packets, both header and payload, looking for known events. When a known event is detected the packet is rejected.

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 on the network. Flow-based NSFs ultimately are packet-processing engines that inspect packets traversing networks, either directly or in context to sessions to 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, the specific profiles and the actions they can apply. An example of 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 session state, timer, and events.

Vendors can register their NSFs using the Subject-Object-Action-Function categories described in Section 2, with detailed specification of each category as shown in the table below:

```
  +-----------------------------------------------------------+
  |         Subject Capability Index                          |
  +---------------+-------------------------------------------+
  | Layer 2       | Layer 2 header fields:                    |
  | Header        | Source/Destination/s-VID/c-VID/EtherType/.|
  |               |                                           |
  |---------------+-------------------------------------------+
  | Layer 3       | Layer  header fields:                     |
  |               |           protocol                        |
  | IPv4 Header   |           port                            |
  |               |           src port                        |
  |               |           dscp                            |
  |               |           length                          |
  |               |           flags                           |
  |               |           ttl                             |
  |               |                                           |
  | IPv6 Header   |                                           |
  |               |           addr                            |
  |               |           protocol/nh                     |
```

```
     |               |            src port                       |
     |               |            length                         |
     |               |            traffic class                  |
     |               |            hop limit                      |
     |               |            flow label                     |
     |               |                                           |
     | 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


     +-----------------------------------------------------------+
     |      Object (context) matching Capability Index           |
     +---------------+-------------------------------------------+
     | Session       |   Session state,                          |
```

```
     |                |   bidirectional state                          |
```

Time	time span days, minutes, seconds, Events
Events	Event URL, variables

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

6. Structure of Rules to NSFs

6.1. Capability Layer Rules and Monitoring

The Capability Layer is to express the explicit rules to individual NSFs on how to treat packets and 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 List 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. Most vendors' IPS/IDS, and HoneyPot have their proprietary functions/profiles. One of the goals of I2NSF is to have common envelop format for exchanging or sharing profiles among different organizations to achieve more effective protection against threats.

The "subject" 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.

The I2NSF Capability Layer has to specify the "Object" (i.e. the states/contexts surrounding the packets).

The I2NSF "actions" should extend the actions specified by [ACL-MODEL] to include applying statistics functions that clients provide.

The rules for Flow-Based NSF can be extended from the Policy Core Information Model [RFC3060] and Policy Core Information Model Extension [RFC3460] which are the bases for ITU-T X.1036 [ITU-T-X1036], as shown below:

```
             +-------------------------+
             | Capability Layer Rules  |
             +-------------------------+
               |    |
     +---------+     +--------+              +---------+ |- Pass
     |Compound |     |        |              |  Simple +-|- Deny
     |Condition|     | action |           +--+  Actions| |- Mirror
     +----+----+     +----+---+           |  +---------+ |- Count
          |<-------+      +---------------+              |- client func
      +---+------+ |                      |
     ++---+-----+| |                      |  +---------+
     | simple   || |Compound Operators:   +--+ function|
     |conditions|+ |  Logical AND: &&        | Profile |
     +--+-----+-+  |  Logical OR: ||         +---------+
        |     |    |  Logical NOT: !
        |     +----+
        +-----------+
 +------+--+     +--+-----+                  +---------+
 | Subject |     | Object |                  |  Time   |
 |  Match  |     | Match  |               +--+---------+
 +-----+---+     +----+---+               |  +---------+
       |              +-------------------+--+  States |
       |                                  |  +---------+
    +--+----------+----------+            |  +---------+
 +--+----+     +--+---+   +--+---+        +--+ Port    |
 |IPv4   |     |IPv6  |   | MAC  |        |  +---------+
 |Header |     |Header|   |Header|        |      *
 +-------+     +------+   +------+        +--+   *
```

Figure 3: Structure of Capability Layer Rules

Capability layer also includes the policy monitoring of the individual NSFs and fault management of the policy execution. In NFV environment, policy consistency among multiple security function instances is very critical because security policies are no longer maintained by one central security devices, but instead are enforced by multiple security functions instantiated at various locations.

6.2. Service Layer Policy

This layer is for clients, applications or Application Controllers to express & 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. Usually these customers are expressing expectations (that can be viewed as loose security requirements). Customers may also express guidelines such as which critical communications are to be preserved during critical events, which hosts are to service even during severe security attacks, etc. As the result, there could be many depths or layers of Service Layer policies. Here are some examples of more abstract service layer security Policies:

- o Pass for Subscriber "xxx" with Port "y"
- o enable basic parental control
- o enable "school protection control"
- o allow Internet traffic from 8:30 to 20:00 [time = 8:30-20:00]
- o scan email for malware detection [check type = malware] protect traffic to corporate network with integrity and confidentiality [protection type = 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 users or Application Gateway when user's service requirements have been changed. [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 I2NSF capability layer, however with more client oriented expression for the subject, object, action, and function.

There have been several industry initiatives to address network policies, such as IETF Policy Core Information Model-PCIM [RFC3060, RFC3460], OpenStack's Group-based Policy (GBP), and others. Since I2NSF is not to tackle the general network service policies, but instead I2NSF is to define a standard interface for clients/applications to inform the Flow-based NSFs on the rules for treating traffic traversing through, it is overkill to inherent the entire policy structures designed for various network services.

However, the notion of Groups (or roles), Targets, Contexts (or conditions), and actions do cover what are needed for clients/applications to express the rules on how their flows to 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 Subject-Object-Action-Function" paradigm.

I2NSF can use PCIM (RFC3060 which the ITU-T X.1036 was based on) as a starting point. However, RFC3060 was created for general network policies, in some aspects more than what I2NSF needs, and in other aspects needs extension. Especially need extension on the Policy Context or condition (i.e. the directions, the time, and other contextual events that govern the policies to NSFs).

The I2NSF simple service layer can have the following entities:

- Composite Groups or Roles (I2NSF-Role): This is a group of users, applications, virtual networks, or traffic patterns to which a service layer policy can be applied. An I2NSF-Role 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-Role can have one or more Policy Rule Sets.
- Target. This is used by the application client to establish communications over the network. A Target is mapped to a physical/logical ingress port, a set of destinations, or a physical/logical egress port.
- Policy Rule Set. A Policy Rule Set is used to determine how the traffic between a pair of I2NSF-Role and Target is to be treated. A Policy Rule Set consists of one or more Policy Rules.
- Policy Rule. A Policy Rule consists of a Policy Conditions and a set of Actions to be applied to the traffic.

- Policy Condition. Describes when a Policy Rule set is 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. 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.

```
     +---------+     +--------+      +-------+ |- Logical Port
     | CTG     |---->| Policy |<-----+Target +-|- Ingress Port
     |         |     |Role Set|      |       | |- Egress Port
     +----+----+     +----+---+      +-------+ |-      *
          |<-------+      +---------------+
       +--+------+ |                      |    +--------+Logical
     +/---+-----+| |                      |  +/-------+ |Combination:
     | Simple   || |Compound Operators:   +--+ Policy | | AND/OR/NOT
     | Group    |+ |  Logical AND: &&        | Rule   | +
     +--+-----+-/  |  Logical OR: ||         +-+----+-/
        |     |    |  Logical NOT: !          /      \
        |     +----+                         +------+  +----------+
        |                                    |Action| -| Condition|
        +----------+--------------+--     +---+--+  +--+-------+
 +------+-+     +--+-----+     +---+-----+     |        |-Direction
 | App    |     |virtual |     | Subnet  |     |        |-timer
 | Group  |     | Subnet |     |host list|     |        |-L4 port
 ++-------+--+  +----+---+     +----+----+     |        |-Protocol
  |Client Grp|       |              |          |        |- *
  +----------+       |              |          |
    +-------------+--+------+-------+---       |
 +--+----+     +--+---+   +--+---+             |-Allow
 |IPv4   |     |IPv6  |   | MAC  |             |-Deny
 |Header |     |Header|   |Header|             |-count
 +-------+     +------+   +------+             |-apply function list
                                               |-   *
```

Figure 4: Rule Structure for Simple Service Layer

7. Capability Negotiation

When a NSF can't perform the desired provisioning due to resource constraint, it has to 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 (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 Gateway:

    - For example, Video Conference Mgr/Controller needs to dynamically inform network to allow or deny flows (some of which are encrypted) based 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

- 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 vNSFs
- 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.

[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

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