

I2NSF
Internet-Draft
Intended status: Experimental
Expires: January 9, 2017

R. Marin-Lopez
G. Lopez-Millan
University of Murcia
July 8, 2016

Software-Defined Networking (SDN)-based IPsec Flow Protection
draft-abad-i2nsf-sdn-ipsec-flow-protection-00

Abstract

This document describes the use case of providing IPsec-based flow protection by means of a Software-Defined Network (SDN) controller and raises the requirements to support this service. It considers two main scenarios: (i) gateway-to-gateway and (ii) host-to-gateway (Road Warrior). For the gateway-to-gateway scenario, this document describes a mechanism to support the distribution of IPsec information to flow-based Network Security Functions (NSFs) that implements IPsec to protect data traffic between network resources to protect data traffic with IPsec and IKE, in intra and inter-SDN cases. The host-to-gateway case defines a mechanism to distribute IPsec information to the NSF to protect data with IPsec between an end user's device (host) and a gateway.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 9, 2017.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

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

Table of Contents

1. Introduction	2
2. Requirements Language	4
3. Terminology	4
4. Objectives	5
5. Case 1: IKE/IPsec in the NSF	5
5.1. Requirements	6
6. Case 2: IPsec (no IKE) in the NSF	7
6.1. Requirements	7
7. Abstract interfaces	8
8. Data model	10
9. Use cases examples	12
9.1. Gateway-to-gateway under the same controller	12
9.2. Gateway-to-gateway under different SDN controllers	15
9.3. Host-to-gateway	17
10. Security Considerations	19
11. Acknowledgements	19
12. References	19
12.1. Normative References	19
12.2. Informative References	20
Authors' Addresses	21

1. Introduction

Software-Defined Networking (SDN) is an architecture that enables users to directly program, orchestrate, control and manage network resources through software. SDN paradigm relocates the control of network resources to a dedicated network element, namely SDN controller. The SDN controller manages and configures the distributed network resources and provides an abstracted view of the network resources to the SDN applications. The SDN application can customize and automate the operations (including management) of the abstracted network resources in a programmable manner via this interface [RFC7149][ITU-T.Y.3300][ONF-SDN-Architecture][ONF-OpenFlow].

Typically, traditional IPsec VPN concentrators and, in general, gateways supporting IKE/IPsec, are configured manually. This makes the IPsec security association (SA) management difficult and

generates a lack of flexibility, specially if the number of security policies and SAs to handle is high. With the grow of SDN-based scenarios where network resources are deployed in an autonomous manner, a mechanism to manage IPsec SAs according to the SDN architecture becomes more relevant. Thus, the SDN-based service described in this document will autonomously deal with IPsec-based data protection also in such as an autonomous manner.

IPsec architecture [RFC4301] defines a clear separation between the processing to provide security services to IP packets and the key management procedures to establish the IPsec security association. In this document, we defined that a service where the key management procedures can be carried by an external entity: the security controller.

First, this document exposes the requirements to support the protection of data flows using IPsec [RFC4301]. We consider two cases:

- 1) The network resource (or Network Security Function, NSF) implements the Internet Key Exchange (IKE) protocol and the IPsec databases: the Security Policy Database (SPD), the Security Association Database (SAD) and the Peer Authorization Database (PAD). The controller is in charge of provisioning the NSF with the required information about IKE, the SPD and the PAD.
- 2) The NSF only implements the IPsec databases (no IKE implementation). The controller will provide the required parameters to create valid entries in the PAD, the SPD and the SAD in the NSF. Therefore, the NSF will have only support for IPsec while automated key management functionality is moved to the controller.

In both cases, an interface/protocol will be required to carry out this provisioning between the security controller and the NSF. In particular, it is required the provision of SPD and PAD entries and the credentials and information related with the IKE negotiation (case 1); or the required SPD, PAD and SAD entries with information such as keys, cryptographic algorithms, IP addresses, IPsec protocol (AH or ESP), IPsec protocol mode (tunnel or transport), lifetime of the SA, etc (case 2). An example for case 1 using NETFCONF/YANG can be found in [netconf-vpn]. A YANG model for IPsec can be found in [I-D.tran-ipsecme-yang].

Second, this document considers two scenarios to manage autonomously IPsec SAs: gateway-to-gateway and host-to-gateway [RFC6071]. The gateway-to-gateway scenario shows how flow protection services are useful when data is to be protected across gateways in the network.

Each gateway will implement a flow-based NSF. The use case described in Section 9.1 depicts how these services could be used to protect IP traffic among various geographically distributed networks under the domain of the same security controller. A variant of this scenario is also covered in Section 9.2, where the NSFs involved are under the control of different security controllers.

The host-to-gateway scenario described in Section 9.3 covers the case where one end user belonging to a network wants to access securely its network from another external network. In such a case, an IPsec SA needs to be established between the end user's host and the gateway, which is a flow-based NSF. In this document, we describe how the security controller can still configure automatically the IPsec SA in the NSF.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119]. When these words appear in lower case, they have their natural language meaning.

3. Terminology

This document uses the terminology described in [RFC7149], [RFC4301], [ITU-T.Y.3300], [ONF-SDN-Architecture], [ONF-OpenFlow], [ITU-T.X.1252], [ITU-T.X.800] and [I-D.ietf-i2nsf-terminology]. In addition, the following terms are defined below:

- o Software-Defined Networking. A set of techniques enabling to directly program, orchestrate, control, and manage network resources, which facilitates the design, delivery and operation of network services in a dynamic and scalable manner [ITU-T.Y.3300].
- o Flow/Data Flow. Set of network packets sharing a set of characteristics, for example IP dst/src values or QoS parameters.
- o Flow Protection Policy. The set of rules defining the conditions under which a data flow MUST be protected with IPsec, and the rules that MUST be applied to the specific flow.
- o IKE. Protocol to establish IPsec Security Associations (SAs). It requires information about the required authentication method (i.e. preshared keys), DH groups, modes and algorithms for IKE phase 1, etc.

- o SPD. IPsec Security Policy Database. It includes information about IPsec policies direction (in, out), local and remote addresses, inbound and outbound SAs, etc.
- o SAD. IPsec Security Associations Database. It includes information about IPsec security associations, such as SPI, destination addresses, authentication and encryption algorithms and keys.
- o PAD. Peer Authorization Database. It provides the link between the SPD and a security association management protocol such as IKE or our SDN-based solution.

4. Objectives

- o Flow-based data protection: controller-based flow protection services based on IPsec to allow the protection of specific data flows based on defined security policies.
- o Establishment and management of IPsec security associations: this service allows the centralized management of IPsec SAs to protect specific data flows.

5. Case 1: IKE/IPsec in the NSF

In this case, the security controller is in charge of controlling and applying SPD and PAD entries in the NSF. It also has to apply IKE configuration parameters and derive and deliver IKE credentials (e.g. a pre-shared key) to the NSF for the IKE negotiation. In short, we would call this IKE credential.

With these entries and credentials, the IKE implementation can operate to establish the IPsec SAs. The application (administrator) will send the IPsec requirements and end points information, and the security controller will translate those requirements into SPD entries that will be installed in the NSF. With that information provisioned in the NSF, when the data flow needs to be protected, the NSF can just run IKE to establish the required IPsec SA. Figure 1 shows the different layers and corresponding functionality.

Advantages: It is simple because current gateways typically have an IKE/IPsec implementation.

Disadvantages: IKE implementations need to renegotiate IPsec SAs upon SPD entries changes without restarting IKE daemon.

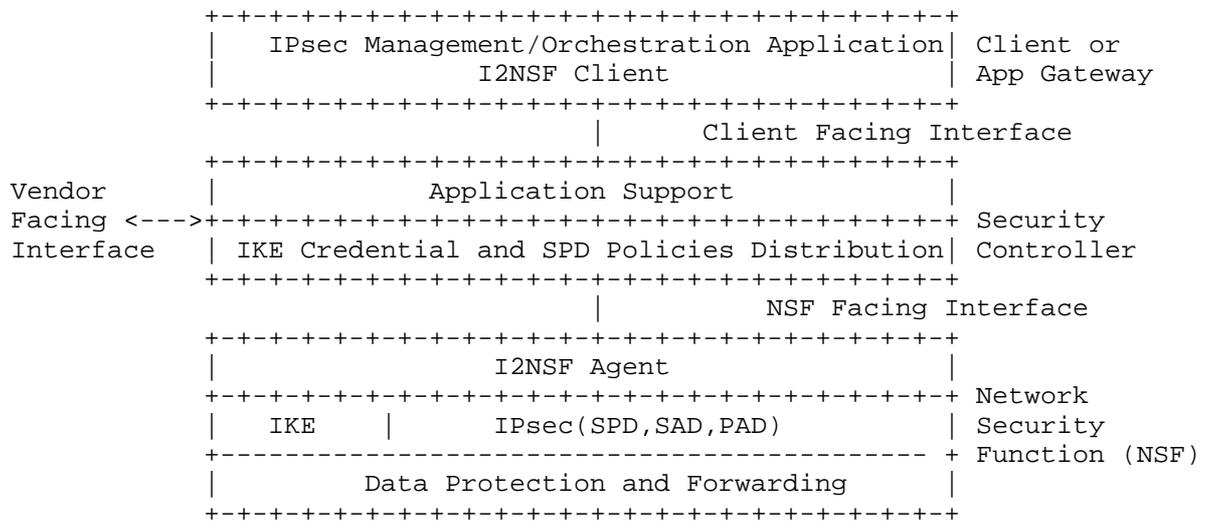


Figure 1: Case 1: IKE/IPsec in the NSF

5.1. Requirements

SDN-based IPsec flow protection services provide dynamic and flexible network resource management to protect data flows among network resources and end users. In order to support this capability in case 1, the following requirements are to be met:

- o The NSF MUST implement IKE and IPsec databases: SPD, SAD and PAD. It MUST provide an (southbound) interface to provision SPD and PAD entries, IKE Credentials and to monitor the IPsec databases and IKE implementation. Note that SAD entries are created in runtime by IKE.
- o A southbound protocol MUST support sending these SPD and PAD entries, and IKE credentials to the NSF.
- o It requires an (northbound) application interface in the security controller allowing the management of IPsec SAs.
- o In scenarios where multiple controllers are implicated, SDN-based flow protection service may require a mechanism to discover which security controller is managing a specific NSF.

6. Case 2: IPsec (no IKE) in the NSF

This section describes the referenced architecture to support SDN-based IPsec flow protection where the security controller performs automated key management tasks.

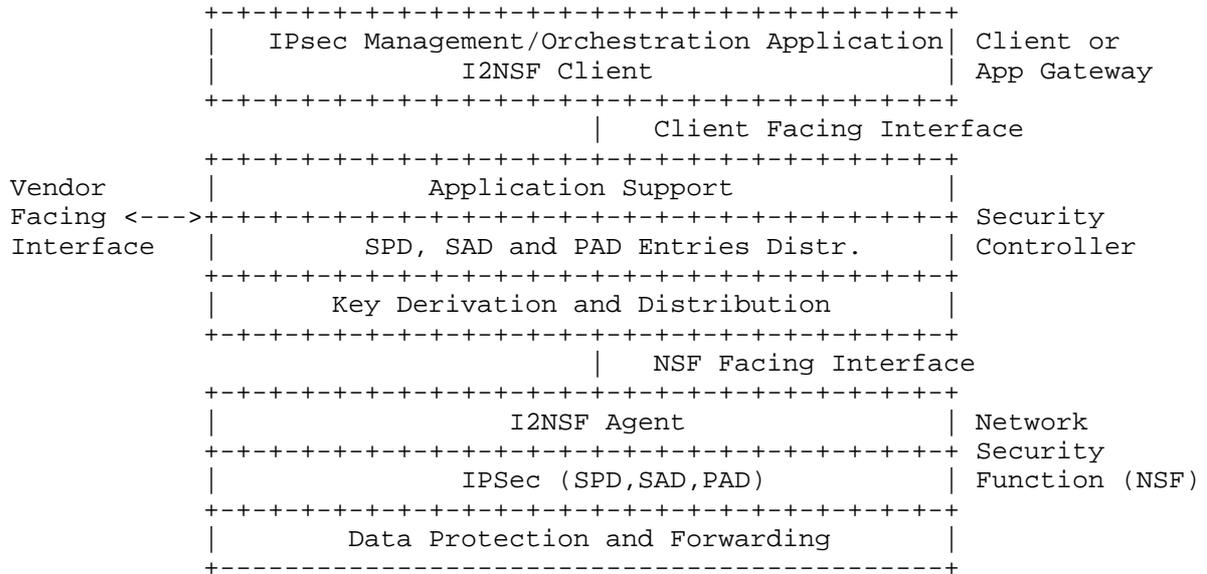


Figure 2: Case 2: IPsec (no IKE) in the NSF

As shown in Figure 2, applications for flow protection run on the top of the security controller. When an administrator enforces flow protection policies through an application interface, the security controller translates those requirements into SPD,PAD and SAD entries that will be installed in the NSF.

Advantages: 1) It allows lighter NSFs (no IKE implementation). 2) IKE does not need to be run in gateway-to-gateway scenario with a single controller (see Section 9.1).

Disadvantages: The overload of IPsec SA establishment is shifted to the security controller since IKE is not required in the NSF.

6.1. Requirements

In order to support case 2, the following requirements are to be met:

- o It requires the provision of SPD, PAD and SAD entries into the NSF. A southbound protocol MUST support sending this information to the NSF.
- o NSF MUST be capable to protect data flows with IPsec, such as the capability to forward data through an IPsec tunnel.
- o It requires an (northbound) application interface in the security controller allowing the management of IPsec policies.
- o In scenarios where multiple controllers are implicated, SDN-based flow protection service may require a mechanism to discover which security controller is managing a specific NSF.

7. Abstract interfaces

The cases presented above require an analysis of the communication channel between the IPsec stack and the security controller that is performing the key management operations.

The IETF RFC 2367 (PF_KEYv2) [RFC2367] provides a generic key management API that can be used not only for IPsec but also for other network security services to manage the IPsec SAD. Besides, as an extension to this API, the document [I-D.pfkey-spd] specifies some PF_KEY extensions to maintain the SPD. This API is accessed using sockets.

An I2NSF Agent implementation in the NSF can interact with both APIs in a kernel and returns and provides the same information using the NSF Facing Interface. In the following, we show a summary of these messages just to show an example of what may provide the NSF Facing Interface. The details and the accurate information is in RFC 2367 and [I-D.pfkey-spd].

To manage the IPsec SAD we have the following messages in the PF_KEYv2 API:

- o The SADB_GETSPI message allows a process to obtain a unique SPI value for given security association type, source address, and destination address. This message followed by an SADB_UPDATE is one way to create a security association (SADB_ADD is the other method).
- o The SADB_UPDATE message allows a process to update the information in an existing Security Association.

- o The SADB_ADD message is nearly identical to the SADB_UPDATE message, except that it does not require a previous call to SADB_GETSPI.
- o The SADB_DELETE message causes the kernel to delete an IPsec SA from the SAD.
- o The SADB_GET message allows a process to retrieve a copy of a Security Association from the SAD.
- o The SADB_ACQUIRE message is typically triggered by an outbound packet that needs security but for which there is no applicable IPsec SA existing in the SAD.
- o The SADB_REGISTER message allows (a socket) to receive SADB_ACQUIRE messages for the type of IPsec SA.
- o The SADB_EXPIRE message is issued when soft limit or hard limit (lifetime) of a IPsec SA has expired.
- o The SADB_FLUSH message causes the kernel to delete all entries in its IPsec SAD.
- o The SADB_DUMP message causes to dump the operating system's entire IPsec SAD.

Although it is not a standard, KAME IPsec has defined a set of extensions to PF_KEY in order to handle the SPD [I-D.pfkey-spd]. The extended API offers the additional extensions:

- o The SADB_X_SPDSETIDX message allows a process to add only selector of the security policy entry to the SPD.
- o The SADB_X_SPDUPDATE message replaces the parameters of an existing SPD entry.
- o The SADB_X_SPDADD is message allows a process to add a new security policy entry to the SPD.
- o The SADB_X_SPDDELETE message causes the kernel to delete an entry from the SPD.
- o The SADB_X_SPDDELETE2 message nearly identical to the SADB_X_SPDDELETE message, except that it specifies the policy id.
- o The SADB_X_SPDGET message is allows a process to retrieve a copy of a security policy entry from the SPD.

- o The SADB_X_SPDACQUIRE message is triggered by an outbound packet that needs security policy but for which there is no applicable information existing in the SPD.
- o The SADB_X_SPDEXPIRE message is issued when limit of a security policy (SPD entry) has expired.
- o The SADB_X_SPDFLUSH message causes the kernel to delete all entries in the IPsec SPD.
- o The SADB_DUMP causes the kernel to dump all entries in the IPsec SPD.

Regarding PAD management, we have not found any related extension. However, from the abstract data model defined in Section 8 for the PAD an interface could be designed.

8. Data model

These cases assume a data model representing the information to be exchanged between controller and network resource through the southbound interface. As described before this data model has to include the following information [RFC4301] (sketch that needs to be developed):

Data model for the SDP entries:

- o Name
- o PFP flags
- o Perfect forward secrecy
- o Selector list:
 - Remote IP addresses(es)
 - Local IP addresses(es)
 - Flow direction
 - Next Layer Protocol
 - Local port
 - Remote port
 - Type code

- o Processing:
 - Extended sequence number
 - Sequence overflow
 - Fragment checking
 - IP compression
 - DF bit
 - DSCP
 - IPsec protocol (AH/ESP)
 - Algorithms
 - Manual SPI
 - Local tunnel endpoint
 - Remote tunnel endpoint
 - Tunnel options

Data model for the SAD entries:

- o SPI
- o Local peer
- o Remote peer
- o SA mode (tunnel or transport)
- o Security protocol
- o Sequence number options
- o Life-time
- o Upper protocol
- o Direction
- o Tunnel source IP address and port

- o Tunnel Destination IP address and port
- o AH parameters
- o ESP parameters
- o IP compression
- o NAT traversal flag
- o Path MTU
- o Anti-replay window

Data model for the PAD entries:

- o Identifies the peers or groups of peers that are authorized to communicate with this IPsec entity.
- o The protocol and method used to authenticate each peer.
- o Authentication data for each peer.
- o Constraints about the types and values of IDs that can be asserted by a peer with regard to child SA creation.
- o Peer gateway location info (e.g., IP address(es) or DNS names).

Data model for the IKE configuration:

- o TBD. (NOTE: It may depend on the IKE version)

9. Use cases examples

This section explains three use cases as examples for the SDN-based IPsec Flow Protection Service.

9.1. Gateway-to-gateway under the same controller

Enterprise A has a headquarter office (HQ) and several branch offices (BO) interconnected through an Internet connection provided by an Internet Service Provider (ISP). This ISP has deployed a SDN-based architecture to provide connectivity to all its clients, including HQ and BOs, so the HQ is provided with a gateway that acts as a router between Internet and each BO's internal network. The gateway implements our Flow-based NSF.

Now, Enterprise A requires that certain traffic between the HQ and BOs MUST be protected, for example, with confidentiality and integrity. The Enterprise A's administrator has to configure flow protection policies in the ISP's security controller, determining that the traffic among Enterprise A's HQ (HQ A) and each BO MUST be protected.

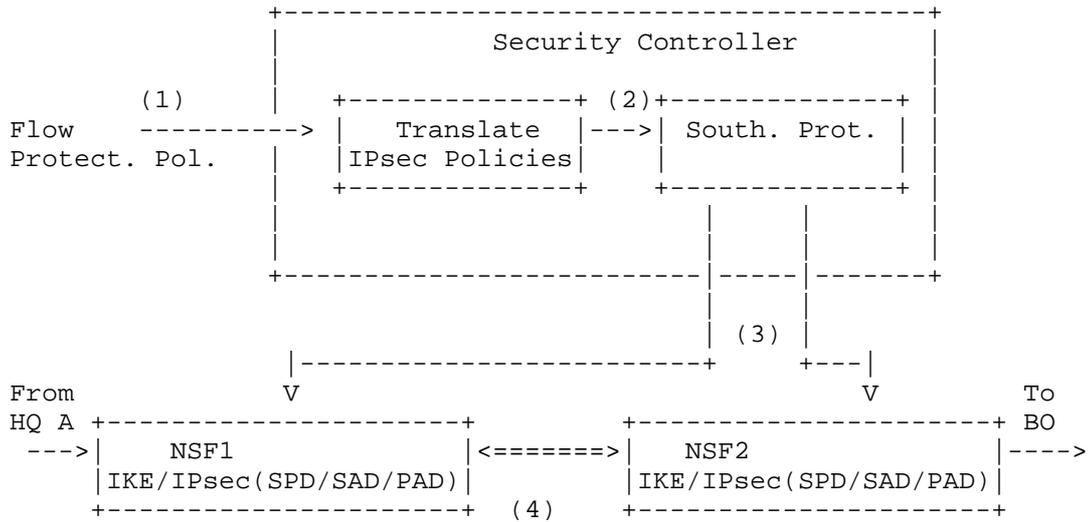


Figure 3: Gateway-to-Gateway single controller flow for case 1 .

Figure 3 describes the case 1:

1. The administrator establishes general Flow Protection Policies.
2. The controller generates IKE credentials and translates the policies into SPD and PAD entries.
3. The controller looks for the NSFs involved (NSF1 and NSF2) and inserts the SPD and PAD entries in both NSF1 and NSF2.
4. All packets belonging to the flow that matches the IPsec SPD inserted by the security controller will trigger the IKE negotiation in NSF1 and NSF2 by using the IKE credentials.

In case 2, Flow Protection Policies defined by the administrator are also translated into IPsec SPD entries and inserted into the corresponding NSFs. Besides, SAD entries will be also defined by the controller and enforced in the NSFs. In this case the execution of IKE is not necessary in the controller, and a Key Derivation function can be used to provide the required cryptographic material for the

IPsec SAs. These keys will be also distributed through the southbound interface. Note that it is possible because both NSF's are managed by the same controller.

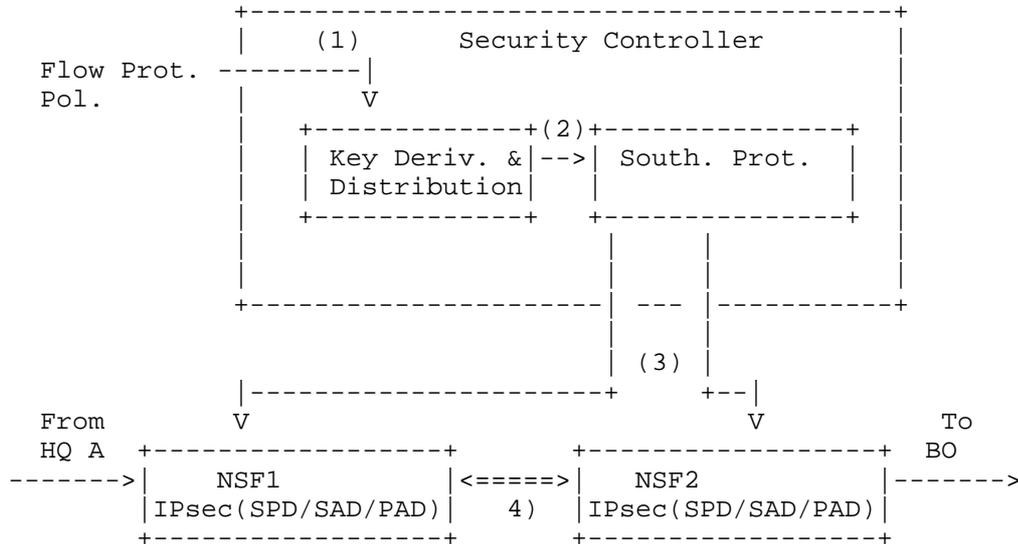


Figure 4: Gateway-to-Gateway single controller flow for case 2.

Figure 4 describes the case 2, when a data packet is sent from HQ A with destination BO :

1. The administrator establishes Flow Protection Policies.
2. The controller translates these policies into IPsec SPD, PAD and SAD entries.
3. The controller looks for the NSF's involved and inserts the these entries in both NSF1 and NSF2 IPsec databases.
4. All packets belonging to the flow are tunneled between NSF1 and NSF2 by using the enforced configuration keys and parameters. No need to run IKE between NSF1 and NSF2.

In general (for case 1 and case 2), this system presents various advantages to the ISP: (i) it allows to create a IPsec SA among two NSF's, with only the application of specific security policies at the application layer. Thus, the ISP can manage all security associations in a centralized point and with an abstracted view of the network; (ii) All NSF's deployed after the application of the new

policies will NOT need to be manually configured, thus allowing its deployment in an automated manner.

9.2. Gateway-to-gateway under different SDN controllers

Two organizations, Enterprise A and Enterprise B, have its headquarters interconnected through an Internet connection provided by different ISPs, called ISP_A and ISP_B. They have deployed a SDN-based architecture to provide Internet connectivity to all its clients, so Enterprise A's headquarters is provisioned with a gateway deployed by ISP_A and Enterprise B's headquarters is provisioned with a gateway deployed by ISP_B.

Now, these organizations require that certain traffic among its headquarters to be protected with confidentiality and integrity, so the ISPs have to configure Flow Protection Policies in their security controllers. Both administrators define Flow Protection Policies in each Security Controller that will end with the translation into SPD and PAD entries and IKE credentials in each NSF so that the specified traffic exchanged among these headquarters will be protected.

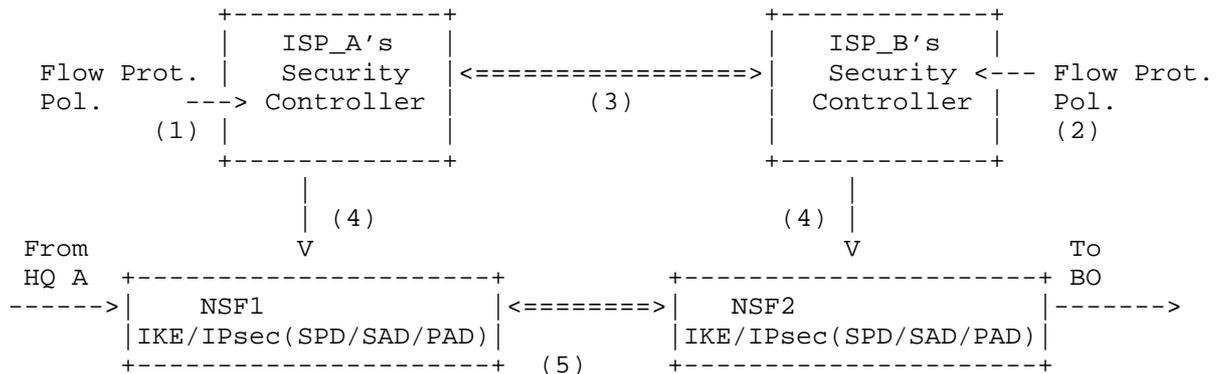


Figure 5: Gateway-to-gateway multi controller flow in case 1

On the one hand, case 1, Figure 5 describes the data and control plane communications required when a data packet is sent from Enterprise A's HQ (HQ A) to destination Enterprise B's HQ (HQ B):

1. The administrator A establishes general Flow Protection Policies in ISP_A's Security Controller
2. The administrator B establishes general Flow Protection Policies in ISP_B's Security Controller

3. The ISP_A's security controller realizes that protection is between the NSF1 and NSF2, which is under the control of another security controller (ISP_B's security controller), so it starts negotiations with the other controller to agree on the IPsec SPD policies and IKE credentials for their respective NSFs. NOTE: This may require extensions in the East/West interface.
4. Then, both security controllers enforce the IKE credentials and related parameters and the SPD and PAD entries in their respective NSFs.
5. All packets belonging to the flow that matches the IPsec SPD inserted by the security controller triggers the IKE negotiation between NSF1 and NSF2 by using the enforced configuration keys and parameters.

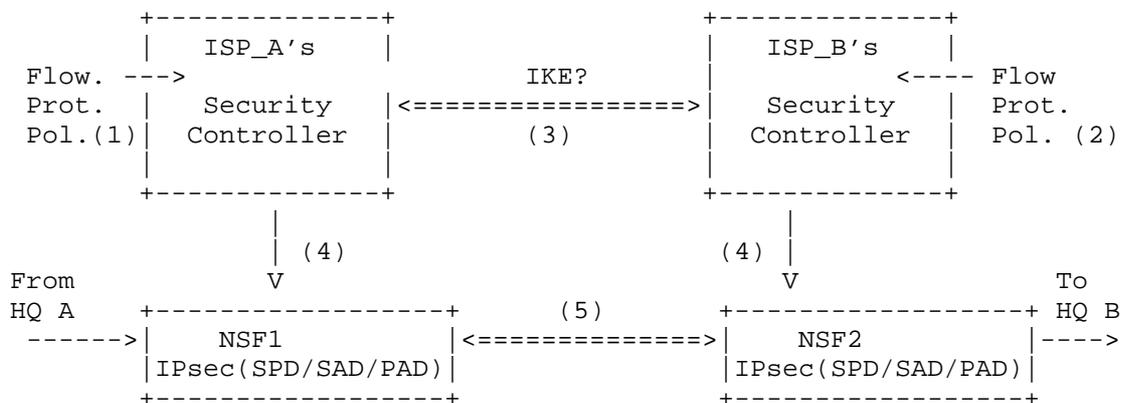


Figure 6: Gateway-to-gateway multi controller flow in case 2

On the other hand, case 2, Figure 6 describes the data and control plane communications required when a data packet is sent from Enterprise A's HQ (HQ A) to destination Enterprise B's HQ (HQ B):

1. The administrator A establishes general Flow Protection Policies in ISP_A's Security Controller
2. The administrator B establishes general Flow Protection Policies in ISP_B's Security Controller
3. The ISP_A's security controller realizes that traffic between NSF1 and NSF2 MUST be protected. Nevertheless, the controller notices that NSF2 is under the control of another security controller, so it starts negotiations with the other controller to agree on the IPsec SPD, PAD, SAD entries that define the IPsec

SAs. NOTE: It would worth evaluating IKE as the protocol for the the East/West interface in this case.

4. Once the controllers have agreed on key material and the details of the IPsec SA, they both enforce this information into their respective NSFs.
5. Therefore, all packets belonging to the flow are protected between NSF1 and NSF2 by using the enforced configuration keys and parameters.

In general (case 1 and case 2), this system presents various advantages to both ISPs: (i) it allows to create a security association among two network resources across ISPs, from each ISP point of view, only the application of specific Flow Protection Policies at the application layer is needed, so they can manage all security associations in a centralized point and with an abstracted view of the network; (ii) All new resources deployed after the application of the new policies will not need to be manually configured, thus allowing its deployment in an automated manner.

9.3. Host-to-gateway

End user is a member of Enterprise A who needs to connect to the HQ's internal network. Enterprise A has deployed a NSF acting as IPsec-based VPN concentrator in its HQ to allow members of the organization to connect to the HQ's internal network in a secure manner.

Traditionally, VPN concentrators are built as appliances, configured manually to authenticate and establish secure associations with incoming end users users, for example, by running IKE to establish an IPsec tunnel. With the SDN-based management of IPsec we can automatize these configurations.

In case 1, as we can see in Figure 7, the administrator configures a Flow Protection Policy in the security controller (1). The controller generates IKE credentials and translates that into SPD and PAD entries and installs them in the corresponding NSF (2). With those policies and IKE credentials, end user and gateway can negotiate IKE.

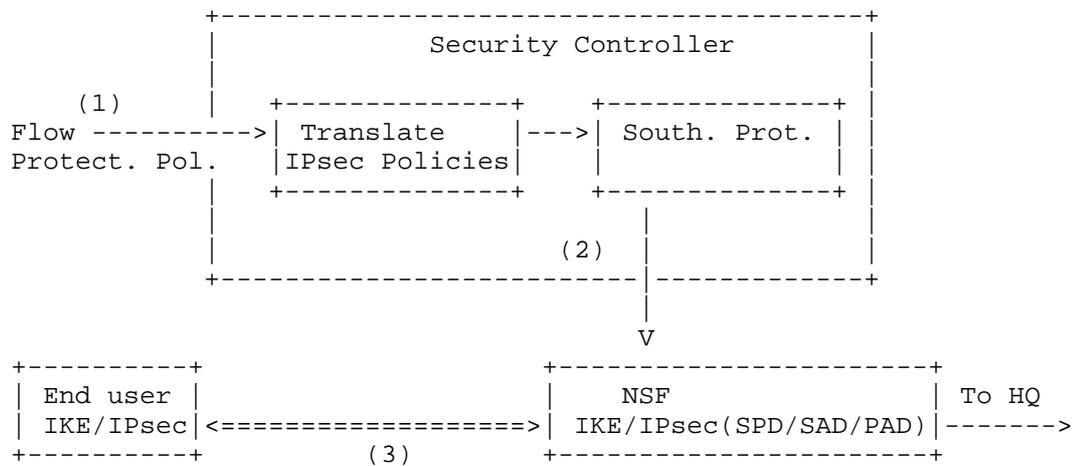


Figure 7: Host-to-gateway flow protection in case 1.

In case 2, IKE implementation now resides in the security controller, as we can see in Figure 8. Here, the NSF needs to forward IKE packets to the controller. Therefore, the IKE negotiation is performed by the end user and the security controller (1), being this fact completely transparent for the end user.

Once the IKE negotiation has been successfully completed, the IPsec SA is available in the end user and in the security controller. The IPsec SA information is to be provisioned into the NSF's SAD, SPD and PAD (2). Now the end user and the NSF share key material, thus being able to establish an IPsec tunnel to protect all traffic among them (3).

In general, this feature allows the configuration of network resources such as VPN concentrators as a service, so these could be deployed and disposed as required by policies, such as network load, in an autonomous manner.

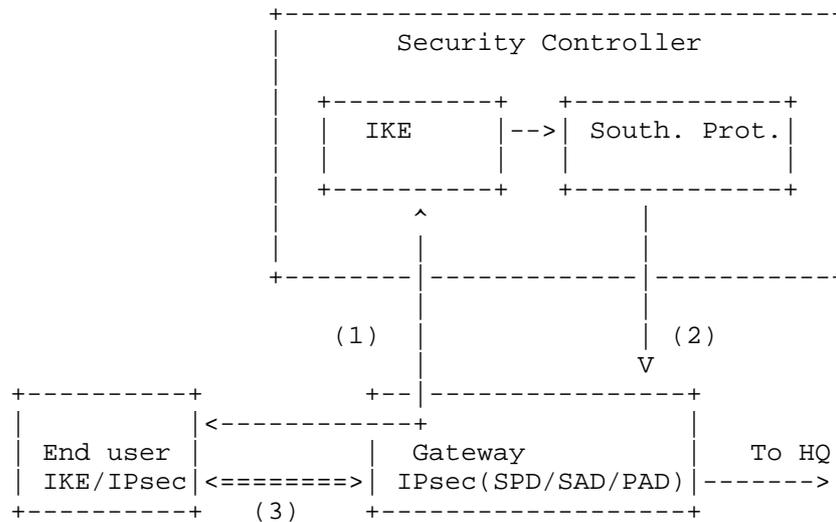


Figure 8: Host-to-gateway flow protection in case 2.

One of the main problems of this scenario is that the security controller has to implement IKE and negotiate with the end user. Additionally, it is still unclear the security implications of performing IKE with a different end point than the NSF. Finally, in terms of implementation, the IKE packets should bypass IPsec protection in the NSF and be forwarded to the security controller.

10. Security Considerations

TBD.

11. Acknowledgements

Authors want to thank Sowmini Varadhan, Linda Dunbar, Carlos J. Bernardos, Alejandro Perez-Mendez and Alejandro Abad-Carrascosa for their valuable comments.

12. References

12.1. 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>>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <<http://www.rfc-editor.org/info/rfc5226>>.

12.2. Informative References

- [I-D.ietf-i2nsf-framework]
elopez@fortinet.com, e., Lopez, D., Dunbar, L., Strassner, J., Zhuang, X., Parrott, J., Krishnan, R., and S. Durbha, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-02 (work in progress), July 2016.
- [I-D.ietf-i2nsf-terminology]
Hares, S., Strassner, J., Lopez, D., and L. Xia, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-00 (work in progress), May 2016.
- [I-D.jeong-i2nsf-sdn-security-services-05]
Jeong, J., Kim, H., Park, J., Ahn, T., and S. Lee, "Software-Defined Networking Based Security Services using Interface to Network Security Functions", draft-jeong-i2nsf-sdn-security-services-05 (work in progress), July 2016.
- [I-D.pfkey-spd]
Sakane, S., "PF_KEY Extensions for IPsec Policy Management in KAME Stack", October 2002.
- [I-D.tran-ipsecme-yang]
Tran, K., Wang, H., Nagaraj, V., and X. Chen, "Yang Data Model for Internet Protocol Security (IPsec)", draft-tran-ipsecme-yang-01 (work in progress), June 2015.
- [ITU-T.X.1252]
"Baseline Identity Management Terms and Definitions", April 2010.
- [ITU-T.X.800]
"Security Architecture for Open Systems Interconnection for CCITT Applications", March 1991.
- [ITU-T.Y.3300]
"Recommendation ITU-T Y.3300", June 2014.

- [netconf-vpn]
Stefan Wallin, "Tutorial: NETCONF and YANG", January 2014.
- [ONF-OpenFlow]
ONF, "OpenFlow Switch Specification (Version 1.4.0)",
October 2013.
- [ONF-SDN-Architecture]
"SDN Architecture", June 2014.
- [RFC2367] McDonald, D., Metz, C., and B. Phan, "PF_KEY Key
Management API, Version 2", RFC 2367,
DOI 10.17487/RFC2367, July 1998,
<<http://www.rfc-editor.org/info/rfc2367>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the
Internet Protocol", RFC 4301, DOI 10.17487/RFC4301,
December 2005, <<http://www.rfc-editor.org/info/rfc4301>>.
- [RFC6071] Frankel, S. and S. Krishnan, "IP Security (IPsec) and
Internet Key Exchange (IKE) Document Roadmap", RFC 6071,
DOI 10.17487/RFC6071, February 2011,
<<http://www.rfc-editor.org/info/rfc6071>>.
- [RFC7149] Boucadair, M. and C. Jacquenet, "Software-Defined
Networking: A Perspective from within a Service Provider
Environment", RFC 7149, DOI 10.17487/RFC7149, March 2014,
<<http://www.rfc-editor.org/info/rfc7149>>.

Authors' Addresses

Rafa Marin-Lopez
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia 30100
Spain

Phone: +34 868 88 85 01
Email: rafa@um.es

Gabriel Lopez-Millan
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia 30100
Spain

Phone: +34 868 88 85 04
Email: gabilm@um.es

I2NSF
Internet-Draft
Intended status: Experimental
Expires: January 9, 2017

C. Basile
PoliTo
D. Lopez
TID
July 8, 2016

A Model of Security Capabilities for Network Security Functions
draft-baspez-i2nsf-capabilities-00

Abstract

This document presents a model of Security Capabilities. Security Capabilities are intended to describe the potentiality that Network Security Functions (NSFs) have for security policy enforcement purposes. Therefore, Security Capabilities are represented as abstract functionalities that a NSF owns in terms of enforcement actions, conditions that can apply in order to determine to which packet or traffic enforce the actions, and other mechanisms that NSF use to determine the actions to enforce. The proposed capability model defines without ambiguities the operations a function can do in term of security policy enforcement.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 9, 2017.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Requirements Language	3
3. Policy model	4
3.1. Geometric model of policies	4
3.2. Condition types	7
4. Capability Model	8
4.1. Algebra of capabilities	9
5. References	10
5.1. Normative References	10
5.2. Informative References	11
Authors' Addresses	11

1. Introduction

Security Capabilities are intended to describe the potentiality that Network Security Functions (NSFs) have for security policy enforcement purposes. Security Capabilities are abstract concepts that are independent of the actual security control that will implement them. However, every NSF will be associated to the capabilities it owns. Security Capabilities are required to allow interoperability among network functions. It would be a market enabler having a way to substitute a NSF with an equivalent one (i.e., having the same functionality). Moreover, Security Capabilities are very useful to reason about generic functions, which may be needed at design time. That is, it is not needed to refer to a specific product when designing the network, rather the functions characterized by their capabilities are considered.

Therefore, we have developed another model where Security Capabilities determine what a security control can do in terms of conditions, actions, resolution strategies, external data, if it supports default action, etc. That is, the Security Capabilities model defines without any ambiguity the things a function can do in term of security policy enforcement. The Security Capability model is built on a predefined general policy model. The type of policies that a NSF can enforce are obtained by customizing the general policy model with the Security Capability information.

The Capability Model has been designed to support at least capability matching, i.e., to identify the NSFs in a catalog that can perform the operations required to enforce a high-level policy.

Moreover, the Capability Model has been preliminarily validated by verifying that it allows the correct description of several existing security controls.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [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.

3. Policy model

The starting point of the design of our capability model is a simple observation. As human beings, we all understand immediately each other when we refer to security controls by just naming their category. For instance, experts agree on what is a NAT, a filtering control, or a VPN concentrator. Network security experts unequivocally refer to "packet filters" as stateless devices able to allow or deny packets forwarding based on conditions on source and destination IP addresses, source and destination ports, and IP protocol type fields [Alshaer]. Moreover, it is known that packet filter rules are prioritized and it is possible to specify a default action. More precisely, packet filters implement the First Matching Rule (FMR) or Last Matching Rule (LMR) resolution strategies.

However, we feel the need for more information in case of other devices, like stateful firewalls or application layer filters. These devices filter packets or communications, but there are differences among products in the packets and communications that they can categorize and the states they maintain. Analogous considerations can be applied for channel protection protocols, where we all understand that they will protect packets by means of symmetric algorithms whose keys could have been negotiated with asymmetric cryptography, but they may work at different layers and support different algorithms and protocols. To ensure protection, these protocols apply integrity, optionally confidentiality, apply anti-reply protections, and authenticate peers.

3.1. Geometric model of policies

We refer in this paper to the policy model defined in [Bas12] as geometric model, which is summarized here. Policies are specified by means of a set of rules in the "if condition then action" format [RFC3198]. Rules are formed by a condition clause and an action clause. All the actions available to the security function are well known and organized in an action set A.

For filtering controls, the enforceable actions are Allow and Deny, thus $A=\{\text{Allow,Deny}\}$. For channel protection controls, they may be informally written as "enforce confidentiality", "enforce data authentication and integrity", and "enforce confidentiality and data authentication and integrity". However, these actions need to be instantiated to the technology used, for instance AH-transport mode and ESP-transport mode (and combinations thereof) are a more precise and univocal definition of channel protection actions.

Conditions are typed predicates concerning a given selector. A selector describes the values that a protocol field may take, e.g.,

the IP source selector is the set of all possible IP addresses, and it may also refer to the part of the packet where the values come from, e.g., the IP source selector refers to the IP source field in the IP header. Geometrically, a condition is the subset of its selector for which it evaluates to true. A condition on a given selector matches a packet if the value of the field referred to by the selector belongs to the condition. For instance, in Figure 1 the conditions are $s1 \subseteq S1$ (read as $s1$ subset of $S1$) and $s2 \subseteq S2$ ($s1$ subset of $S2$), both $s1$ and $s2$ match the packet $x1$, while only $s2$ matches $x2$.

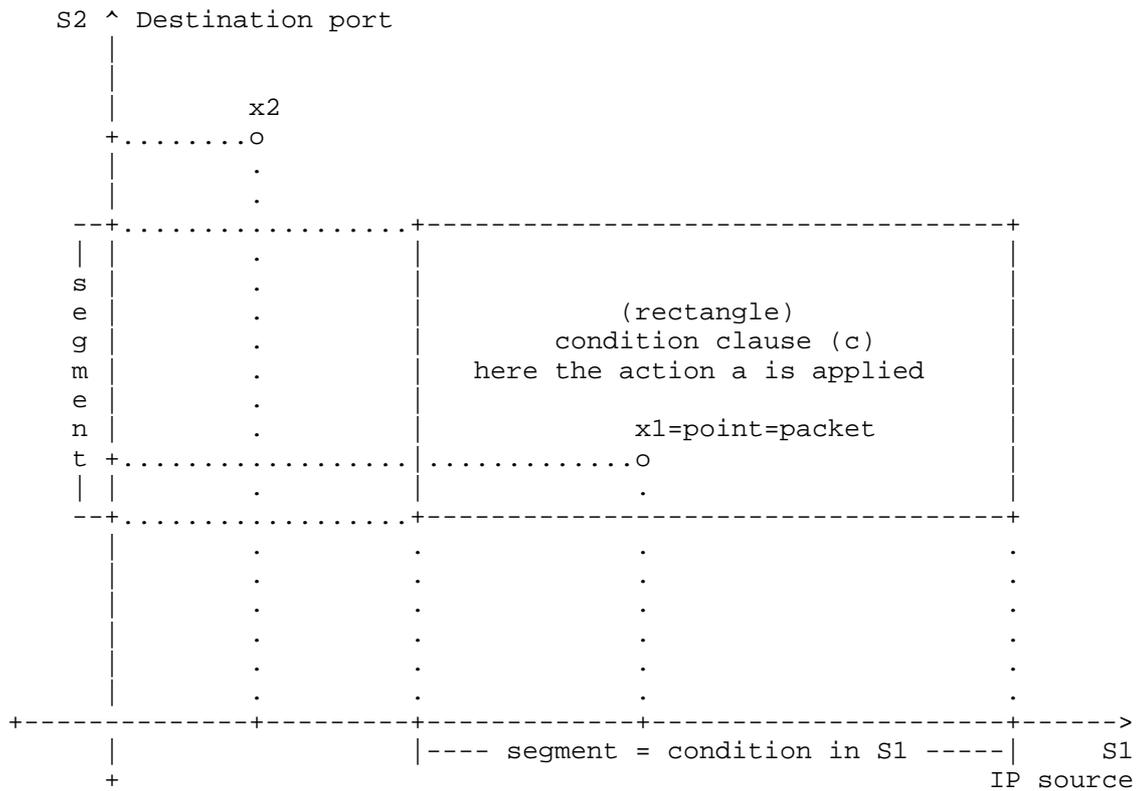


Figure 1: Geometric representation of a rule $r=(c,a)$ that matches $x1$ but does not match $x2$.

To consider conditions in different selectors, the decision space is extended using the Cartesian product because distinct selectors refer to different fields, possibly from different protocol headers. Hence, given a policy-enabled element that allows the definition of conditions on the selectors $S1, S2, \dots, Sm$ (where m is the number of

selectors available at the security control we want to model), its selection space is:

$$S = S_1 \times S_2 \times \dots \times S_m$$

To consider conditions in different selectors, the decision space is extended using the Cartesian product because distinct selectors refer to different fields, possibly from different protocol headers.

Accordingly, the condition clause c is a subset of S :

$$c = s_1 \times s_2 \times \dots \times s_m \quad [\quad S_1 \times S_2 \times \dots \times S_m = S$$

S represents the totality of the packets that are individually selectable by the security control to model when we use it to enforce a policy. Unfortunately, not all its subsets are valid condition clauses: only hyper-rectangles or union of hyper-rectangles (as they are Cartesian product of conditions) are valid. This is an intrinsic constraint of the policy languages as they specify rules by defining a condition for each selector. Languages that allow specification of conditions as relations over more fields are modelled by the geometric model as more complex geometric shapes determined by the equations. However, the algorithms to compute intersections are much more sophisticated than intersection hyper-rectangles. Figure 1 graphically represents a condition clause c in a two-dimensional selection space.

In the geometric model, a rule is expressed as $r=(c,a)$, where $c \in S$ (the condition clause is a subset of the selection space), and the action a belongs to A . A condition clause of a rule matches a packet, or briefly a rule matches a packet, if all the conditions forming the clause match the packet: in Figure 1, the rule with condition clause c matches the packet x_1 but not x_2 .

The rule set R is composed of n rules $r_i=(c_i,a_i)$.

The decision criteria for the action to apply when a packet matches two or more rules is abstracted by means of the resolution strategy $RS: Pow(R) \rightarrow A$, where $Pow(R)$ is the power set of rules in R .

Formally, given a set of rules, the resolution strategy maps all the possible subsets of rules to an action a in A . When no rule matches a packet, the security controls may select the default action d in A , if they support one.

Resolution strategies may use, besides intrinsic rule data (i.e., condition clause and action clause), also "external data" associated to each rule, such as priority, identity of the creator,

and creation time. Formally, every rule ri is associated by means of the function $e(.)$ to:

$$e(ri) = (ri, f1(ri), f2(ri), \dots)$$

where $E = \{fj:R \rightarrow Xj\}$ is the set that includes all the functions that map rules to external attributes in Xj . However, E , e , and all the Xj are determined by the resolution strategy used.

A policy is thus a function $p: S \rightarrow A$ that connects each point of the selection space to an action taken from the action set A according to the rules in R . By also assuming $RS(0)=d$ (where 0 is the empty-set) and $RS(ri)=ai$, the policy p can be described with this formula $p(x)=RS(\text{match}\{R(x)\})$.

Therefore, in the geometric model, a policy is completely defined by the 4-tuple (R, RS, E, d) : the rule set R , the resolution function RS , the set E of mappings to the external attributes, and the default action d .

Note that, the geometric model also supports ECA paradigms by simply modelling events like an additional selector.

3.2. Condition types

After having analysed the literature and the existing security controls, we have categorized the types of selectors in exact match, range-based, regex-based, and custom-match [Bas15][Lunt].

Exact match selectors are (unstructured) sets: elements can only be checked for equality, as no order is defined on them. As an example, the protocol type field of the IP header is a unordered set of integer values associated to protocols.

Range-based selectors are ordered sets where it is possible to naturally specify ranges as they can be easily mapped to integers. As an example, the ports in the TCP protocol are well represented using a range-based selector (e.g., 1024-65535). We include in the range-based selectors all the category of selectors that have been defined by Al-Shaer et al. as prefix match [Alshaer]. These selectors allow the specification of ranges of values by means of simple regular expressions. The typical case is the IP address selector (e.g., 10.10.1.*). There is no need to distinguish between prefix match and range-based selectors as 10.10.1.* easily maps to [10.10.1.0, 10.10.1.255].

Another category of selector types includes the regex-based selectors, where the matching is performed by using regular

expressions. This selector type is frequent at the application layer, where data are often represented as strings of text. The regex-based selector type also includes as sub-case the string-based selectors, where matching is evaluated using string matching algorithms (SMA) [Cormen]. Indeed, for our purposes, string matching can be mapped to regular expressions, even if in practice SMA are much faster. For instance, Squid (<http://www.squid-cache.org/>), a popular Web caching proxy that offers various access control capabilities, allows the definition of conditions on URLs that can be evaluated with SMA (e.g., `dstdomain`) or regex matching (e.g., `dstdom_regex`).

Finally, we introduce the idea of custom check selectors. For instance the malware analysis looks for specific patterns and returns a Boolean value is an example of custom check selector, if the logic of checking is not seen (nor really interesting) from the outside. In order to be properly used by high-level policy based processed (like reasoning systems, refinement systems) these custom check selector need at least to be described as black-boxes, that is, the list of fields that they process (inputs) in order to return the Boolean verdict (output).

4. Capability Model

Our model of capabilities is based on actions and traffic classification features. Indeed, the need for enforcing one of the actions that a security control can apply to packets/flows is the main reason to use a security control. Moreover, security controls have classification features that permit the identification of the target packets/flows of the actions enforced, i.e., the selectors presented in Section 3.1. A security manager decides for a specific security control depending on the actions and classification features. If the security control can enforce the needed actions and has the classification features needed to identify the packets flows required by a policy, then the security control is capable of enforcing the policy. Our refinement model needs to know NSF's capabilities to perform its operations.

However, security controls may have specific characteristics that automatic processes or administrators need to know when they have to generate configurations, like the available resolution strategies and the possibility to set default actions. We have ignored, to simplify this presentation, options to generate configurations that may have better performance, like the use of chains or ad hoc structures [Taylor]. Adding support to these forms of optimization is certainly feasible with a limited effort but it was outside the scope of this paper, that is, to show that adding security awareness to NFV

management and orchestration features is possible. It is one of the task for future work.

Capabilities can be used for two purposes: describing generic security functions, and describing specific products. With the term generic security function (GNSF) we denote known classes of security functions. The idea is to have generic components whose behaviour is as well understood as for the network components (i.e., a switch is a switch and we know to use it even if it may have some vendor-specific functions). These generic functions can be substituted by any product that owns the required capability at instantiation time.

We have analysed several classes of NSFs to prove the validity of our approach. We found the common features and defined a set of generic NSFs, including packet filter, URL filter, HTTP filter, VPN gateway, anti-virus, anti-malware, content filter, monitoring, anonymity proxy that will be described in a data model TBD.

Moreover, we have also categorized common extensions of the generic NSFs, packet filters that may decide based on time information. Moreover, some other packet filters add stateful features at ISO/OSI layer 4.

The next section will introduce our algebra to compose capabilities, defined to associate NSFs to capabilities and to check whether a NSF has the capabilities needed to enforce policies.

4.1. Algebra of capabilities

Our capabilities are defined by a 4-tuple:

$$(Ac; Cc; RSc; Dc) \quad [\quad (AC; CC; RSC; DC) = K$$

where AC is the set of all the supported actions, CC is set of all the supported conditions types, RSC is the set of all the supported resolutions strategies, and Dc [DC={F} U A, where F indicates that the default action is supported and can be freely selected by the policy editor, and explicitly indicates the default action if this cannot be explicitly configured.

We defined a syntax to specify:

- o subsets of actions: $Ac = \{a:action1, a:action2, \dots\} \quad [\quad AC$
- o subset of conditions: $Cc = \{c:cond1, c:cond2, \dots\} \quad [\quad CC$
- o subset of resolution strategies $RSc = \{rs:res1, rs:res2, \dots\} \quad [\quad RSC$

Given $cap1=(Ac1,Cc1,RSc1,def1)$ and $cap2=(Ac2,Cc2,RSc2,def2)$, we define

- o capability addition: $cap1+cap2 = (Ac1 \cup Ac2, Cc1 \cup Cc2, RSc1, def1)$
- o capability subtraction: $cap_1-cap_2 = ({Ac1 \setminus Ac2, Cc1 \setminus Cc2, RSc1, def1})$

Note that addition and subtraction do not alter the resolution strategy and the default action method, as our main intent was to model addition of modules

As an example, a generic packet filter that supports the first matching rule resolution strategies, allows the explicit specification of default actions and also supports time-based conditions. The description of its capabilities is the following:

- o $Apf = \{a:Allow, a:Deny\}$
- o $Cpf = \{c:IPsrc, c:IPdst, c:Psrc, c:Pdst, c:protType\}$
- o $Ctime = \{c:timestart, c:days, c:datestart, c:datestop\}$
- o $cap_pf=(Apf; Cpf; \{FMR\}; F)$
- o $cap_pf+time=cap_pf + Ctime$

By abuse of notation, we wrote $cap_pf+time=cap_pf + Ctime$ to shorten the more correct expression $cap_pf+time=cap_pf + (;Ctime; ;)\$$.

5. References

5.1. Normative References

- [I-D.ietf-i2nsf-framework] elopez@fortinet.com, e., Lopez, D., Dunbar, L., Strassner, J., Zhuang, X., Parrott, J., Krishnan, R., and S. Durbha, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-02 (work in progress), July 2016.
- [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>>.

[RFC3198] Westerinen, A., Schnizlein, J., Strassner, J., Scherling, M., Quinn, B., Herzog, S., Huynh, A., Carlson, M., Perry, J., and S. Waldbusser, "Terminology for Policy-Based Management", RFC 3198, DOI 10.17487/RFC3198, November 2001, <<http://www.rfc-editor.org/info/rfc3198>>.

5.2. Informative References

- [Alshaer] Al Shaer, E. and H. Hamed, "Modeling and management of firewall policies", 2004.
- [Bas12] Basile, C., Cappadonia, A., and A. Liroy, "Network-Level Access Control Policy Analysis and Transformation", 2012.
- [Bas15] Basile, C. and A. Liroy, "Analysis of application-layer filtering policies with application to HTTP", 2015.
- [Cormen] Cormen, T., "Introduction to Algorithms", 2009.
- [Lunt] van Lunteren, J. and T. Engbersen, "Fast and scalable packet classification", 2003.
- [Taylor] Taylor, D. and J. Turner, "Scalable packet classification using distributed crossproducting of field labels", 2004.

Authors' Addresses

Cataldo Basile
Politecnico di Torino
Corso Duca degli Abruzzi, 34
Torino, 10129
Italy

Phone: +39 011 090 7173
Email: cataldo.basile@polito.it

Diego R. Lopez
Telefonica I+D
Zurbaran, 12
Madrid, 28010
Spain

Phone: +34 913 129 041
Email: diego.r.lopez@telefonica.com

I2NSF
Internet-Draft
Intended status: Standards Track
Expires: January 2, 2017

S. Hares
Huawei
R. Moskowitz
HTT Consulting
July 1, 2016

I2NSF Capability Yang Model
draft-hares-i2nsf-capability-yang-00.txt

Abstract

This document defines a yang model that enables a I2NSF controller to control various network security functions in Network security devices.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 2, 2017.

Copyright Notice

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

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

Table of Contents

1. Introduction 2

2. High-level Yang 3

 2.1. capability per NSF 3

 2.2. Network Security Control 4

 2.3. Security Content Capabilities 6

 2.4. Attack Mitigation Capabilities 8

 2.5. IT Resources linked to Capabilities 9

3. Use of filter-based RIBS 9

4. YANG Modules 10

5. IANA Considerations 22

6. Security Considerations 22

7. References 22

 7.1. Normative References 22

 7.2. Informative References 22

Authors' Addresses 24

1. Introduction

[I-D.ietf-i2nsf-problem-and-use-cases] proposes two different types of interfaces:

- o North-bound interface (NBI) provided by the network security functions (NSFs)
- o Interface between I2NSF user/client with network controller:

This document provides a yang models that define the capabilities for security devices that can be utilized by I2NSF NBI between the I2RS network controller and the NSF devices to express the NSF devices capabilities. It can also be used by the IN2SF user application (or I2NSF client) to network controller to provide a complete list of the I2NSF capabilities the Network controller can control.

This document defines a yang data models based on the [I-D.xia-i2nsf-capability-interface-im], and initial work done in [I-D.xia-i2nsf-service-interface-dm]. Terms used in document are defined in [I-D.ietf-i2nsf-terminology].

[I-D.xia-i2nsf-capability-interface-im] defines the following type of functionality in NSFs.

- o network security control
- o content security control, and
- o attack mitigation control

This document contains high-level yang for each type of control. The features in each section have been built up from the following sources:

open-source: firewalls, IDS, IPS. This includes ECA policy for
 basic-firewalls: in router, switches, firewalls,
 firewall products commercial level
 specialized devices IDS, IPS

2. High-level Yang

This section provides an overview of the high level yang.

2.1. capability per NSF

The high level yang capabilities per NSF device, controller, or application is the following:

```
ietf-i2nsf-capability
  +--rw nsf-capabilities
    +--rw capability* [name]
      +--rw nsf-name string
      +--rw cfg-net-secctl-capabilities
        | uses pkt-eca-policy:pkt-eca-policy-set
      +--rw cfg-net-sec-content-capabilities
        | uses i2nsf-content-caps
        | uses i2nsf-content-sec-actions
      +--rw cfg-attack-mitigate-capabilities*
        | uses i2nsf-mitigate-caps
      +--rw ITResource [ITresource-name]
        | uses cfg-ITResources
```

Each of these section mirror sections in: [I-D.xia-i2nsf-capability-interface-im]. The high level yang for `cfg-net-secctl-capabilities`, `cfg-net-sec-content-capabilities`, and `cfg-attack-mitigate-capabilities`. This draft is also utilizes the concepts originated in Basile, Liyo, Pitscheider, and Zhao[2015] concerning conflict resolution, use of external data, and ITResources. The authors are grateful to Cataldo for pointing out this excellent work.

2.2. Network Security Control

This section defines the network security control capabilities for each NSF entity (device, controller, APP). The portion of the top level model that this explains is the following:

```

    +--rw cfg-net-secctl-capabilities
    |   uses pkt-eca-policy:pkt-eca-policy-set
  
```

Note that yang simply uses the `ietf-pkt-eca-policy-cfg` from [I-D.ietf-i2rs-pkt-eca-data-model].

Network Security Control Filter rules

```

module ietf-pkt-eca-policy
  +--rw pkt-eca-policy-cfg
  |   +--rw pkt-eca-policy-set
  |   |   +--rw groups* [group-name]
  |   |   |   +--rw group-name string
  |   |   |   +--rw vrf-name string
  |   |   |   +--rw address-family
  |   |   |   +--rw group-rule-list* [rule-name]
  |   |   |   |   +--rw rule-name
  |   |   |   |   +--rw rule-order-id
  |   |   |   |   +--rw default-action-id integer
  |   |   |   |   +--rw default-resolution-strategy-id integer
  |   |   +--rw rules* [order-id rule-name]
  |   |   |   +--rw order-id
  |   |   |   +--rw rule-name
  |   |   |   +--rw cfg-rule-conditions [cfgr-cnd-id]
  |   |   |   |   +--rw cfgr-cnd-id integer
  |   |   |   |   +--rw eca-event-match
  |   |   |   |   |   +--rw time-event-match*
  |   |   |   |   |   |   ...
  |   |   |   |   |   +--rw user-event-match*
  |   |   |   |   |   |   ...
  |   |   |   |   +--rw eca-condition-match
  |   |   |   |   |   +--rw eca-pkt-matches*
  |   |   |   |   |   |   ... (L1-L4 matches)
  |   |   |   |   |   +--rw eca-user-matches*
  |   |   |   |   |   |   ... (user, schedule, region, target,
  |   |   |   |   |   |   state, direction)
  |   |   |   +--rw cfg-rule-actions [cfgr-action-id]
  |   |   |   |   +--rw cfgr-action-id
  |   |   |   |   +--rw eca-actions* [action-id]
  |   |   |   |   |   +--rw action-id uint32
  |   |   |   |   |   +--rw eca-ingress-act*
  |   |   |   |   |   |   ... (permit, deny, mirror)
  
```


| | (other details on external data)

2.3. Security Content Capabilities

This section expands the

```
+-rw cfg-net-sec-content-capabilities
   |  uses i2nsf-content-caps
   |  uses i2nsf-content-sec-actions
```

Content Security Control

```

+--rw cfg-netsec-content-caps*
|
|  +--rw cfg-groups* [group-name]
|  |
|  |  +--rw group-name string
|  |  +--rw group-rule-list* [rule-name]
|  |  |
|  |  |  +--rw rule-name string
|  |  |  +--rw rule-order-id integer
|  |  |  +--rw default-action-id integer
|  |  |  +--rw default-resolution-strategy-id integer|
|  +--rw cfg-netsec-content-rules* [rule-order-id rule-name]
|  |
|  |  +--rw cfg-netsec-content-rule
|  |  |
|  |  |  +--rw rule-order-id integer
|  |  |  +--rw rule-name string
|  |  |  +--rw cfg-filter-rules
|  |  |  |
|  |  |  |  +--rw cfg-anti-virus-rule
|  |  |  |  |
|  |  |  |  |  +--rw source string //std or vendor name
|  |  |  |  |  ... description
|  |  |  +--rw cfg-IPS-rule
|  |  |  |
|  |  |  |  +--rw source string //std or vendor name
|  |  |  |  |
|  |  |  |  |  ... description
|  |  |  +--rw cfg-IDS-rule
|  |  |  |
|  |  |  |  +--rw source string //std or vendor name
|  |  |  |  |
|  |  |  |  |  ... description
|  |  |  +--rw cfg-url-filter-rule
|  |  |  |
|  |  |  |  +--rw source string //std or vendor name
|  |  |  |  |
|  |  |  |  |  ... description
|  |  |  +--rw cfg-file-block-rule
|  |  |  |
|  |  |  |  +--rw source string //std or vendor name
|  |  |  |  |
|  |  |  |  |  ... description
|  |  |  +--rw cfg-data-filter-rule
|  |  |  |
|  |  |  |  +--rw source string //std or vendor name
|  |  |  |  |
|  |  |  |  |  ... description
|  |  |  +--rw cfg-APP-behave-rule
|  |  |  |
|  |  |  |  +--rw source string //std or vendor name
|  |  |  |  |
|  |  |  |  |  ... description
|  |  |  +--rw cfg-mail-filter-rule
|  |  |  |
|  |  |  |  +--rw source string //std or vendor name
|  |  |  |  |
|  |  |  |  |  ... description
|  |  |  +--rw cfg-pkt-capture-rule
|  |  |  |
|  |  |  |  +--rw source string //std or vendor name
|  |  |  |  |
|  |  |  |  |  ... description
|  |  |  +--rw cfg-file-isolate-rule
|  |  |  |
|  |  |  |  +--rw source string //std or vendor name
|  |  |  |  |
|  |  |  |  |  ... description
+--rw cfg-sec-content-actions
      (need input on the actions )

```

2.4. Attack Mitigation Capabilities

The high level yang below expands the following section of the top-level model:

```

+--rw cfg-attack-mitigate-capabilities
|   uses cfg-attack-mitigate-caps

```

Attack mitigation

```

+--rw cfg-attack-mitigate-caps
|   +--rw cfg-groups* [group-name]
|   |   +--rw group-name string
|   |   +--rw group-rule-list* [rule-name]
|   |   |   +--rw rule-name string
|   |   |   +--rw rule-order-id integer
|   |   |   +--rw default-action-id integer
|   |   |   +--rw default-resolution-strategy-id integer|
|   +--rw cfg-netsec-content-rules* [rule-order-id rule-name]
|   |   +--rw rule-order-id integer
|   |   +--rw rule-name string
|   |   +--rw cfg-sync-flood* [sync-flood-fcn]
|   |   |   +--rw udp-flood-fcn string //std or vendor name
|   |   |   +--rw udp-flood-supported boolean
|   |   +--rw cfg-udp-flood* [udp-flood-fcn]
|   |   |   +--rw udp-flood-fcn string //std or vendor name
|   |   |   +--rw udp-flood-fcn-supported boolean
|   |   +--rw cfg-icmp-flood* [icmp-flood-fcn]
|   |   |   +--rw icmp-flood-fcn string //std/vendor name
|   |   |   +--rw icmp-flood-supported boolean
|   |   +--rw cfg-ip-frag-flood* [ipfrag-flood-fcn]
|   |   |   +--rw ipfrag-flood-fcn string //std/vendor name
|   |   |   +--rw ipfrag-flood-fcn-supported boolean
|   |   +--rw cfg-http-flood* [http-flood-fcn]
|   |   |   +--rw http-flood-fcn string //std or vendor name
|   |   |   +--rw http-flood-fcn-supported boolean
|   |   +--rw cfg-dns-flood* [dns-flood-fcn]
|   |   |   +--rw dns-flood-fcn string //std or vendor name
|   |   |   +--rw dns-flood-fcn-supported boolean
|   |   +--rw cfg-dns-amplify* [dns-amp-fcn]
|   |   |   +--rw dns-amp-fcn string //std or vendor name
|   |   |   +--rw dns-amp-fcn-supported boolean
|   |   +--rw cfg-SSL-DDoS-rule
|   |   |   +--rw ssl-dos-fcn string //std or vendor name
|   |   |   +--rw ssl-ddos-fcn-support boolean
|   |   +--rw cfg-IP-Sweep* [ipsweep-fcn]
|   |   |   +--rw ipsweep-fcn string //std or vendor name

```

```

| | | | |--rw ipsweep-fcn-supported boolean
| | | | |--rw cfg-Port-scanning [port-scan-fcn]
| | | | |--rw port-scan-fcn string //std or vendor name
| | | | |--rw port-scan-fcn-supported boolean
| | | | |--rw cfg-ping-of-death* [pingd-function]
| | | | |--rw pingd-fcn string //std or vendor name
| | | | |--rw pingd-fcn-supported boolean
| | | | |--rw cfg-oversize-ICMP* [o-icmp-fcn]
| | | | |--rw o-icmp-fcn string //std or vendor name
| | | | |--rw o-icmp-fcn-supported boolean

```

2.5. IT Resources linked to Capabilities

This section provides a link between capabilities and IT resources. This section has a list of IT Resources by name. Additional input is needed.

```

|--rw cfg-ITResources
| |--ITResources* [ITresource-name]
| | |--rw ITresource-name string
| | ..

```

3. Use of filter-based RIBS

The packet-eca policy is kept for configuration, I2RS ephemeral state, and BGP stored policy state in filter-based RIBS. These RIBS have the high-level yang structures below and are described in [I-D.ietf-i2rs-fb-rib-data-model]. These filter-ribs may be leveraged in I2NSF storage devices for the policy storage.

```

+--rw fb-ribs
  +--rw fb-rib* [rib-name]
    |   +--rw rib-name string
    |   |   rw fb-type identityref /config, i2rs, bgp
    |   +--rw rib-afi rt:address-family
    |   +--rw fb-rib-intf* [name]
    |   |   +--rw name string
    |   |   +--rw intf if:interface
    |   +--rw default-ribs
    |   |   +--rw rt-rib string           // routing kernel rib
    |   |   +--rw config-rib string;      // static rt-rib
    |   |   +--rw i2rs-rib string;        // ephemeral rt-rib
    |   |   +--rw bgp-instance-name string // bgp instance
    |   |   +--rw bgp-rib string         // bgp rib
    |   +--rw fb-rib-refs
    |   |   +--rw fb-rib-update-ref uint32 //count of writes
    |   +--rw mounts-using*
    |   |   +--rw mount-name string      //
    |   +--use pkt-eca:pkt-eca-policy-set

```

4. YANG Modules

```

<CODE BEGINS> file "ietf-i2nsf-capability@2016-06-26.yang"
module ietf-i2nsf-capability {
  namespace "urn:ietf:params:xml:ns:yang:ietf-i2nsf-capability";
  // replace with iana namespace when assigned
  prefix "i2nsf-capability";
  import ietf-pkt-eca-policy {
    prefix pkt-eca-policy;
  }
  // meta

  organization "IETF I2NSF WG";

  contact
    "email: Susan Hares: shares@ndzh.com
     email: Robert Moskowitz rgm@htt-consult.com;
     email: Frank Xia
     email: Aldo Basile cataldo.basile@polito.it";

  description
    "This module describes a capability model
     for I2NSF devices .";

  revision "2016-06-26" {
    description "initial revision";
    reference "draft-hares-i2nsf-capability-dm-00.txt";
  }
}

```

```
    }
  grouping ITResources {
    list ITResource {
      key ITResource-id;
      leaf ITResource-id {
        type uint64;
        description "ID for ITResource";
      }
      leaf ITResource-name {
        type string;
        description "ITResource name.";
      }
      description "list of IT Resources.";
    }
    description "IT Resource grouping.";
  }

  grouping cfg-sec-content-caps {
    list cfg-fcn-groups { // functions in 2 lists:
      key "group-name"; // group and functions
      leaf group-name {
        type string;
        description " name of function
          group";
      }
      list group-fnc-list {
        key "fnc-name";
        leaf fcn-name {
          type string;
          description "security content
            function name";
        }
        leaf fcn-order-id {
          type uint64;
          description "function order
            in list of functions.";
        }
        leaf default-action-id {
          type uint64;
          description "default
            extended action id";
        }
        leaf default-cr-resolve-id {
          type uint32;
          description "default
            policy conflict resolution
          ";
        }
      }
    }
  }
}
```

```
        policy identifier.";
    }
    description "list of
functions per group.
e.g. group A has
5 functions.";
}

description "list of
groups with associated
security content functions.";
}

list cfg-sec-content-fcns {
    key "fcn-order-id function-name";
    leaf fcn-order-id {
        type uint64;
        description "order id for rule";
    }
    leaf function-name {
        type string;
        description "rule name";
    }
    list anti-virus {
        key "anti-virus-name";
        leaf anti-virus-name {
            type string;
            description "name of
anti-virtus functionality";
        }
        leaf anti-virus-supported {
            type boolean;
            description "anti-virus
feature supported";
        }
    }
    description "anti-virus functions";
}
list IPS {
    key "IPS-name";
    leaf IPS-name {
        type string;
        description "name of
anti-virtus functionality";
    }
    leaf IPS-supported {
        type boolean;
        description "IPS
capability";
    }
}
```

```
        supported";
    }
    description "IPS capability";
}

list IDS {
    key "IDS-name";
    leaf IDS-name {
        type string;
        description "name of IDS";
    }
    leaf IDS-supported {
        type boolean;
        description "anti-virus
        feature supported";
    }
    description "IDS
    capabilities";
}

list url-filter {
    key "url-filter-name";
    leaf url-filter-name {
        type string;
        description "name of IDS";
    }
    leaf url-filter-supported {
        type boolean;
        description "url filter
        feature supported";
    }
    description "URL filter
    capabilities";
}

list file-block {
    key "fblock-name";
    leaf fblock-name {
        type string;
        description "name of
        file block function";
    }
    leaf fblock-supported {
        type boolean;
        description "anti-virus
        feature supported";
    }
}
```

```
    description "file block
capabilities";
}

list data-filter {
  key "dfilter-name";
  leaf dfilter-name {
    type string;
    description "name of
data filer";
  }
  leaf dfilter-supported {
    type boolean;
    description "anti-virus
feature supported";
  }
  description "data filter
capabilities";
}

list app-behave {
  key "app-behave-name";
  leaf app-behave-name {
    type string;
    description "name of
application behavior
control function.";
  }
  leaf app-behave-supported {
    type boolean;
    description "application
behavior control
security capability
supported.";
  }
  description "Application
behavior control security
capabilities";
}

list mail-filter {
  key "mfilter-name";
  leaf mfilter-name {
    type string;
    description "name of
data filer";
  }
  leaf mfilter-supported {
```

```
        type boolean;
        description "mail filter
supported";
    }
    description "mail filter";
}

list pkt-capture {
    key "pkt-capture-name";
    leaf pkt-capture-name {
        type string;
        description "name of
data filer";
    }
    leaf pkt-capture-supported {
        type boolean;
        description "pkt capture
facility supported";
    }
    description "packet capture
facility supported ";
}

list file-isolate {
    key "f-isolate-name";
    leaf f-isolate-name {
        type string;
        description "name of
file isolate capability";
    }
    leaf f-isolate-supported {
        type boolean;
        description "file isolate
capability supported ";
    }
    description "file isolate
capability ";
}
description "list of
security content capabilities.";
}
description "configured
security content capabilities";
}

grouping cfg-content-sec-actions {
```

```
list content-sec-actions {
    key "action-name";
    leaf action-name {
        type string;
        description "name of extra
                    content security action
                    beyond function policy";
    }
    description "list
                of content security actions";
}

description "configure
            content security actions
            configured beyond capability
            function existence";
}

grouping cfg-attack-mitigate-caps {
    // group and then rules
    list cfg-mitigate-fncs-groups {
        key "group-name";
        leaf group-name {
            type string;
            description " name of function
                        group";
        }
    }
    list group-mitigate-fncs-list {
        key "fcn-name";
        leaf fcn-name {
            type string;
            description "security content
                        function name";
        }
        leaf fcn-order-id {
            type uint64;
            description "function order
                        in list of functions.";
        }
        leaf default-action-id {
            type uint64;
            description "default
                        extended action id";
        }
        leaf default-cr-resolve-id {
            type uint32;
            description "default
                        policy conflict resolution
                        policy identifier.";
        }
    }
}
```

```
    }
    description "list of
functions per group.
e.g. group A has
5 functions.";
}

description "list of
groups with associated
attack mitigate functions.";
}

list cfg-attack-mitigate-rule {
  key "rule-order-id rule-name";
  leaf rule-order-id {
    type uint64;
    description "order id for
configured mitigate
function";
  }
  leaf rule-name {
    type string;
    description "mitigate
rule name";
  }
  list cfg-sync-flood {
    key sync-flood-fcn;
    leaf sync-flood-fcn {
      type string;
      description "name of
sync flood functionality";
    }
    leaf sync-flood-fcn-supported {
      type boolean;
      description "sync-flood
mitigation fcn supported";
    }
    description "list of
sync flood mitigation
functions ";
  }
  list cfg-udp-flood {
    key "udp-flood-fcn";
    leaf udp-flood-fcn {
      type string;
      description "name of
udp flood mitigation function ";
    }
  }
}
```

```
    }
    leaf udp-flood-fcn-supported {
      type boolean;
      description "udp flood
        prevent function
        capability supported";
    }
    description "list of
      udp-flood mitigation
      functions node
      (configured capability).";
  }

  list cfg-icmp-flood {
    key "icmp-flood-fcn";
    leaf icmp-flood-fcn {
      type string;
      description "name of
        icmp flood prevention
        function";
    }
    leaf icmp-flood-fcn-supported {
      type boolean;
      description "icmp
        flood mitigation
        feature supported";
    }
    description "list for
      icmp flood prevention
      functions part of
      attack mitigation
      capabilities.";
  }

  list cfg-http-flood {
    key "http-flood-fcn";
    leaf http-flood-fcn {
      type string;
      description "name of
        http flood
        mitigation function";
    }
    leaf http-flood-fcn-supported {
      type boolean;
      description "support
        for http flood function
        capability is active.";
    }
  }
}
```

```
    }
    description "list of
http flood
mitigation functions
configured ";
}

list cfg-dns-flood {
  key "dns-flood-fcn";
  leaf dns-flood-fcn {
    type string;
    description "name of
dns flood mitigation
function";
  }
  leaf dns-flood-fcn-supported {
    type boolean;
    description "dns flood
mitigation support is
active.";
  }
  description "list of
dns flood
mitigation functions
configured.";
}

list cfg-dns-amplify {
  key "dns-amplify-fcn";
  leaf dns-amplify-fcn {
    type string;
    description "name of
dns amplify mitigation
function.";
  }
  leaf dfilter-supported {
    type boolean;
    description "dns
amplification mitigation
function is active.";
  }
  description "list of
dns amplification
mitigation functions
configured.";
}

list SSL-DoS {
```

```
key "ssl-dos-fcn";
leaf ssl-dos-fcn {
  type string;
  description "name of
SSL DoS mitigation
function";
}
leaf ssl-dos-supported {
  type boolean;
  description "SSL DoS
mitigation function is
active.";
}
description "List of
SSL DoS functions configured.";
}

list cfg-IP-Sweep {
  key "ipsweep-fcn";
  leaf ipsweep-fcn {
    type string;
    description "name of
ip sweep mitigation
function.";
  }
  leaf ipsweep-fcn-supported {
    type boolean;
    description "IP Sweep
mitigation function
active.";
  }
  description "list of
IP Sweep mitigation
functions in NSF device.";
}

list cfg-Port-scanning {
  key "port-scan-fcn";
  leaf port-scan-fcn {
    type string;
    description "name of
port-scan mitigation
function.";
  }
  leaf port-scan-fcn-supported {
    type boolean;
    description "port scanning
mitigation fcn supported.";
  }
}
```

```
    }
    description "List of
port scanning mitigation
functions. ";
}

list cfg-ping-of-death {
  key "pingd-fcn";
  leaf pingd-fcn {
    type string;
    description "name of
ping of death
mitigation function";
  }
  leaf pingd-fcn-supported{
    type boolean;
    description "active support
for this ping of death
mitigation function";
  }
  description "List of ping of
death mitigation
functions.";
}
description "attack
mitigation rule .";
} // rules
description "configured
attack mitigation functions.";

} // cfg-attack-mitigate-policy-set

container i2nsf-capabilities {
  list capabilty {
    key "nsf-name";
    leaf nsf-name {
      type string;
      description "name of
nsf or nsf group
capabilities drawn from.";
    }
  }
  container cfg-net-secctl-capabilities {
    uses pkt-eca-policy:pkt-eca-policy-set;
    description "network security
control capabilities configured.";
  }
  container cfg-sec-content-capabilities {
    uses cfg-sec-content-caps;
  }
}
```

```
        uses cfg-content-sec-actions;
        description "security content
        capabilities configured.";
    }
    container cfg-attack-mitigate-capabilites {
        uses cfg-attack-mitigate-caps;
        description "attack mitigation capabilities";
    }
    container cfg-ITResources {
        uses ITResources;
        description "IT Resources
        associated with NSF.";
    }
    description "List of NSF
    capabilities per nsf, nsf group
    or nsf application.";
} //end of list

description "I2NSF capabilities";
} // end of container
}
<CODE ENDS>
```

5. IANA Considerations

No IANA considerations exist for this document at this time. URL will be added.

6. Security Considerations

Security of I2NSF is defined in (need reference here).

7. References

7.1. 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>>.

7.2. Informative References

[I-D.ietf-i2nsf-gap-analysis]
Hares, S., Moskowitz, R., and D. Zhang, "Analysis of Existing work for I2NSF", draft-ietf-i2nsf-gap-analysis-00 (work in progress), February 2016.

[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-00 (work in progress), February 2016.

[I-D.ietf-i2nsf-terminology]

Hares, S., Strassner, J., Lopez, D., and L. Xia, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-00 (work in progress), May 2016.

[I-D.ietf-i2rs-fb-rib-data-model]

Hares, S., Kini, S., Dunbar, L., Krishnan, R., Bogdanovic, D., and R. White, "Filter-Based RIB Data Model", draft-ietf-i2rs-fb-rib-data-model-00 (work in progress), June 2016.

[I-D.ietf-i2rs-pkt-eca-data-model]

Hares, S., Wu, Q., and R. White, "Filter-Based Packet Forwarding ECA Policy", draft-ietf-i2rs-pkt-eca-data-model-00 (work in progress), June 2016.

[I-D.ietf-netmod-acl-model]

Bogdanovic, D., Koushik, K., Huang, L., and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-06 (work in progress), December 2015.

[I-D.ietf-opsawg-firewalls]

Baker, F. and P. Hoffman, "On Firewalls in Internet Security", draft-ietf-opsawg-firewalls-01 (work in progress), October 2012.

[I-D.xia-i2nsf-capability-interface-im]

Xia, L., Zhang, D., elopez@fortinet.com, e., Bouthors, N., and L. Fang, "Information Model of Interface to Network Security Functions Capability Interface", draft-xia-i2nsf-capability-interface-im-05 (work in progress), March 2016.

[I-D.xia-i2nsf-service-interface-dm]

Xia, L., Strassner, J., and D. Bogdanovic, "Data Model of Interface to Network Security Functions Service Interface", draft-xia-i2nsf-service-interface-dm-00 (work in progress), February 2015.

- [RFC2975] Aboba, B., Arkko, J., and D. Harrington, "Introduction to Accounting Management", RFC 2975, DOI 10.17487/RFC2975, October 2000, <<http://www.rfc-editor.org/info/rfc2975>>.
- [RFC3198] Westerinen, A., Schnizlein, J., Strassner, J., Scherling, M., Quinn, B., Herzog, S., Huynh, A., Carlson, M., Perry, J., and S. Waldbusser, "Terminology for Policy-Based Management", RFC 3198, DOI 10.17487/RFC3198, November 2001, <<http://www.rfc-editor.org/info/rfc3198>>.
- [RFC3234] Carpenter, B. and S. Brim, "Middleboxes: Taxonomy and Issues", RFC 3234, DOI 10.17487/RFC3234, February 2002, <<http://www.rfc-editor.org/info/rfc3234>>.
- [RFC3539] Aboba, B. and J. Wood, "Authentication, Authorization and Accounting (AAA) Transport Profile", RFC 3539, DOI 10.17487/RFC3539, June 2003, <<http://www.rfc-editor.org/info/rfc3539>>.
- [RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <<http://www.rfc-editor.org/info/rfc4949>>.
- [RFC7277] Bjorklund, M., "A YANG Data Model for IP Management", RFC 7277, DOI 10.17487/RFC7277, June 2014, <<http://www.rfc-editor.org/info/rfc7277>>.

Authors' Addresses

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

Phone: +1-734-604-0332
Email: shares@ndzh.com

Robert Moskowitz
HTT Consulting
Oak Park, MI
USA

Phone: +1-248-968-9809
Email: rgm@htt-consult.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 5, 2017

S. Hyun
S. Woo
Y. Yeo
J. Jeong
Sungkyunkwan University
J. Park
ETRI
July 4, 2016

Service Function Chaining-Enabled I2NSF Architecture
draft-hyun-i2nsf-sfc-enabled-i2nsf-00

Abstract

This document describes an architecture of the I2NSF framework using security function chaining for security policy enforcement. Security function chaining enables composite inspection of network traffic by steering the traffic through multiple types of security functions according to the information model for the capability layer interface in the I2NSF framework. This document explains the additional components integrated into the I2NSF framework and their functionalities to achieve security function chaining. It also describes representative use cases to address major benefits from the proposed architecture.

Status of This Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-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 January 5, 2017.

Copyright Notice

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

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

Table of Contents

1. Introduction	3
2. Objective	3
3. Terminology	4
4. Architecture	6
4.1. SFC Policy Manager	8
4.2. SFC Catalog Manager	8
4.3. Developer's Management System	9
4.4. Classifier	9
4.5. Service Function Forwarder (SFF)	9
5. Use Cases	10
5.1. Dynamic Path Alternation	10
5.2. Enforcing Different SFPs Depending on Trust Levels	11
5.3. Effective Load Balancing with Dynamic SF Instantiation	12
6. Security Considerations	13
7. Acknowledgements	13
8. References	13
8.1. Normative References	13
8.2. Informative References	14

1. Introduction

To effectively cope with emerging sophisticated network attacks, it is necessary that various security functions cooperatively analyze network traffic [sfc-ns-use-cases] [RFC7498] [i2nsf-problem-and-use-cases] [i2nsf-capability-interface-im]. In addition, depending on the characteristics of network traffic and their suspiciousness level, the different types of network traffic need to be analyzed through the different sets of security functions. In order to meet such requirements, besides security policy rules for individual security functions, we need an additional policy about service function chaining (SFC) for network security [sfc-ns-use-cases] which determines a set of security functions through which network traffic packets should pass for inspection. In addition, [i2nsf-capability-interface-im] proposes an information model for capability layer interface of the I2NSF framework that enables a security function to trigger further inspection by executing additional security functions based on its own analysis results [i2nsf-framework]. However, the current design of the I2NSF framework does not consider network traffic steering fully in order to enable such chaining between security functions.

In this document, we propose an architecture that integrates additional components for security function chaining into the I2NSF framework. We extend the security controller's functionalities such that it can interpret a high-level policy of security function chaining into a low-level policy and manage them. It also keeps the track of the available service function (SF) instances for security functions and their information (e.g., network information and workload), and makes a decision on which SF instances to use for a given security function chain/path. Based on the forwarding information provided by the security controller, the service function forwarder (SFF) performs network traffic steering through various required security functions. A classifier is deployed for the enforcement of SFC policies given by the security controller. It performs traffic classification based on the policies so that the traffic passes through the required security function chain/path by the SFF.

2. Objective

- o Policy configuration for security function chaining: SFC-enabled I2NSF architecture allows policy configuration and management of security function chaining. Based on the chaining policy, relevant network traffic can be analyzed through various security functions in a composite, cooperative manner.

- o Network traffic steering for security function chaining: SFC-enabled I2NSF architecture allows network traffic to be steered through multiple required security functions based on the SFC policy. Moreover, the I2NSF information model for capability layer interface [i2nsf-capability-interface-im] requires a security function to call another security function for further inspection based on its own inspection result. To meet this requirement, SFC-enabled I2NSF architecture also enables traffic forwarding from one security function to another security function.
- o Load balancing over security function instances: SFC-enabled I2NSF architecture provides load balancing of incoming traffic over available security function instances by leveraging the flexible traffic steering mechanism. For this objective, it also performs dynamic instantiation of a security function when there are an excessive amount of requests for that security function.

3. Terminology

This document uses the terminology described in [RFC7665], [RFC7665] [sfc-ns-use-cases] [i2nsf-terminology][ONF-SFC-Architecture].

- o Service Function/Security Function (SF): A function that is responsible for specific treatment of received packets. A Service Function can act at various layers of a protocol stack (e.g., at the network layer or other OSI layers) [RFC7665]. In this document, SF is used to represent both Service Function and Security Function. Sample Security Service Functions are as follows: Firewall, Intrusion Prevention/Detection System (IPS/IDS), Deep Packet Inspection (DPI), Application Visibility and Control (AVC), network virus and malware scanning, sandbox, Data Loss Prevention (DLP), Distributed Denial of Service (DDoS) mitigation and TLS proxy.
- o Classifier: An element that performs Classification. It uses a given policy from SFC Policy Manager.
- o Service Function Chain (SFC): A service function chain defines an ordered set of abstract service functions and ordering constraints that must be applied to packets and/or frames and/or flows selected as a result of classification [RFC7665].
- o Service Function Forwarder (SFF): A service function forwarder is responsible for forwarding traffic to one or more connected service functions according to information carried in the SFC encapsulation, as well as handling traffic coming back from the SF. Additionally, an SFF is responsible for delivering traffic to

a classifier when needed and supported, transporting traffic to another SFF (in the same or the different type of overlay), and terminating the Service Function Path (SFP) [RFC7665].

- o Service Function Path (SFP): The service function path is a constrained specification of where packets assigned to a certain service function path must be forwarded. While it may be so constrained as to identify the exact locations for packet processing, it can also be less specific for such locations [RFC7665].
- o SFC Policy Manager: It is responsible for translating a high-level policy into a low-level policy, and performing the configuration for SFC-aware nodes, passing the translated policy and configuration to SFC-aware nodes, and maintaining a stabilized network.
- o SFC Catalog Manager: It is responsible for keeping the track of the information of available SF instances. For example, the information includes the supported transport protocols, IP addresses, and locations for the SF instances.
- o Control Nodes: It collectively refer to SFC Policy Manager, SFC Catalog Manager, SFF, and Classifier.
- o Service Path Identifier (SPI): It identifies a service path. The classifier MUST use this identifier for path selection and the Control Nodes MUST use this identifier to find the next hop [sfc-nsh].
- o Service Index (SI): It provides a location within the service path. SI MUST be decremented by service functions or proxy nodes after performing the required services [sfc-nsh].
- o Network Service Header (NSH): The header is used to carry SFC related information. Basically, SPI and SI should be conveyed to the Control Nodes of SFC via this header.
- o SF Forwarding Table: SFC Policy Manager maintains this table. It contains all the forwarding information on SFC-enabled I2NSF architecture. Each entry includes SFF identifier, SPI, SI, and next hop information. For example, an entry ("SFF: 1", "SPI: 1", "SI: 1", "IP: 192.168.xx.xx") is interpreted as follows: "SFF 1" should forward the traffic containing "SPI 1" and "SI 1" to "IP=192.168.xx.xx".

4. Architecture

This section describes an SFC-enabled I2NSF architecture and the basic operations of service chaining. It also includes details about each component of the architecture.

Figure 1 describes the components of SFC-enabled I2NSF architecture. Our architecture is designed to support a composite inspection of traffic packets in transit. According to the inspection result of each SF, the traffic packets could be steered to another SF for further detailed analysis. It is also possible to reflect a high-level SFC-related policy and a configuration from I2NSF Client on the components of the original I2NSF framework. Moreover, the proposed architecture provides load balancing, auto supplementary SF generation, and the elimination of unused SFs. In order to achieve these design purposes, we integrate several components to the original I2NSF framework. In the following sections, we explain the details of each component.

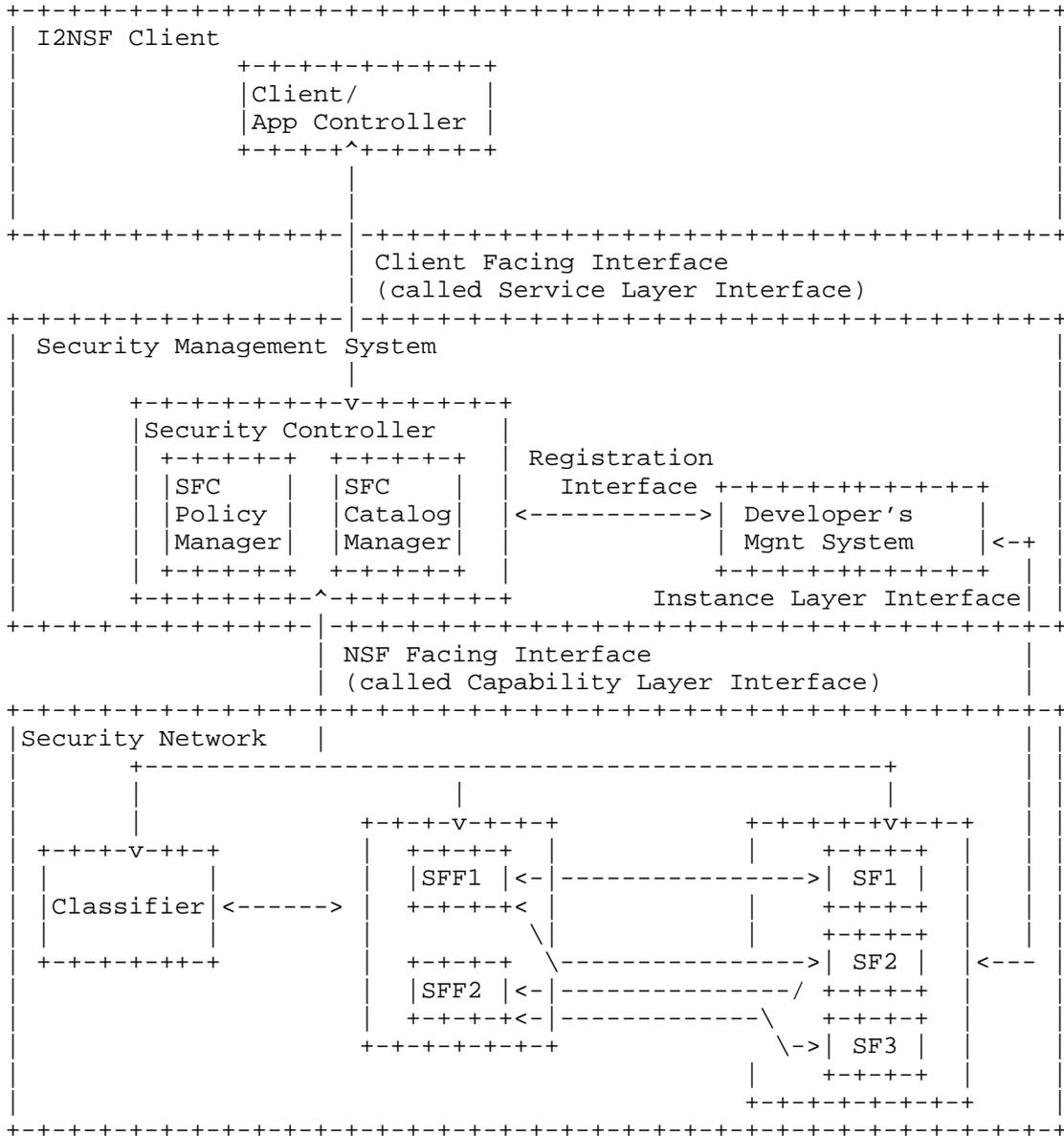


Figure 1: SFC-enabled I2NSF

4.1. SFC Policy Manager

SFC Policy Manager is a core component in our system. It is responsible for the following two things: (1) Interpreting a high-level SFC policy (or configuration) into a low-level SFC policy (or configuration), which is given by I2NSF Client, and delivering the interpreted policy to Classifiers for security function chaining. (2) Generating an SF forwarding table and distributing the forwarding information to SFF(s) by consulting with SFC Catalog Manager. As Figure 1 describes, SFC Policy Manager performs these additional functionalities through Service Layer Interface and Capability Layer Interface.

Given a high-level SFC policy/configuration from I2NSF Client via Service Layer Interface, SFC Policy Manager interprets it into a low-level policy/configuration comprehensible to Classifier(s), and then delivers the resulting low-level policy to them. Moreover, SFC Policy Manager possibly generates new policies for the flexible change of traffic steering to rapidly react to the current status of SFs. For instance, it could generate new rules to forward all subsequent packets to "Firewall Instance 2" instead of "Firewall Instance 1" in the case where "Firewall Instance 1" is under congestion.

SFC Policy Manager gets information about SFs from SFC Catalog Manager to generate SF forwarding table. In the table generation process, SFC Policy Manager considers various criteria such as SFC policies, SF load status, SF physical location, and supported transport protocols. An entry of the SF forwarding table consists of SFF Identifier, SFP, SI, and next hop information. The examples of next hop information includes the IP address and supported transport protocols (e.g., VxLAN and GRE). These forwarding table updates are distributed to SFFs with either push or pull methods.

4.2. SFC Catalog Manager

In Figure 1, SFC Catalog Manager is a component integrated into Security Controller. It is responsible for the following three things: (1) Maintaining the information of every available SF instance such as IP address, supported transport protocol, service name, and load status. (2) Responding the queries of available SF instances from SFC Policy Manager so as to help to generate a forwarding table entry relevant to a given SFP. (3) Requesting Developer's Management System for the dynamic instantiation of supplementary SF instances to avoid service congestion or the elimination of an existing SF instance to avoid resource waste.

Whenever a new SF instance is registered, Developer's Management

System passes the information of the registered SF instance to SFC Catalog Manager, so Catalog Manager maintains a list of the information of every available SF instance. Once receiving a query of a certain SFP from SFC Policy Manager, SFC Catalog Manager searches for all the available SF instances applicable for that SFP and then returns the search result to SFC Policy Manager.

In our system, each SF instance periodically reports its load status to SFC Catalog Manager. Based on such reports, SFC Catalog Manager updates the information of the SF instances and manages the pool of SF instances by requesting Developer's Management System for the additional instantiation or elimination of the SF instances. Consequently, SFC Catalog Manager enables efficient resource utilization by avoiding congestion and resource waste.

4.3. Developer's Management System

We extend Developer's Management System for additional functionalities as follows. As mentioned above, SFC Catalog Manager requests Developer's Management System to create additional SF instances when the existing instances of that service function are congested. On the other hand, when there are an excessive number of instances for a certain service function, SFC Policy Manager requests Developer's Management System to eliminate some of the SF instances. As a response to such requests, Developer's Management System creates and/or removes SF instances. Once it creates a new SF instance or removes an existing SF instance, the changes must be notified to SFC Catalog Manager.

4.4. Classifier

Classifier is a logical component that may exist as a standalone component or a submodule of another component. In our system, the initial classifier is typically located at an entry point like a border router of the network domain, and performs the initial classification of all incoming packets according to the SFC policies, which are given by SFC policy manager. The classification means determining the SFP through which a given packet should pass. Once the SFP is decided, the classifier constructs an NSH that specifies the corresponding SPI and SI, and attaches it to the packet. The packet will then be forwarded through the determined SFP on the basis of the NSH information.

4.5. Service Function Forwarder (SFF)

It is responsible for the following two functionalities: (1) Forwarding the packets to the next SFF/SF. (2) Handling re-classification request from SF.

An SFF basically takes forwarding functionality, so it needs to find the next SF/SFF for the incoming traffic. It will search its forwarding table to find the next hop information that corresponds to the given traffic. In the case where the SFF finds a target entry on its forwarding table, it just forwards the traffic to the next SF/SFF specified in the next hop information. If an SFF does not have an entry for a given packet, it will request the next hop information to SFC Policy Manager with SFF identifier, SPI, and SI information. The SFC Policy Manager will respond to the SFF with next hop information, and then the SFF updates its forwarding table with the response, forwarding the traffic to the next hop.

Sometimes an SF may want to forward the traffic, which is highly suspicious, to another SF for further inspection. The SF then appends the inspection result to the MetaData field of the NSH and delivers it to the source SFF. The attached MetaData includes a re-classification request to change the SFP of the traffic to another SFP for stronger inspection. When the SFF receives the traffic requiring re-classification, it forwards the traffic to the Classifier where re-classification will be eventually performed.

5. Use Cases

This section introduces three use cases for the SFC-enabled I2NSF architecture : (1) Dynamic Path Alternation, (2) Enforcing Different SFPs Depending on Trust Levels, and (3) Effective Load Balancing with Dynamic SF Instantiation.

5.1. Dynamic Path Alternation

In SFC-enabled I2NSF architecture, a Classifier determines the initial SFP of incoming traffic according to the SFC policies. The classifier then attaches an NSH specifying the determined SFP of the packets, and they are analyzed through the SFs of the initial SFP. However, SFP is not a static property, so it could be changed dynamically through re-classification. A typical example is for a certain SF in the initial SFP to detect that the traffic is highly suspicious (likely to be malicious). In this case, the traffic needs to take stronger inspection through a different SFP which consists of more sophisticated SFs.

Figure 2 illustrates an example of such dynamic SFP alternation in a DDoS attack scenario. SFP-1 represents the default Service Function Path that the traffic initially follows, and SFP-1 consists of AVC, Firewall, and IDS/IPS. If the IDS/IPS suspects that the traffic is attempting DDoS attacks, it will change the SFP of the traffic from the default to SFP-2 so that the DDoS attack mitigator can execute a proper countermeasure against the attack.

Such SFP alternation is possible in the proposed architecture with re-classification. In Figure 1, to initiate re-classification, the IDS/IPS appends its own inspection result to the MetaData field of NSH and deliver it to the SFF from which it has originally received the traffic. The SFF then forwards the received traffic including the inspection result from the IDS/IPS to Classifier for re-classification. Classifier checks the inspection result and determines the new SFP (SFP-2) associated with the inspection result in the SFC policy, and updates the NSH with the SPI of SFP-2. The traffic is forwarded to the DDoS attack mitigator.

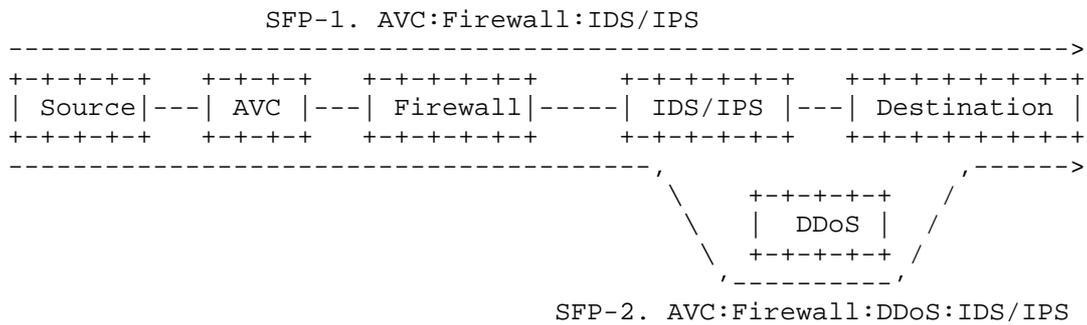


Figure 2: Dynamic SFP Alternation Example

5.2. Enforcing Different SFPs Depending on Trust Levels

Because the traffic coming from a trusted source is highly likely to be harmless, it does not need to be inspected excessively. On the other hand, the traffic coming from an untrusted source requires an in-depth inspection. By applying minimum required security functions to the traffic from a trusted source, it is possible to prevent the unnecessary waste of resources. In addition, we can concentrate more resources on potential malicious traffic. In the SFC-enabled I2NSF architecture, by configuring an SFC Policy to take into account the levels of trust of traffic sources, we can apply different SFPs to the traffic coming from different sources.

Figure 3(a) and Figure 3(b) represent SFPs applicable to traffic from trusted and untrusted sources, respectively. In Figure 3(a), we assume a lightweight IDS/IPS which is configured to perform packet header inspection only. In this scenario, when receiving the traffic from a trusted source, the classifier determines the SFP in Figure 3(a) such that the traffic passes through just a simple analysis by the lightweight IDS/IPS. On the other hand, traffic from an untrusted source passes more thorough examination through the SFP

1. SFC Catalog Manager detects that the firewall instance is receiving too much requests. Currently, there are no additional firewall instances available.
2. SFC Catalog Manager requests Developer's Management System to create a new firewall instance.
3. Developer's Management System creates a new firewall instance and then registers the information of the new firewall instance to SFC Catalog Manager.
4. SFC Catalog Manager updates the SFC Information Table to reflect the new firewall instance, and notifies SFC Policy Manager of this update.
5. Based on the received information, SFC Policy Manager updates the forwarding information for traffic steering and sends the new forwarding information to the SFF.
6. According to the new forwarding information, the SFF forwards the subsequent traffic to the new firewall instance. As a result, we can effectively alleviate the burden of the existing firewall instance.

6. Security Considerations

To enable security function chaining in the I2NSF framework, we adopt the additional components in the SFC architecture. Thus, this document shares the security considerations of the SFC architecture that are specified in [RFC7665] for the purpose of achieving secure communication among components in the proposed architecture.

7. Acknowledgements

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No.R-20160222-002755, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning).

8. References

8.1. Normative References

- | | |
|-----------|--|
| [RFC7665] | Boucadair, M. and C. Jacquenet,
"Software-Defined Networking: A
Perspective from within a Service
Provider Environment", RFC 7665,
March 2014. |
|-----------|--|

[sfc-nsh] Lopez, E., Lopez, D., Dunbar, L., Zhuang, X., Parrott, J., Krishnan, R., and S. Durbha, "Framework for Interface to Network Security Functions", draft-ietf-sfc-nsh-05 , June 2015.

[sfc-ns-use-cases] Wang, E., Leung, K., Felix, J., and J. Iyer, "Service Function Chaining Use Cases for Network Security", draft-wang-sfc-ns-use-cases-01 , March 2016.

8.2. Informative References

[RFC7498] Quinn, P. and T. Nadeau, "Problem Statement for Service Function Chaining", RFC 7498, April 2015.

[i2nsf-capability-interface-im] Xia, L., Zhang, D., Lopez, E., Bouthors, N., and L. Fang, "Information Model of Interface to Network Security Functions Capability Interface", draft-xia-i2nsf-capability-interface-im-05 , March 2016.

[i2nsf-framework] Lopez, E., Lopez, D., Dunbar, L., Strassner, J., Zhuang, X., Parrott, J., Krishnan, R., and S. Durbha, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-00 , May 2016.

[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-00 , February 2016.

[i2nsf-terminology] Hares, S., Strassner, J., Lopez, D., and L. Xia, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-00 , April 2016.

[ONF-SFC-Architecture]

ONF, "L4-L7 Service Function
Chaining Solution Architecture",
June 2015.

Authors' Addresses

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

Phone: +82 31 290 7222
Fax: +82 31 299 6673
EMail: swhyun77@skku.edu
URI: <http://imtl.skku.ac.kr/>

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

Phone: +82 31 290 7222
Fax: +82 31 299 6673
EMail: suwoo@imtl.skku.ac.kr,
URI: http://imtl.skku.ac.kr/index.php?mid=member_student

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

Phone: +82 31 290 7222
Fax: +82 31 299 6673
EMail: yunsuk@imtl.skku.ac.kr,
URI: http://imtl.skku.ac.kr/index.php?mid=member_student

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

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

Jung-Soo Park
Electronics and Telecommunications Research Institute
218 Gajeong-Ro, Yuseong-Gu
Daejeon 305-700
Republic of Korea

Phone: +82 42 860 6514
EMail: pjs@etri.re.kr

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

July 5, 2016

Framework for Interface to Network Security Functions
draft-ietf-i2nsf-framework-02.txt

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, except to publish it as an RFC and to translate it into languages other than English.

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-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 January 5, 2009.

Copyright Notice

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

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

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.....3
- 2. Conventions used in this document.....4
- 3. I2NSF Reference Model.....4
 - 3.1. Client Facing Interface.....5
 - 3.2. NSFs Facing Interface.....6
 - 3.3. Registration Interface.....7
- 4. Threats Associated with Externally Provided NSFs.....8

5. Potential pitfalls to Avoid in Managing Flow-based NSFs.....	9
6. The Network Connecting I2NSF Components.....	10
6.1. Network connecting I2NSF Clients and I2NSF Controller....	10
6.2. Network Connecting the Security Controller and NSFs.....	10
6.3. Interface to vNSFs.....	11
7. I2NSF Flow Security Policy Structure.....	12
7.1. Client Facing Flow Security Policy structure.....	13
7.2. NSF Facing Flow Security Policy structure.....	14
7.3. Difference from ACL data model.....	15
8. Capability Negotiation.....	16
9. Registration consideration.....	17
9.1. Flow-based NSF Capability Characterization.....	17
9.2. Registration Categories.....	18
10. Manageability Considerations.....	20
11. Security Considerations.....	21
12. IANA Considerations.....	21
13. References.....	21
13.1. Normative References.....	21
13.2. Informative References.....	22
14. Acknowledgments.....	23

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

vNSF: refers to NSF being instantiated on Virtual Machines.

3. I2NSF Reference Model

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

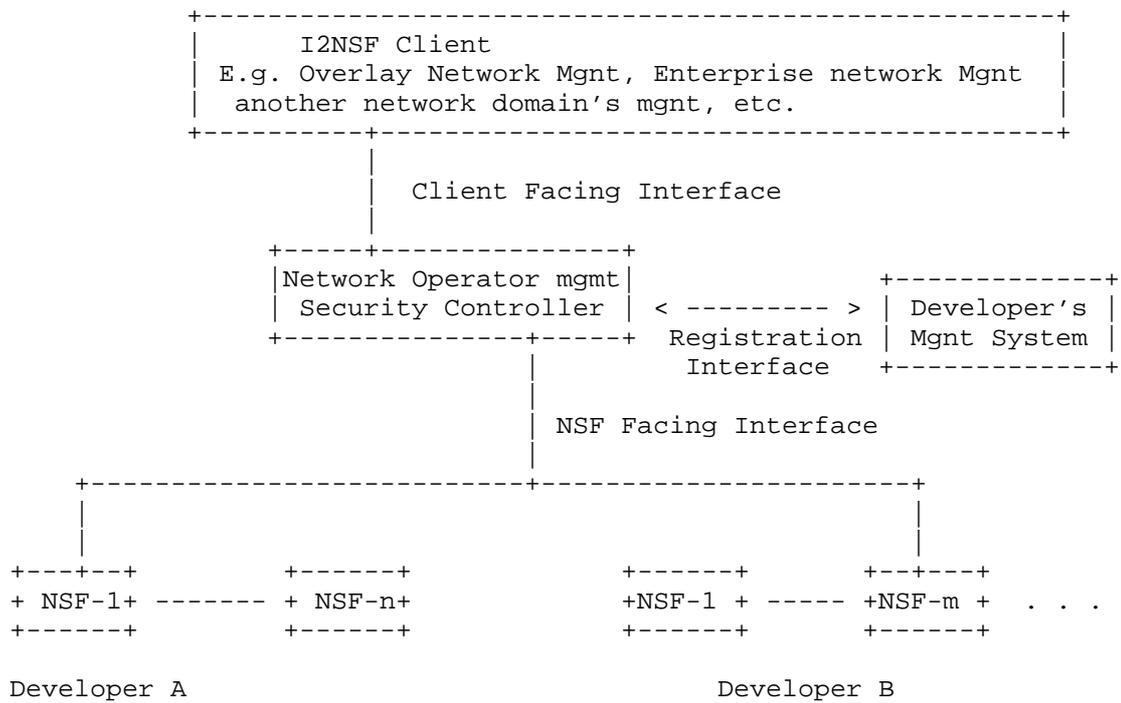


Figure 1: I2NSF Reference Model

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.

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

3.2. NSF's Facing Interface

The NSF's 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) Configuration - deals with the management and configuration of the NSF device itself, such as port address configurations. Configuration deals with attributes that are relatively static.
- 2) Signaling - which represents logging and query functions between the NSF and external systems. Signaling API functions may also be defined by other protocols, such as SYSLOG and DOTS.
- 3) Rule Provisioning - used to control the rules that govern how packets are treated by the NSFs. Due to the need of applications/controllers to dynamically control what traffic they need to receive, much of the I2NSF efforts towards interface development will be in this area.

This draft proposes that a rule provisioning interface to NSFs can be developed on a 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.

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

- o 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.
- o 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.

Mutual 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 MUST be performed, establishing the desired level of assurance. This level of assurance will determine how stringent are the requirements for authentication (in both directions), and how detailed any other attestation procedures (as described in [Remote-Attestation]) will be.

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

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's sub-controller, which in turn disseminates the policies 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.

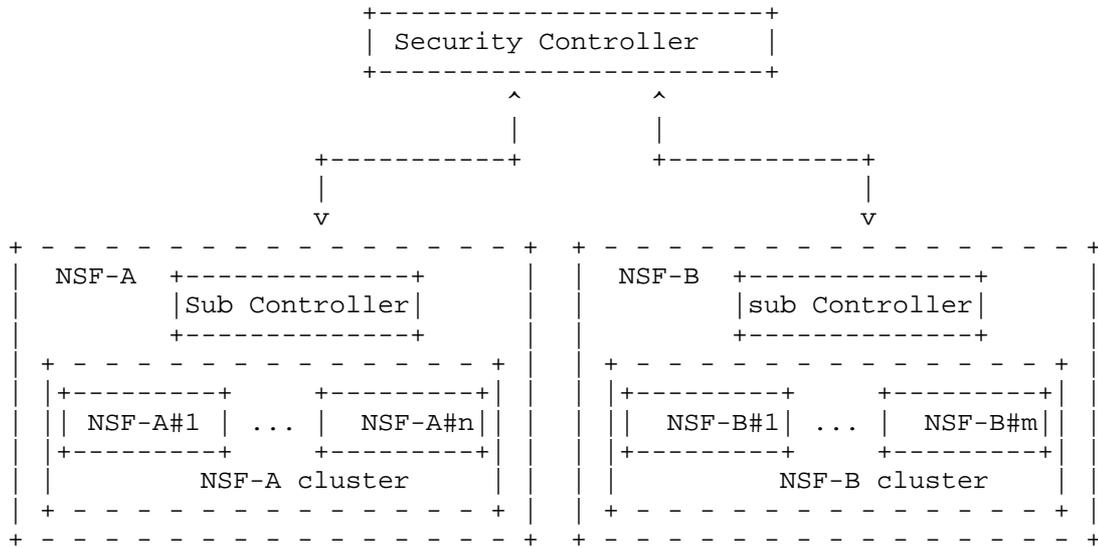


Figure 2: Cluster of NSF Instantiations Management

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 service layer security Policies:

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

One 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	Port
SCTP	syn
DCCP	ack
	fin

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

Table 1: Subject Capability Index

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

Connection 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 NSF's
- 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 NSF's 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 NSF's 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 Virtualization (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
Email: 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

I2NSF WG
Internet-Draft
Intended status: Informational
Expires: January 21, 2017

S. Hares
R. Moskowitz
Huawei
D. Zhang
July 20, 2016

Analysis of Existing work for I2NSF
draft-ietf-i2nsf-gap-analysis-02.txt

Abstract

This document analyzes the current state of the art for security management devices and security devices technologies in industries and the existing IETF work/protocols that are relevant to the Interface to Network Security Function (I2NSF). The I2NSF focus is to define data models and interfaces in order to control and monitor the physical and virtual aspects of network security functions.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 21, 2017.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	3
1.1.	What is I2NSF	3
1.2.	Structure of this Document	4
1.3.	Terms and Definitions	5
1.3.1.	Requirements Terminology	5
1.3.2.	Definitions	5
2.	IETF Gap analysis	6
2.1.	Traffic Filters	6
2.1.1.	Overview	6
2.1.2.	Middle-box Filters	9
2.1.3.	Security Work	10
3.	ETSI NFV	13
3.1.	ETSI Overview	13
3.2.	I2NSF Gap Analysis	15
4.	OPNFV	15
4.1.	OPNFV Moon Project	15
4.2.	Gap Analysis for OPNFV Moon Project	17
5.	OpenStack Security Firewall	17
5.1.	Overview of API for Security Group	18
5.2.	Overview of Firewall as a Service	18
5.3.	I2NSF Gap analysis	19
6.	CSA Secure Cloud	19
6.1.	CSA Overview	19
6.1.1.	CSA Security as a Service (SaaS)	20
6.1.2.	Identity Access Management (IAM)	21
6.1.3.	Data Loss Prevention (DLP)	22
6.1.4.	Web Security (Web)	23
6.1.5.	Email Security (email))	24
6.1.6.	Security Assessment	25
6.1.7.	Intrusion Detection	26
6.1.8.	Security Information and Event Management(SIEM)	27
6.1.9.	Encryption	28
6.1.10.	Business Continuity and Disaster Recovery (BC/DR)	29
6.1.11.	Network Security Devices	30
6.2.	I2NSF Gap Analysis	31
7.	IEEE security	31
7.1.	Port-based Network Access Control [802.1X]	31
7.2.	MAC security (802.1AE)	32
7.3.	Secure Device Identity [802.1AR]	33
8.	In-depth Review of IETF protocols	34
8.1.	NETCONF and RESTCONF	34
8.2.	I2RS Protocol	35
8.3.	NETMOD YANG modules	35

8.4. COPS 36

8.5. PCP 37

8.6. NSIS - Next Steps in Signaling 38

9. IANA Considerations 39

10. Security Considerations 39

11. Contributors 39

12. References 39

 12.1. Normative References 39

 12.2. Informative References 40

Authors' Addresses 48

1. Introduction

This documents provides a gap analysis for I2NSF.

1.1. What is I2NSF

A Network Security Function (NSF) ensures integrity, confidentiality and availability of network communications, detects unwanted activity, and/or blocks out or at least mitigates the effects of unwanted activity. NSFs are provided and consumed in increasingly diverse environments. For example, users of NSFs could consume network security services offered on multiple security products hosted one or more service provider, their own enterprises, or a combination of the two.

The lack of standard interfaces to control and monitor the behavior of NSFs makes it virtually impossible for security service providers to automate service offerings that utilize different security functions from multiple vendors.

The Interface to Network Service Functions (I2NSF) work proposes to standardize a set of software interfaces to control and monitor the physical and virtual NSFs. Since different security vendors support different features and functions, the I2NSF will focus on the flow-based NSFs that provide treatment to packets or flows such found in IPS/IDS devices, web filtering devices, flow filtering devices, deep packet inspection devices, pattern matching inspection devices, and re-mediation devices.

There are two layers of interfaces envisioned in the I2NSF approach:

- o The I2NSF Capability Layer specifies how to control and monitor NSFs at a functional implementation level. This is the focus for this phase of the I2NSF Work.

- o The I2NSF Service Layer defines how the security policies of clients may be expressed and monitored. The Service Layer is out of scope for this phase of I2NSF's work.

For the I2NSF Capability Layer, the I2NSF work proposes an interoperable protocol that passes NSF provisioning rules and orchestration information between the I2NSF client on a network manager and the I2NSF agent on an NSF. It is envisioned that clients of the I2NSF interfaces include management applications, service orchestration systems, network controllers, or user applications that may solicit network security resources.

The I2NSF work to define this protocol includes the following work:

- o defining an informational model that defines the concepts for standardizing the control and monitoring of NSFs,
- o defining a set of YANG data models from the information model that identifies the data that must be passed,
- o creating a capability registry (an IANA registry) that identifies the characteristics and behaviours of NSFs in vendor-neutral vocabulary without requiring the NSFs to be standardized.
- o examining existing secure communication mechanisms to identify the appropriate ones for carrying the data that provisions and monitors information between the NSFs and their management entity (or entities).

1.2. Structure of this Document

This document provides an analysis of the gaps in the state of art in the following industry forums:

IETF working groups (Section 2)

ETSI Network Functions Virtualization Industry Specification Group (ETSI NFV ISG), (Section 3)

OPNFV Open Source Group (Section 4)

Open Stack - Firewall as a service (OpenStack Firewall FaaS) (Section 5) (http://docs.openstack.org/admin-guide-cloud/content/install_neutron-fwaas-agent.html)

Cloud Security Alliance Security (CSA) as a Service (Section 6) (https://cloudsecurityalliance.org/research/secaas/#_overview)

In-Depth Review of Some IETF Protocols (Section 7)

1.3. Terms and Definitions

1.3.1. Requirements Terminology

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, BCP 14 [RFC2119] and indicate requirement levels for compliant CoAP.

1.3.2. Definitions

The following are a few definitions out of the terminology draft utilized in this draft. For additional definitions please see: [I-D.hares-i2nsf-terminology].

Network Security Function (NSF): is a function that is provided as a set of security-related service function. Typically, an NSF may be responsible for detecting unwanted activity and blocking/mitigating the effect of such unwanted activity in order to fulfil the service requirements. The NSF can help in supporting communication stream integrity and confidentiality.

Cloud Data Center (DC): A data center that may/may not be run on the premises of enterprises, but has compute/storage resources that can be requested or purchased by the enterprises. The enterprise is actually getting a virtual data center. The Cloud Security Alliance (CSA) (<http://cloudsecurityalliance.org>) focuses on adding security to this environment. A specific research topic is security as a service within the cloud data center.

Cloud-based security functions: Network Security Functions (NSFs) that may be hosted and managed by service providers or a different administrative entity.

Domain: The term Domain in this draft has the following different connotations in different scenarios:

- * Client--Provider relationship, i.e. client requesting some network security functions from its provider;
- * Domain A - Domain B relationship, i.e. one operator domain requesting some network security functions from another operator domain; or

- * Applications -- Network relationship, i.e. an application (e.g. cluster of servers) requesting some functions from network, etc.

The domain context is important because it indicates the interactions the security is focused on.

I2NSF server/agent: A software instance that implements a network security function that receives provisioning information and requests operational data (e.g. monitoring data) from an I2NSF client. It is also responsible for enforcing the policies that it receives from an I2NSF client.

I2NSF client: A security client software that utilizes the I2NSF protocol to read, write or change the provisioning network security device via software interface using the I2NSF protocol (denoted as I2RS Agent)

I2NSF Management System: I2NSF Client operates within an network management system which serves as a collections and distribution point for security provisioning and filter data.

2. IETF Gap analysis

The IETF gap analysis first examines the IETF mechanisms which have been developed to secure the IP traffic flows through a network. Traffic filters have been defined by IETF specifications at the access points, the middle-boxes, or the routing systems. Protocols have been defined to carry provisioning and filtering traffic between a management system and an IP system (router or host system). Current security work (SACM working group (WG), MILE WG, and DOTS WG) is providing correlation of events monitored with the policy set by filters. This section provides a review the filter work, protocols, and security correlation for monitors.

2.1. Traffic Filters

2.1.1. Overview

The earliest filters defined by IETF were access filters which controlled the acceptance of IP packet data flows. Additional policy filters were created as part of the following protocols:

- o COPS protocol [RFC2748] for controlling access to networks,
- o Next Steps in Signalling (NSIS) work (architecture: [RFC4080] protocol: [RFC5973]) - for supporting signaling about a data flow along its path, and

- o Port Control Protocol (PCP) - allows the IPv4/IPv6 host to control how IPv6/IPv4 packets are translated and forwarded by NATS and firewalls.

Today NETMOD and I2RS Working groups are specifying additional filters in YANG modules to be used as part of the NETCONF or I2RS enhancement of NETCONF/RESTCONF.

Route filtering is outside the scope of the flow filtering, but the flow filtering may be impacted by route filtering. An initial model for routing policy is in [I-D.ietf-rtgwg-policy-model]

This section provides an overview of the flow filtering as an introduction to the I2NSF Gap analysis. Additional detail on NETCONF, NETMOD, I2RS, PCP, and NSIS is available in Section 7.

2.1.1.1. Data Flow Filters in NETMOD and I2RS

The current work on expanding these filters is focused on combining a configuration and monitoring protocol with YANG data models. [I-D.ietf-netmod-acl-model] provides a set of access list filters which can permit or deny traffic flow based on headers at the MAC Layer, IP Layer, and Transport Layer. The configuration and monitoring protocols which can pass the filters are: NETCONF protocol [RFC6241], RESTCONF [I-D.ietf-netconf-restconf], and the I2RS protocol. The NETCONF and RESTCONF protocols install these filters into forwarding tables. The I2RS protocol uses the ACLs as part of the filters installed in an ephemeral protocol-independent filter-based RIB [I-D.kini-i2rs-fb-rib-info-model] which controls the flow of traffic on interfaces specifically controlled by the I2RS filter-based FIB.

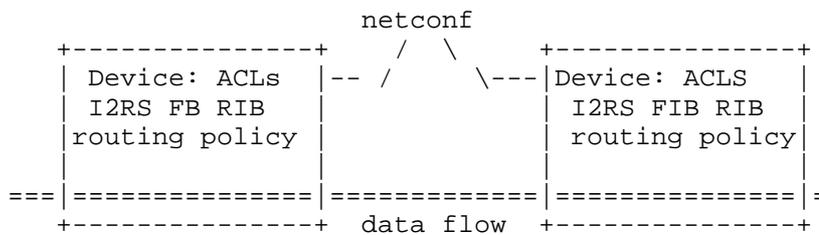


Figure 1

The I2RS protocol is a programmatic interface to the routing system. At this time, the I2RS is targeted to be extensions to the NETCONF/RESTCONF protocols to allow the NETCONF/RESTCONF protocol to support a highly programmatic interface with high bandwidth of data, highly reliable notifications, and ephemeral state (see

[I-D.ietf-i2rs-architecture]). See Section 7.2 on I2RS for additional details on I2RS.

The vocabulary of the [I-D.ietf-netmod-acl-model] is limited, so additional protocol independent filters has been written for the I2RS Filter-Based RIBs in [I-D.hares-i2rs-pkt-eca-data-model].

One thing important to note is that NETCONF and RESTCONF manage device layer YANG models. However, as Figure 2 shows, there are multiple device level, network-wide level, and application level YANG modules. The access lists defined by the device level forwarding table may be impacted by the routing protocols, the I2RS ephemeral protocol independent Filter-Based FIB, or some network-wide security issue (IPS/IDS).

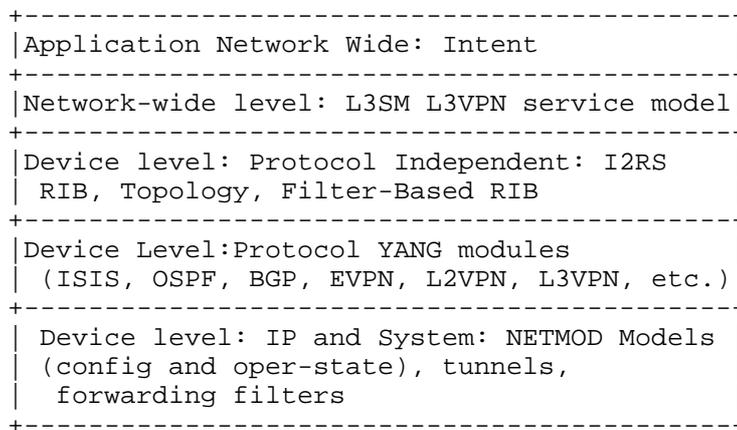


Figure 2 Levels of YANG modules

2.1.1.2. I2NSF Gap analysis

The gap is that none of the current work on these filters considers all the variations of data necessary to do IPS/IDS, web-filters, stateful flow-based filtering, security-based deep packet inspection, or pattern matching with re-mediation. The I2RS Filter-Based RIB work is the closest associated work, but the focus has not been on IDS/IPS, web-filters, security-based deep packet inspection, or pattern matching with re-mediation.

The I2RS Working group (I2RS WG) is focused on the routing system so the requisite security expertise for such NSFs (IDP/IPS, Web-filter, security-based deep-packet inspection, etc.) has not been targeted for this WG.

Another gap is there is no capability registry (an IANA registry) that identifies the characteristics and behaviours of NSFs in vendor-neutral vocabulary without requiring the NSFs to be standardized.

What I2NSF can use from NETCONF/RESTCONF and I2RS

I2NSF should consider using NETCONF/RESTCONF protocol and the I2RS proposed enhancement to the NETCONF/RESTCONF protocol.

2.1.2. Middle-box Filters

2.1.2.1. Midcom

Midcom Summary: MIDCOM developed the protocols for applications to communicate with middle boxes. However, MIDCOM have not been used by the industry for a long time. A main reason is that MIDCOM had a lot of IPR encumbered technology and IPR was likely a bigger problem for IETF at that time than it is today. MIDCOM is not specific to SIP. It was very much oriented to NAT/FW devices. SIP was just one application that needed the functionality. MIDCOM is reservation-oriented and there was an expectation that the primary deployment environment would be VoIP and real-time conferencing, including SIP, H.323, and other reservation-oriented protocols. There was an assumption that there would be some authoritative service that would have a view into endpoint sessions and be able to authorize (or not) resource allocation requests. In other words, there is a trust model in MIDCOM that may not be applicable to endpoint-driven requests without some sort of trusted authorization mechanisms/tools. Therefore, there is a specific information model applied to security devices, and security device requests, that was developed in the context of an SNMP MIB. There is also a two-stage reservation model, which was specified in order to allow better resource management.

Why I2NSF is Different from Midcom

MIDCOM is different from I2NSF because its SNMP scheme does not work with the virtual network security functions (vNSF) management.

MidCom RFCs:

[RFC3303] - Midcom architecture

[RFC5189] - Midcom Protocol Semantics

[RFC3304] - Midcom protocol requirements

2.1.3. Security Work

2.1.3.1. Overview

Today's NSFs in security devices can handle flow-based security by providing treatment to packets/flows, such as IPS/IDS, Web filtering, flow filtering, deep packet inspection, or pattern matching and remediation. These flow-based security devices are managed and provisioned by network management systems.

No standardized set of interoperable interfaces control and manage the NSFs so that a central management system can be used across security devices from multiple Vendors. I2NSF work plan is to standardize a set of interfaces by which control and management of NSFs may be invoked, operated, and monitored by:

Creating an information model that defines concepts required for standardizing the control and monitoring of NSFs, and from the information model create data models. (The information model will be used to get early agreement on key technical points.)

Creating a capability registry (at IANA) that enables the characteristics and behavior of NSFs to be specified using a vendor-neutral vocabulary without requiring the NSFs themselves to be standardized.

Defining the requirements for an I2NSF protocol to pass this traffic. (Ideally by re-using existing protocols.)

The flow-filtering configuration and management must fit into the existing security area's work plan. This section considers how the I2NSF fits into the security area work under way in the SACM (Security Automation and Continuous Monitoring), DOTS (DDoS Open Threat Signalling), and MILE (Management Incident Lightweight Exchange).

2.1.3.2. Security Work and Filters

In the proposed I2NSF work plan, the I2NSF security network management system controls many NSF nodes via the I2NSF Agent. This control of data flows is similar to the COPS example in Section 7.4.

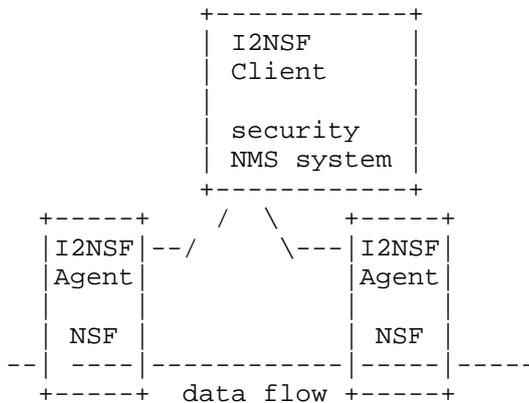


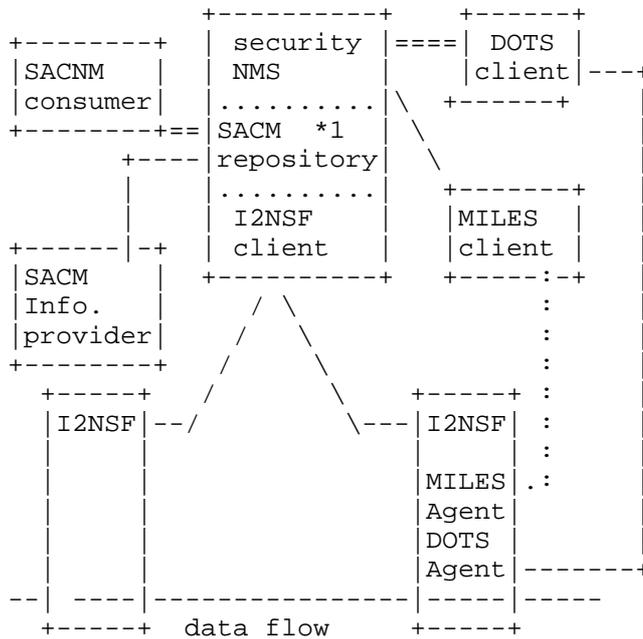
Figure 2

The other security protocols work to interact within the network to provide additional information in the following way:

- o SACM [I-D.ietf-sacm-architecture] describes an architecture which tries to determine if the end-point security policies and the reality (denoted as security posture) align. [I-D.ietf-sacm-terminology] defines posture as the configuration and/or status of hardware or software on an endpoint as it pertains to an organization’s security policy. Filters can be considered on the configuration or status pieces that needs to be monitored.
- o DOTS (DDoS Open Threat Signalling) - is working on coordinating the mitigation of DDoS attacks. A part of DDoS attach mitigation is to provide lists of addresses to be filtered via IP header filters.
- o MILE (Managed Incident Lightweight Exchange) - is working on creating a standardized format for incident and indicator reports, and creating a protocol to transport this information. The incident information MILE collects may cause changes in data-flow filters on one or more NSFs.

2.1.3.3. I2NSF interaction

The network management system that the I2NSF client resides on may interact with other clients or agents developed for the work ongoing in the SACM, DOTS, and MILES working groups. This section describes how the addition of I2NSF’s ability to control and monitor NSF devices is compatible and synergistic with these existing efforts.



*1 - this is the SACM Controller (CR) with its broker/proxy/repository show as described in the SACM architecture.

Figure 3

Figure 3 provides a diagram of a system in which the I2NSF, SACM, DOTS and MILE client-agent or consumer-broker-provider are deployed together. The following are possible positive interactions these scenario might have:

- o An security network management system (NMS) can contain a SACM repository and be connected to SACM information providers and SACM consumers. The I2NSF may provide one of the ways to change the forwarding filters.
- o The security NMS may also be connected to DOTS DDoS clients managing the information and configuring the rules. The I2NSF may provide one of the ways to change forwarding filters.
- o The MILE client on a security network management system talking to the MILE agent on the node may react to the incidents by using I2NSF to set filters. DOTS creates black-lists, but does not have a complete set of filters.

2.1.3.4. Benefits from the Interaction

I2NSF's ability to provide a common interoperable and vendor neutral interface may allow the security NMS to use a single change to change filters. SACM provides an information model to describe end-points, but does not link this directly to filters.

DOTS creates black-lists based on source and destination IP address, transport port number, protocol ID, and traffic rate. Like ACLs defined NETMOD, the DOTS black-lists are not sufficient for all filters or control desired by the NSF boxes.

The incident data captured by MILE will not have enough filter information to provide NSF devices with general services. The I2NSF will be able to handle the MILE incident data and create alerts or reports for other security systems.

3. ETSI NFV

3.1. ETSI Overview

Network Functions Virtualization (NFV) provides service providers with flexibility, cost effective and agility to offer their services to customers. One such service is the network security function which guards the exterior of a service provider or its customers. However, the exterior network beyond the service provider NSFs or its customer's NSFs is becoming extremely narrow as NSFs are becoming more pervasive in any portion of networks (service providers, customers, or access networks).

The flexibility and agility of NFV encourages service providers to provide different products to address business trends in their market to provide better service offerings to their end user. A traditional product such as the network security function (NSF) may be broken into multiple virtual devices each hosted from another vendor. In the past, network security devices may have been sourced from a small set of vendors - but in the NFV version of NSF devices, this reduced set of sources will not provide a competitive edge. Due to this market shift, the network security vendors are realizing that the proprietary provisioning protocols and formats of data may be a liability. Out of the NFV work has arisen a desire for a single interoperable network security device provisioning and control protocol.

The I2NSF framework is complementary to the NFV and other security frameworks. The I2NSF management protocol will be deployed in networks to provide a common management protocol to manage NSF software/devices whether the devices are physical or virtual. The

ETSI NFV security is also deployed along-side other security functions (AAA, SACM, DOTS, or MILE devices) and deep-packet stateful inspections.

The ETSI Network Functions Virtualization: NFV security: Security and Trust Guidance document (ETSI NFV SEC 003 1.1.1 (2014-12)) indicates that multiple administrative domains will be deployed in carrier networks. One example of these multiple domains is hosting of multiple tenant domains (telecom service providers) on a single infrastructure domain (infrastructure service) as Figure 4 shows. The ETSI Inter-VNFM document (aka Ve-Vnfn) between the element management system and the Virtual network function is the equivalent of the interface between the I2NSF client on a management system and the I2NSF agent on the network security feature VNF.

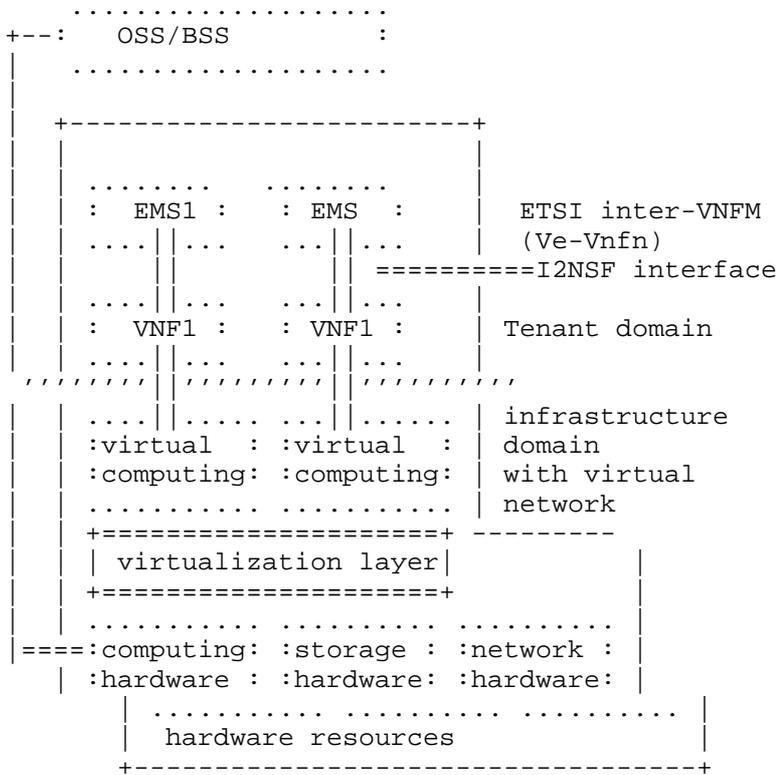


Figure 4

The ETSI proof-of-concept demonstrations have been done for the security proof of concepts:

- o #16 - NFVIaaS with Secure, SDN controlled WAN Gateway,

3.2. I2NSF Gap Analysis

The I2NSF protocol/interface can be deployed for security devices along-side the network/host configuration done by NETCONF/RESTCONF or the routing system interface provided by I2RS that handles.

In the current NFV-related architecture, there is no interoperable protocol defined between a security manager and various NSF devices to provision security functions. The result is that service providers have to manage the interoperability security manager and different NSF devices using proprietary protocols. In response to this problem, the device manufacturers and the service providers have begun to discuss an I2NSF protocol for interoperable passing of provisioning and filter information.

Open source work (such as OPNFV) provides a common code base on which providers may start their NFV development work. However, this code base faces the same problem. There is no defacto standard protocol.

4. OPNFV

The OPNFV (www.opnfv.org) is a carrier-grade integrated, open source platform focused on accelerating the introduction of new Network Functions Virtualization (NFV) products and service. The OPNFV Moon project is focused on adding the security interface for a network management system within the tenant NFVs and the infrastructure NFVs (as shown in Figure 4). This section provides an overview of the OPNFV Moon project and a gap analysis between I2NSF and the OPNFV Moon Project.

4.1. OPNFV Moon Project

The OPNFV Moon project (<https://wiki.opnfv.org>) is a security management system. NFV uses cloud computing technologies to virtualize the resources and automate the control. The Moon project is working on a security manager for the cloud computing infrastructure (<https://wiki.opnfv.org/moon>). The Moon project proposes to provision a set of different cloud resources/services for VNFs (Virtualized Network Functions) while managing the isolation of VNS, protection of VNFs, and monitoring of VNS. Moon is creating a security management system for OPNFV with security managers to protect different layers of the NFV infrastructure. The Moon project is choosing various security project mechanisms "a la carte" to enforcement related security managers. A security management system integrates mechanisms of different security aspects. This project intends propose a security manager that specifies users' security

requirements. It will also enforce the security managers through various mechanisms like authorization for access control, firewall for networking, isolation for storage, logging for tractability, etc.

The Moon security manager operates a VNF security manager at the ETSI VeVnfm level where the I2NSF protocol is targeted as Figure 5 shows. This figure also shows how the OPNFV VNF Security project mixes the I2NSF level with the device level.

The Moon project lists the following gaps in OpenStack:

- o No centralized control for compute, storage, and networking. Open Stack uses Nova for compute and Swift for object storage. Each system has a configuration file and its own security policy. The system lacks a synchronization mechanism to build a complete secure configuration for OPNFV.
- o No dynamic control so that if a user obtains the token, so there is no way to obtain control over the user.
- o No customization or flexibility to allow integration into different vendors,
- o No fine grained authorization at user level. Authorization is only at the API level.

Moon addresses these issues adding authorization, logging, IDS, enforcement of network policy, and storage protection. Moon release C (2016) plans to:

- o Define an identity federation scenario between OpenStack and OpenDaylight,
- o Implement an authentication driver in ODL to delegate authentication to OpenStack/Keystone,
- o Implement a command line tool for administration and testing,
- o Implement a graphic interface for identity management for both OpenStack and OpenDaylight,
- o Set up identity federation testbed,
- o Define identity federation scenarios with other SDN controllers, and
- o Define authorization federation scenarios with OpenDaylight.

Deliverable time frame: Moon Release 3 (mid-year 2016)

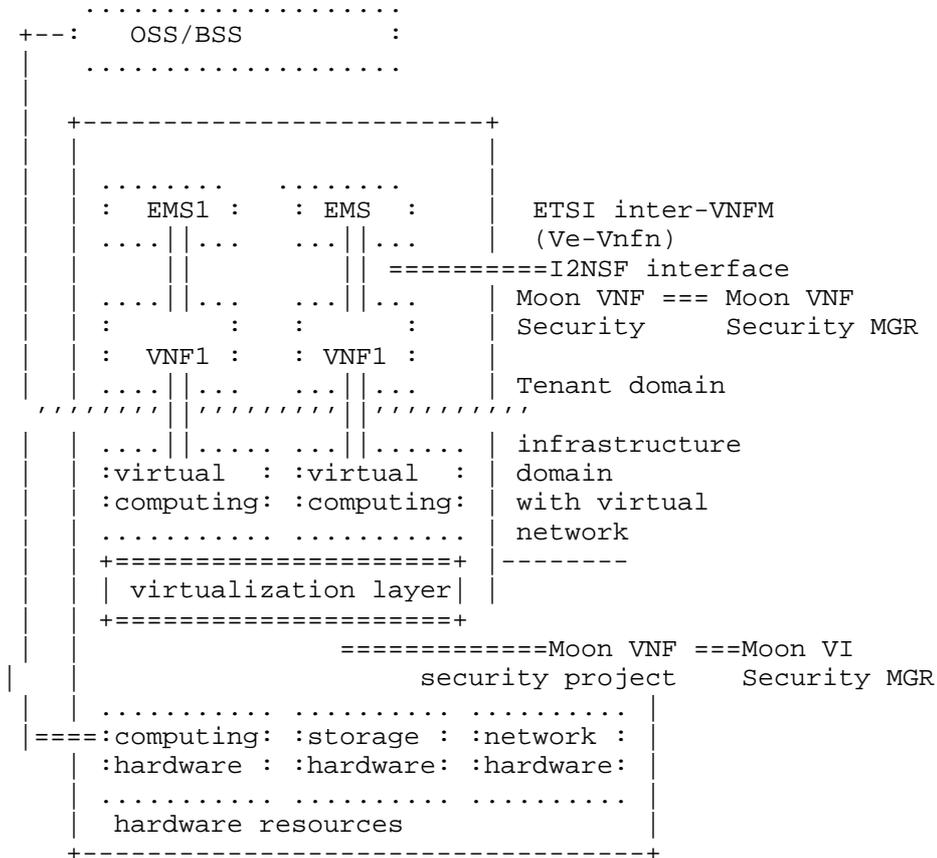


Figure 5

4.2. Gap Analysis for OPNFV Moon Project

OpenStack Congress does not provide vendor independent systems.

5. OpenStack Security Firewall

OpenStack has advanced features of: a) API for managing security groups (http://docs.openstack.org/admin-guide-cloud/content/section_securitygroups.html) and b) firewalls as a service (http://docs.openstack.org/admin-guide-cloud/content/fwaas_api_abstractions.html).

This section provides an overview of this open stack work, and a gap analysis of how I2NSF provides additional functions

5.1. Overview of API for Security Group

The security group rules provide ingress and egress traffic filters based on port. The default rule for the group policy drops all ingress traffic and allows all egress traffic. The group policy allows users to add additional groups with additional filters that change the default behaviour. To utilize the security groups, the networking plug-in for OpenStack must implement the security group API. The following plug-ins in OpenStack currently implement this security: ML2, Open vSwitch, Linux Bridge, NEC, and VMware NSX. In addition, the correct firewall driver must be added to make this functional.

5.2. Overview of Firewall as a Service

Firewall as a service is an early release of an API that allows early adopters to test network implementations. It contains APIs with parameters for firewall rules, firewall policies, and firewall identifiers. The firewall rules include the following information:

- o identification of rule (id, name, description)
- o identification tenant rule associated with,
- o links to installed firewall policy,
- o IP protocol (TCP, UDP, ICMP, or none)
- o source and destination IP address
- o source and destination port
- o action: allow or deny traffic
- o status: position and enable/disabled

The firewall policies include the following information:

- o identification of the policy (id, name, description),
- o identification of tenant associated with,
- o ordered list of firewall rules,
- o indication if policy can be seen by tenants other than owner, and

- o indication if firewall rules have been audited.

The firewall table provides the following information:

- o identification of firewall (id, name, description),
- o tenant associated with this firewall,
- o administrative state (up/down),
- o status (active, down, pending create, pending delete, pending update, pending error)
- o firewall policy ID this firewall is associated with

5.3. I2NSF Gap analysis

The OpenStack work is preliminary (security groups and firewall as a service). This work does not allow any of the existing network security vendors provide a management interface. The OpenStack work provides an interesting simple set of filters, and may in the future provide some virtual filter service. However, at this time this open source work does not address the need for a single management interfaces for a variety of security devices.

Phase 1 of I2NSF is proposes rules that will include Event-Condition-Action matches (ECA) rules with:

- packet based matches on L2, L3, and L4 headers and/or specific addresses within these headers, and

- context based matches on schedule state and schedule.

- basic ations of deny, permit, and mirror,

- advanced actions of: IPS signature filtering and URL filtering.

[Editorial note: do we need more matches or actions?]

6. CSA Secure Cloud

6.1. CSA Overview

The Cloud Security Alliance (CSA)(www.cloudsecurityalliance.org) defined security as a service (SaaS) in their Security as a Service working group (SaaS WG) during 2010-2012. The CSA SaaS group defined ten categories of network security (<https://downloads.cloudsecurityalliance.org/initiatives/secaas/>

SecaaS_V1_0.pdf) and provides implementation guidance for each of these ten categories. This section provides an overview of the CSA SaaS working groups documentation and a gap analysis for I2NSF

6.1.1. CSA Security as a Service (SaaS)

The CSA SaaS working group defined the following ten categories, and provided implementation guidance on these categories:

1. Identity Access Management (IAM)
(https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_1_IAM_Implementation_Guidance.pdf)
2. Data Loss Prevention (DLP)
(https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_2_DLP_Implementation_Guidance.pdf)
3. Web Security (web)
(https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_3_Web_Security_Implementation_Guidance.pdf),
4. Email Security (email)
(https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_4_Email_Security_Implementation_Guidance.pdf),
5. Security Assessments
(https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_5_Security_Assessments_Implementation_Guidance.pdf),
6. Intrusion Management
(https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_6_Intrusion_Management_Implementation_Guidance.pdf),
7. Security information and Event Management
(https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_7_SIEM_Implementation_Guidance.pdf),
8. Encryption
(https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_8_Encryption_Implementation_Guidance.pdf),
9. Business Continuity and Disaster Recovery (BCDR)
https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_9_BCDR_Implementation_Guidance.pdf), and
10. Network Security
(https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_10_Network_Security_Implementation_Guidance.pdf).

The sections below give an overview these implementation guidelines.

6.1.2. Identity Access Management (IAM)

document:
(https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_1_IAM_Implementation_Guidance.pdf)

The identity management systems include the following services:

- o Centralized Directory Services,
- o Access Management Services,
- o Identity Management Services,
- o Identity Federation Services,
- o Role-Based Access Control Services,
- o User Access Certification Services,
- o Privileged User and Access Management,
- o Separation of Duties Services, and
- o Identity and Access Reporting Services.

The IAM device communications with the security management system that controls the filtering of data. The CSA SaaS IAM specification states that interoperability between IAM devices and secure access network management systems is a problem. This 2012 implementation report confirms there is a gap with IAM.

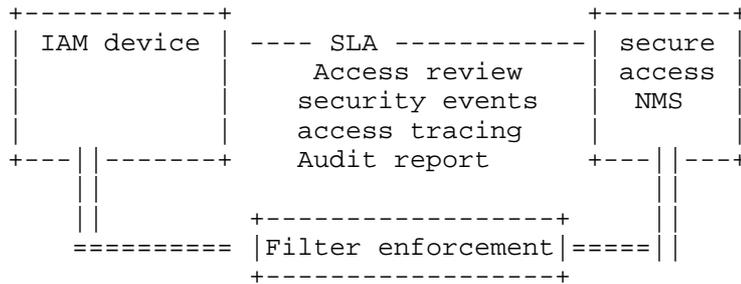


Figure 6

6.1.4. Web Security (Web)

Document:

https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_3_Web_Security_Implementation_Guidance.pdf

The web security services must address:

- o Web 2.0/Social Media controls,
- o Malware and Anti-Virus controls,
- o Data Loss Prevention controls (over Web-based services like Gmail or Box.net),
- o XSS, JavaScript and other web specific attack controls
- o Web URL Filtering,
- o Policy control and administrative management,
- o Bandwidth management and quality of service (QoS) capability, and
- o Monitoring of SSL enabled traffic.

The CSA SaaS Web services device communications require that it have the enforcement capabilities to do the following:

- alert and log malware or anti-virus data patterns,
- delete data (malware and virus) passing through systems,
- filter out (block/quarantine) data,
- filter Web URLs,
- interact with policy and network management systems,
- control bandwidth and QoS of traffic, and
- monitor encrypted (SSL enabled) traffic,

All of these features either require the I2NSF standardized I2NSF client to I2NSF agent to provide multi-vendor interoperability.

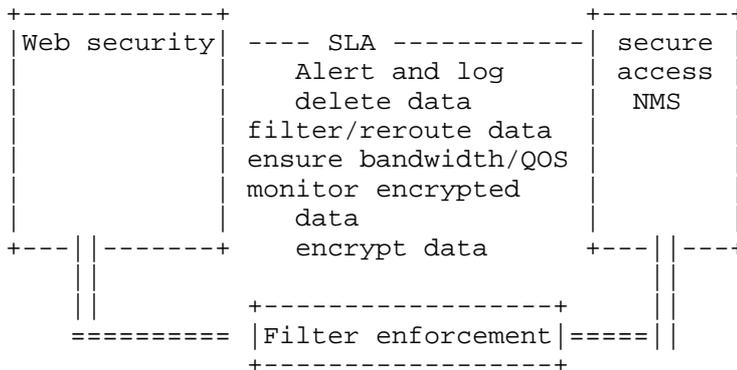


Figure 8

6.1.5. Email Security (email))

Document:

https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_4_Email_Security_Implementation_Guidance.pdf

The CSA Document recommends that email security services must address:

- o Common electronic mail components,
- o Electronic mail architecture protection,
- o Common electronic mail threats,
- o Peer authentication,
- o Electronic mail message standards,
- o Electronic mail encryption and digital signature,
- o Electronic mail content inspection and filtering,
- o Securing mail clients, and
- o Electronic mail data protection and availability assurance techniques

The CSA SaaS Email security services requires that it have the enforcement capabilities to do the following:

provide the malware and spam detection and removal,

- o network security compliance systems,
- o Servers and workstations,
- o applications,
- o network vulnerabilities systems,
- o internal auditor and intrusion detection/prevention systems (IDS/IPS), and
- o web application systems.

All of these features require the I2NSF working group standardize the way to pass these assessments to and from the I2NSF client on the I2NSF management system and the I2NSF Agent.

6.1.1.7. Intrusion Detection

Document:

https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_6_Intrusion_Management_Implementation_Guidance.pdf

The CSA SaaS Intrusion detection management includes intrusion detection through: devices:

- o Network traffic inspection, behavioural analysis, and flow analysis,
- o Operating System, Virtualization Layer, and Host Process Events monitoring,
- o Monitoring of Application Layer Events, and
- o Correlation Techniques, and other Distributed and Cloud-Based Capabilities

Intrusion response includes both:

- o Automatic, Manual, or Hybrid Mechanisms,
- o Technical, Operational, and Process Mechanisms.

The CSA SaaS recommends the intrusion security management systems include provisioning and monitoring of all of these types of intrusion detection or intrusion protection devices. Management of these systems requires:

A SIEM system would be a prime candidate to have an I2NSF client that gathers data from an I2NSF Agent associated with these various types of security systems. The CSA SaaS SIEM functionality document suggests that one concern is to have standards that allow timely recording and sharing of data. I2NSF can provide this.

6.1.9. Encryption

Document:

https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_8_Encryption_Implementation_Guidance.pdf

The CSA SaaS encryption implementation guidance document considers how one implements and manages the following security systems:

- Key management systems (KMS), control of keys, and key life cycle;

- Shared Secret encryption (Symmetric ciphers),

- No-Secret or Public Key Encryption (asymmetric ciphers),

- Hashing algorithms,

- Digital Signature Algorithms,

- Key Establishment Schemes,

- Protection of Cryptographic Key Material (FIPS 140-2; 140-3),

- Interoperability of Encryption Systems, Key Conferencing, Key Escrow Systems, and others

- Application of Encryption for Data at rest, data in transit, and data in use;

- PKI (including certificate revocation "CRL");

- Future application of such technologies as Homomorphic encryption, Quantum Cryptography, Identity-based Encryption, and others;

- Crypto-system Integrity (How bad implementations can under mind a crypto-system), and

- Cryptographic Security Standards and Guidelines

Encryption services typically require that security management systems be able to provision, monitor, and control the systems that are being used to encrypt data. This document indicates in the

implementation sections that the standardization of interfaces to/from management systems are key to good key management systems, encryption systems, and crypto-systems.

6.1.10. Business Continuity and Disaster Recovery (BC/DR)

Document:

https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_9_BCDR_Implementation_Guidance.pdf

The CSA SaaS Business Continuity and Disaster Recovery (BC/DR) implementation guidance document considers the systems that implement the contingency plans and measures designed and implemented to ensure operational resiliency in the event of any service interruptions. BC/DR systems includes:

Business Continuity and Disaster Recovery BC/DR as a Service, including categories such as complete Disaster Recovery as a Service (DRaaS), and subsets such as file recovery, backup and archive,

Storage as a Service including object, volume, or block storage;

Cold Site, Warm Site, Hot Site backup plans;

IaaS (Infrastructure as a Service), PaaS (Platform as a Service), and SaaS (Software as a Service);

Insurance (and insurance reporting programs)

Business Partner Agents (business associate agreements);

System Replication (for high availability);

Fail-back to Live Systems mechanisms and management;

Recovery Time Objective (RTO) and Recovery Point Objective (RPO);

Encryption (data at rest [DAR], data in motion [DIM], field level);

Realm-based Access Control;

Service-level Agreements (SLA); and

ISO/IEC 24762:2008, BS25999, ISO 27031, and FINRA Rule 4370

These BC/DR systems must handle data backup and recovery, server backup/recovery, and data center (virtual/physical) backup and recovery. Recovery as a Service (RaaS) means that the BC/DR services are being handled by management systems outside the enterprise.

BC/DR requires security management systems to be able to communicate provisioning, monitor, and control those systems that are being used to back-up and restore data. An interoperable protocol that allows provision and control of data center's data, servers, and data center management devices is extremely important to this application. Recovery as a Service (SaaS) indicates that these services need to be able to be remotely management.

The CSA SaaS BC/BR documents indicate how important a standardized I2NSF protocol is.

6.1.11. Network Security Devices

Document:

https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_10_Network_Security_Implementation_Guidance.pdf

The CSA SaaS Network Security implementation recommendation includes advice on:

- How to segment networks,

- Network security controls,

- Controlling ingress and egress controls such as Firewalls (Stateful), Content Inspection and Control (Network-based), Intrusion Detection System/Intrusion Prevention Systems (IDS/IPS), and Web Application Firewalls,

- Secure routing and time,

- Denial of Service (DoS) and Distributed Denial of Service (DDoS) Protection/Mitigation,

- Virtual Private Network (VPN) with Multiprotocol Label Switching (MPLS) Connectivity (over SSL), Internet Protocol Security (IPsec) VPNs, Virtual Private LAN Service (VPLS), and Ethernet Virtual Private Line (EVPL),

- Threat Management,

- Forensic Support, and

Privileged User/Use Monitoring.

These network security systems require provisioning, monitoring, and the ability for the security management system to subscribe to receive logs, snapshots of capture data, and time synchronization. This document states the following:

"It is critical to understand what monitoring APIs are available from the CSP, and if they match risk and compliance requirements",

"Network security auditors are challenged by the need to track a server and its identity from creation to deletion. Audit tracking is challenging in even the most mature cloud environments, but the challenges are greatly complicated by cloud server sprawl, the situation where the number of cloud servers being created is growing more quickly than a cloud environments ability to manage them."

A valid threat vector for cloud is the API access. Since a majority of CSPs today support public API interfaces available within their networks and likely over the Internet."

The CSA SaaS network security indicates that the I2NSF must be secure so that the I2NSF Client-Agent protocol does not become a valid threat vector. In additions, the need for the management protocol like I2NSF is critical in the sprawl of Cloud environment.

6.2. I2NSF Gap Analysis

The CSA Security as a Service (SaaS) document show clearly that there is a gap between the ability of the CSA SaaS devices to have a vendor neutral, inoperable protocol that allow the multiple of network security devices to communicate passing provisioning and informational data. Each of the 10 implementation agreements points to this as a shortcoming. Standard I2NSF YANG models and an I2NSF protocol are needed according to the CSA SaaS documents.

7. IEEE security

7.1. Port-based Network Access Control [802.1X]

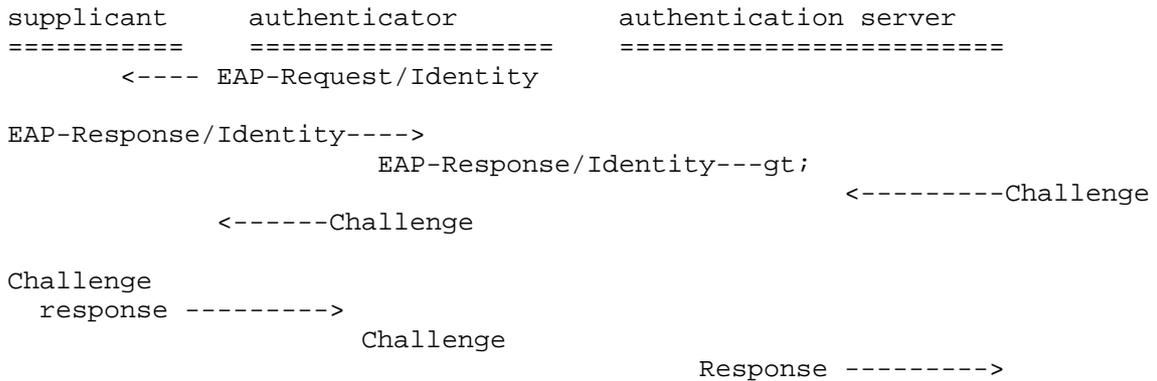
802.1x defines encapsulation of Extensible Authentication Protocol (EAP) [RFC3748] over IEEE 802 (EAP over LAN, or EAPOL). It is widely deployed on both wired and Wi-Fi Networks.

EAP provides support for security from passwords to challenge-response tokens and public-key infrastructure certificates.

802.1 has three concepts:

- o the supplicant - the user or client who wants to be authenticated
- o authentication server - the server doing the authentication (e.g. radius server), and
- o the authenticator - the device in-between authentication server and supplicant (e.g. wireless Access Point (AP)).

A normal sequence is below:



Gap:

This basic service provides access, but today's access use cases are more complex. IEEE 801.X has been attacked using the Man-in-the-middle attacks. Another weakness of 802.1X is the speed of the EAP protocols processing with the radius server.

Note: Editor - more is needed here

7.2. MAC security (802.1AE)

MACsec security secures a link rather than a conversation for 802.1 LANs (VLANs 802.1Q, Provider Bridges 802.1AD). MACsec counters the 802.1X man-in the middle attacks.

MACsec (in 802.1AE) provides MAC-layer encryption over wired networks by using out-of-band methods for encryption keying. The MACsec Key Agreement (MKA) Protocol provides the required session keys and manages the required encryption keys. MKA and MACsec are implemented after successful authentication using the 802.1x Extensible Authentication Protocol (EAP) framework. Only hosts link which face

the network can be secured with MACSec. These links connect the host to the network access devices.

Switch using MACsec accepts either MACsec or non-MACsec frames based on policy set. The NSF controller can set within the switches configuration whether MACSec frames are accepted. Accepted MACsec frames are encrypted and protected with an integrity check value (ICV). The switch after receiving frames from the client, decrypts them and calculates the correct ICV by using session keys provided by MKA. The switch compares that ICV to the ICV within the frame. If they are not identical, the frame is dropped. The switch also encrypts and adds an ICV to any frames sent over the secured port (the access point used to provide the secure MAC service to a client) using the current session key.

The MKA Protocol manages the encryption keys used by the underlying MACsec protocol. The basic requirements of MKA are defined in 802.1x-REV. The MKA Protocol extends 802.1x to allow peer discovery with confirmation of mutual authentication and sharing of MACsec secret keys to protect data exchanged by the peers. MKA protocol uses EAP-over-LAN (EAPOL) packet. EAP authentication produces a master session key (MSK) shared by both partners in the data exchange. Entering the EAP session ID generates a secure connectivity association key name (CKN). Because the switch is the authenticator, and the key server, it can generate a random 128-bit secure association key (SAK), which it sends it to the client partner. The client is never a key server and can only interact with a single MKA entity, the key server. After key derivation and generati

Gap Analysis:

I2NSF Devices must handle the existence of MSEC within the network.

7.3. Secure Device Identity [802.1AR]

802.1AR does the following:

- Supports trail of trust from manufacturer to user, and

- Defines how a Secure Device Identifier (DevId) may be cryptographically bound to a device to support device identity authentication.

- DevIDs are composed of a secure device identifier secret and a secure device identifier credential. These IDs can be controlled by the product manufacturer (IDevID, an initially installed identity) or by the end-user (LDevID, a subsequent locally

significant identity derived from the IDevID). DevIDs cannot be changed by the end-user.

One attack mitigation can be to disable support for DeVIDs or limit to know DeVIDs.

GAP analysis:

I2NSF controllers need to support 802.1AR device management.

8. In-depth Review of IETF protocols

8.1. NETCONF and RESTCONF

The IETF NETCONF working group has developed the basics of the NETCONF protocol focusing on secure configuration and querying operational state. The NETCONF protocol [RFC6241] may be run over TLS [RFC6639] or SSH ([RFC6242]. NETCONF can be expanded to defaults [RFC6243], handling events ([RFC5277] and basic notification [RFC6470], and filtering writes/reads based on network access control models (NACM, [RFC6536]). The NETCONF configuration must be committed to a configuration data store (denoted as config=TRUE). YANG models identify nodes within a configuration data store or an operational data store using a XPath expression (document root ---to --- target source). NETCONF uses an RPC model and provides protocol for handling configs (get-config, edit-config, copy-config, delete-config, lock, unlock, get) and sessions (close-session, kill-session). The NETCONF Working Group has developed RESTCONF, which is an HTTP-based protocol that provides a programmatic interface for accessing data defined in YANG, using the data stores defined in NETCONF.

RESTCONF supports "two edit condition detections" - time stamp and entity tag. RESTCONF uses URI encoded path expressions. RESTCONF provides operations to get remote servers options (OPTIONS), retrieve data headers (HEAD), get data (GET), create resource/invoke operation (POST), patch data (PATCH), delete resource (DELETE), or query.

RFCs for NETCONF

- o NETCONF [RFC6242]
- o NETCONF monitoring [RFC6022]
- o NETCONF over SSH [RFC6242]
- o NETCONF over TLS [RFC5539]

- o NETCONF system notification> [RFC6470]
- o NETCONF access-control (NACM) [RFC6536]
- o RESTCONF [I-D.ietf-netconf-restconf]
- o NETCONF-RESTCONF call home [I-D.ietf-netconf-call-home]
- o RESTCONF collection protocol
[I-D.ietf-netconf-restconf-collection]
- o NETCONF Zero Touch Provisioning [I-D.ietf-netconf-zerotouch]

8.2. I2RS Protocol

Based on input from the NETCONF working group, the I2RS working group decided to re-use the NETCONF or RESTCONF protocols and specify additions to these protocols rather than create yet another protocol (YAP).

The required extensions for the I2RS protocol are in the following drafts:

- o [I-D.ietf-i2rs-ephemeral-state],
- o [I-D.ietf-i2rs-pub-sub-requirements] (Publication-Subscription notifications,
- o [I-D.ietf-i2rs-traceability]
- o [I-D.ietf-i2rs-protocol-security-requirements]

At this time, NETCONF and RESTCONF cannot handle the ephemeral data store proposed by I2RS, the publication and subscription requirements, the traceability, or the security requirements for the transport protocol and message integrity.

8.3. NETMOD YANG modules

NETMOD developed initial YANG models for interfaces [RFC7223]), IP address ([RFC7277]), IPv6 Router advertisement ([RFC7277]), IP Systems ([RFC7317]) with system ID, system time management, DNS resolver, Radius client, SSH, syslog ([I-D.ietf-netmod-syslog-model]), ACLS ([I-D.ietf-netmod-acl-model]), and core routing blocks ([I-D.ietf-netmod-routing-cfg] The routing working group (rtgwg) has begun to examine policy for routing and tunnels.

Protocol specific Working groups have developed YANG models for ISIS ([I-D.ietf-isis-yang-isis-cfg]), OSPF ([I-D.ietf-ospf-yang]), and BGP ([I-D.ietf-idr-bgp-model]).

BGP Services YANG models have been proposed for

- o EVPN [I-D.brissette-bess-evpn-yang],
- o L2VPN [I-D.shah-bess-l2vpn-yang],
- o L3VPN [I-D.li-bess-l3vpn-yang] and [I-D.hu-bess-l2vpn-service-yang],

TEAS working group has proposed [I-D.ietf-teas-yang-te-topol], and [I-D.ietf-teas-yang-rsvp].

MPLS/PCE/CCAMP groups have proposed the following Yang modules:

- o [I-D.raza-mpls-ldp-mldp-yang]
- o [I-D.saad-mpls-static-yang],
- o [I-D.zheng-mpls-lsp-ping-yang-cfg],
- o [I-D.pkd-pce-pcep-yang], and
- o [I-D.zhang-ccamp-transport-ctrlnorth-yang].

8.4. COPS

One early focus on flow filtering based on policy enforcement of traffic entering a network is the 1990s COPS [RFC2748] design (PEP and PDP) as shown in Figure 11. The COPS policy decision points (PDP) managed network-wide policy (e.g. ACLs) by installing this policy in policy enforcement points (PEPs) on the edge of the network. These PEPs had firewall-like functions that control what data flows into the network at a PEP point, and data flow out of a network at a PEP. [RFC3084] describes COPS usages for policy provisioning.

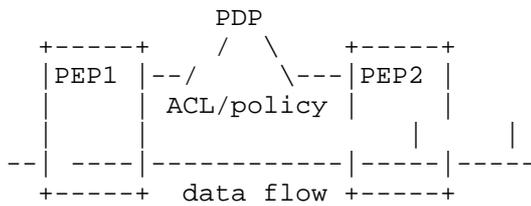


Figure 11

Why COPS is no longer used

Network security today uses specific devices (IDS/IPS, NAT firewall, etc.) with specific policies and profiles for each types of device. No common protocol or policy format exists between the policy manager (PDP) and security enforcement points.

COPS RFCs: [RFC4261], [RFC2940], [RFC3084], and [RFC3483].

Why I2NSF is Different from COPS

COPS was a protocol for policy related to Quality of Service (QoS) and signaling protocols (e.g. RSVP) (security, flow, and others). I2NSF creates a common protocol between security policy decision points (SPDP) and security enforcement points (SEP). Today's security devices currently only use proprietary protocols. Manufacturers would like a security specific policy enforcement protocol rather than a generic policy protocol.

8.5. PCP

As indicated by the name, the Port Control Protocol (PCP) enables an IPv4 or IPv6 host to flexibly manage the IP address and port mapping information on Network Address Translators (NATs) or firewalls, to facilitate communication with remote hosts.

PCP RFCs:

[RFC6887]

[RFC7225]

[I-D.ietf-pcp-authentication]

[I-D.ietf-pcp-optimize-keepalives]

[I-D.ietf-pcp-proxy]

Why is I2NSF Different from PCP:

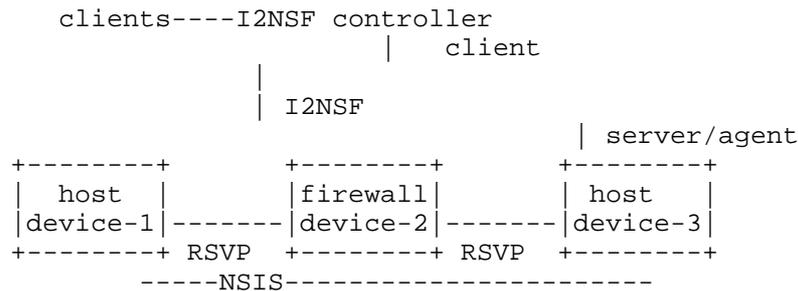
Here are some aspects that I2NSF is different from PCP:

- o PCP only supports management of port and address information rather than any other security functions

8.6. NSIS - Next Steps in Signaling

NSIS aims to standardize an IP signaling protocol (RSVP) along the data path for end points to request their unique QoS characteristics, unique FW policies or NAT needs (RFC5973) that are different from the FW/NAT original settings. The requests are communicated directly to the FW/NAT devices. NSIS is like east-west protocols that require all involved devices to fully comply to make it work.

NSIS is path-coupled; it is possible to message every participating device along a path without having to know its location, or its location relative to other devices (This is particularly a pressing issue when one or more NATs present in the network, or when trying to locate appropriate tunnel endpoints).



Why I2NSF is Different from NSIS:

- o The I2NSF request does not require all network functions in a path to comply, but it is a protocol between the I2NSF client and the I2NSF Agent/Server
- o I2NSF defines client (applications) oriented descriptors (profiles, or attributes) to request/negotiate/validate the network security functions that are not on the local premises.

Why I2NSF may have higher chance to be deployed than NSIS:

- o OpenStack already has a proof-of-concept/preliminary implementation, but the specification is not complete. IETF can

play an active role to make the specification for I2NSF is complete. IETF can complete and extend the OpenStack implementation to provide an interoperable specification that can meet the needs and requirements of operators and is workable for suppliers of the technology. The combination of a carefully designed interoperable IETF specification with an open-source code development OpenStack will leverage the strengths of the two communities, and expand the informal ties between the two groups. A software development cycle has the following components: architecture, design specification, coding, and interoperability testing. The IETF can take ownership of the first two steps, and provide expertise and a good working atmosphere (in hack-a-thons) in the last two steps for OpenStack or other open-source coders.

- o IETF has the expertise in security architecture and design for interoperable protocols that span controllers/routers, middle-boxes, and security end-systems.
- o IETF has a history of working on interoperable protocols or virtualized network functions (L2VPN, L3VPN) that are deployed by operators in large scale devices. IETF has a strong momentum to create virtualized network functions (see SFC WG in routing) to be deployed in network boxes. [Note: We need to add SACM and others here].

9. IANA Considerations

No IANA considerations exist for this document.

10. Security Considerations

No security considerations are involved with a gap analysis.

11. Contributors

The following people have contributed to this document: Hosnieh Rafiee, and Myo Zarny. Myo Zarny provided the authors with extensive comments, great suggestions, and valuable insights on alternative views.

12. References

12.1. Normative References

- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <<http://www.rfc-editor.org/info/rfc791>>.

[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>>.

12.2. Informative References

- [I-D.brissette-bess-evpn-yang]
Brissette, P., Shah, H., Li, Z., Tiruveedhula, K., Singh, T., and I. Hussain, "Yang Data Model for EVPN", draft-brissette-bess-evpn-yang-01 (work in progress), December 2015.
- [I-D.hares-i2nsf-terminology]
Hares, S., Strassner, J., Lopez, D., and L. Xia, "I2NSF Terminology", draft-hares-i2nsf-terminology-00 (work in progress), March 2016.
- [I-D.hares-i2rs-info-model-service-topo]
Hares, S., Wu, W., Wang, Z., and J. You, "An Information model for service topology", draft-hares-i2rs-info-model-service-topo-03 (work in progress), January 2015.
- [I-D.hares-i2rs-pkt-eca-data-model]
Hares, S., Wu, Q., and R. White, "Filter-Based Packet Forwarding ECA Policy", draft-hares-i2rs-pkt-eca-data-model-02 (work in progress), February 2016.
- [I-D.hu-bess-l2vpn-service-yang]
hu, f., Chen, R., and J. Yao, "L2VPN Service YANG Model", draft-hu-bess-l2vpn-service-yang-00 (work in progress), March 2016.
- [I-D.ietf-i2rs-architecture]
Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", draft-ietf-i2rs-architecture-11 (work in progress), December 2015.
- [I-D.ietf-i2rs-ephemeral-state]
Haas, J. and S. Hares, "I2RS Ephemeral State Requirements", draft-ietf-i2rs-ephemeral-state-14 (work in progress), July 2016.
- [I-D.ietf-i2rs-problem-statement]
Atlas, A., Nadeau, T., and D. Ward, "Interface to the Routing System Problem Statement", draft-ietf-i2rs-problem-statement-11 (work in progress), May 2016.

- [I-D.ietf-i2rs-protocol-security-requirements]
Hares, S., Migault, D., and J. Halpern, "I2RS Security Related Requirements", draft-ietf-i2rs-protocol-security-requirements-06 (work in progress), May 2016.
- [I-D.ietf-i2rs-pub-sub-requirements]
Voit, E., Clemm, A., and A. Prieto, "Requirements for Subscription to YANG Datastores", draft-ietf-i2rs-pub-sub-requirements-09 (work in progress), May 2016.
- [I-D.ietf-i2rs-rib-data-model]
Wang, L., Ananthakrishnan, H., Chen, M., amit.dass@ericsson.com, a., Kini, S., and N. Bahadur, "A YANG Data Model for Routing Information Base (RIB)", draft-ietf-i2rs-rib-data-model-06 (work in progress), July 2016.
- [I-D.ietf-i2rs-rib-info-model]
Bahadur, N., Kini, S., and J. Medved, "Routing Information Base Info Model", draft-ietf-i2rs-rib-info-model-09 (work in progress), July 2016.
- [I-D.ietf-i2rs-traceability]
Clarke, J., Salgueiro, G., and C. Pignataro, "Interface to the Routing System (I2RS) Traceability: Framework and Information Model", draft-ietf-i2rs-traceability-11 (work in progress), May 2016.
- [I-D.ietf-i2rs-usecase-reqs-summary]
Hares, S. and M. Chen, "Summary of I2RS Use Case Requirements", draft-ietf-i2rs-usecase-reqs-summary-01 (work in progress), May 2015.
- [I-D.ietf-i2rs-yang-l2-network-topology]
Dong, J. and X. Wei, "A YANG Data Model for Layer-2 Network Topologies", draft-ietf-i2rs-yang-l2-network-topology-03 (work in progress), July 2016.
- [I-D.ietf-i2rs-yang-network-topo]
Clemm, A., Medved, J., Varga, R., Tkacik, T., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A Data Model for Network Topologies", draft-ietf-i2rs-yang-network-topo-04 (work in progress), July 2016.

- [I-D.ietf-idr-bgp-model]
Shaikh, A., Shakir, R., Patel, K., Hares, S., D'Souza, K., Bansal, D., Clemm, A., Zhdankin, A., Jethanandani, M., and X. Liu, "BGP Model for Service Provider Networks", draft-ietf-idr-bgp-model-01 (work in progress), January 2016.
- [I-D.ietf-isis-yang-isis-cfg]
Litkowski, S., Yeung, D., Lindem, A., Zhang, J., and L. Lhotka, "YANG Data Model for ISIS protocol", draft-ietf-isis-yang-isis-cfg-02 (work in progress), March 2015.
- [I-D.ietf-l3sm-l3vpn-service-model]
Litkowski, S., Shakir, R., Tomotaki, L., Ogaki, K., and K. D'Souza, "YANG Data Model for L3VPN service delivery", draft-ietf-l3sm-l3vpn-service-model-05 (work in progress), March 2016.
- [I-D.ietf-netconf-call-home]
Watsen, K., "NETCONF Call Home and RESTCONF Call Home", draft-ietf-netconf-call-home-06 (work in progress), May 2015.
- [I-D.ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-04 (work in progress), January 2015.
- [I-D.ietf-netconf-restconf-collection]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Collection Resource", draft-ietf-netconf-restconf-collection-00 (work in progress), January 2015.
- [I-D.ietf-netconf-zerotouch]
Watsen, K., Clarke, J., and M. Abrahamsson, "Zero Touch Provisioning for NETCONF Call Home (ZeroTouch)", draft-ietf-netconf-zerotouch-02 (work in progress), March 2015.
- [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-02 (work in progress), March 2015.
- [I-D.ietf-netmod-routing-cfg]
Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-19 (work in progress), May 2015.

- [I-D.ietf-netmod-syslog-model]
Wildes, C. and K. Sreenivasa, "SYSLOG YANG model", draft-ietf-netmod-syslog-model-03 (work in progress), March 2015.
- [I-D.ietf-ospf-yang]
Yeung, D., Qu, Y., Zhang, J., Bogdanovic, D., and K. Sreenivasa, "Yang Data Model for OSPF Protocol", draft-ietf-ospf-yang-00 (work in progress), March 2015.
- [I-D.ietf-pcp-authentication]
Wasserman, M., Hartman, S., Zhang, D., and T. Reddy, "Port Control Protocol (PCP) Authentication Mechanism", draft-ietf-pcp-authentication-09 (work in progress), May 2015.
- [I-D.ietf-pcp-optimize-keepalives]
Reddy, T., Patil, P., Isomaki, M., and D. Wing, "Optimizing NAT and Firewall Keepalives Using Port Control Protocol (PCP)", draft-ietf-pcp-optimize-keepalives-06 (work in progress), May 2015.
- [I-D.ietf-pcp-proxy]
Perreault, S., Boucadair, M., Penno, R., Wing, D., and S. Cheshire, "Port Control Protocol (PCP) Proxy Function", draft-ietf-pcp-proxy-08 (work in progress), May 2015.
- [I-D.ietf-rtgwg-policy-model]
Shaikh, A., Shakir, R., D'Souza, K., and C. Chase, "Routing Policy Configuration Model for Service Provider Networks", draft-ietf-rtgwg-policy-model-00 (work in progress), September 2015.
- [I-D.ietf-sacm-architecture]
Cam-Winget, N., Lorenzin, L., McDonald, I., and l.loxx@cisco.com, "Secure Automation and Continuous Monitoring (SACM) Architecture", draft-ietf-sacm-architecture-05 (work in progress), October 2015.
- [I-D.ietf-sacm-terminology]
Birkholz, H., Lu, J., and N. Cam-Winget, "Secure Automation and Continuous Monitoring (SACM) Terminology", draft-ietf-sacm-terminology-09 (work in progress), March 2016.

[I-D.ietf-teas-yang-rsvp]

Beeram, V., Saad, T., Gandhi, R., Liu, X., Shah, H., Chen, X., Jones, R., and B. Wen, "A YANG Data Model for Resource Reservation Protocol (RSVP)", draft-ietf-teas-yang-rsvp-03 (work in progress), March 2016.

[I-D.ietf-teas-yang-te-topo]

Liu, X., Bryskin, I., Beeram, V., Saad, T., Shah, H., and O. Dios, "YANG Data Model for TE Topologies", draft-ietf-teas-yang-te-topo-05 (work in progress), July 2016.

[I-D.kini-i2rs-fb-rib-info-model]

Kini, S., Hares, S., Dunbar, L., Ghanwani, A., Krishnan, R., Bogdanovic, D., and R. White, "Filter-Based RIB Information Model", draft-kini-i2rs-fb-rib-info-model-03 (work in progress), February 2016.

[I-D.li-bess-l3vpn-yang]

Li, Z., Zhuang, S., Liu, X., Haas, J., Esale, S., and B. Wen, "Yang Data Model for BGP/MPLS IP VPN", draft-li-bess-l3vpn-yang-00 (work in progress), October 2015.

[I-D.pkd-pce-pcep-yang]

Dhody, D., Hardwick, J., Beeram, V., and j. jefftant@gmail.com, "A YANG Data Model for Path Computation Element Communications Protocol (PCEP)", draft-pkd-pce-pcep-yang-06 (work in progress), July 2016.

[I-D.raza-mpls-ldp-mldp-yang]

Raza, K., Asati, R., Liu, X., Esale, S., Chen, X., and H. Shah, "YANG Data Model for MPLS LDP and mLDP", draft-raza-mpls-ldp-mldp-yang-04 (work in progress), July 2016.

[I-D.saad-mpls-static-yang]

Saad, T., Raza, K., Gandhi, R., Liu, X., Beeram, V., Shah, H., Bryskin, I., Chen, X., Jones, R., and B. Wen, "A YANG Data Model for MPLS Static LSPs", draft-saad-mpls-static-yang-03 (work in progress), May 2016.

[I-D.shah-bess-l2vpn-yang]

Shah, H., Brissette, P., Rahman, R., Raza, K., Li, Z., Zhuang, S., Wang, H., Chen, I., Ahmed, S., Bocci, M., Hardwick, J., Esale, S., Tiruveedhula, K., tsingh@juniper.net, t., Hussain, I., Wen, B., Walker, J., Delregno, N., Jalil, L., and M. Joecylyn, "YANG Data Model for MPLS-based L2VPN", draft-shah-bess-l2vpn-yang-01 (work in progress), March 2016.

- [I-D.zhang-ccamp-transport-ctrlnorth-yang]
Zhang, X., Jing, R., Jian, W., Ryoo, J., Xu, Y., and D. King, "YANG Models for the Northbound Interface of a Transport Network Controller: Requirements, Functions, and a List of YANG Models", draft-zhang-ccamp-transport-ctrlnorth-yang-00 (work in progress), March 2016.
- [I-D.zheng-mpls-lsp-ping-yang-cfg]
Zheng, L., Aldrin, S., Zheng, G., Mirsky, G., and R. Rahman, "Yang Data Model for LSP-PING", draft-zheng-mpls-lsp-ping-yang-cfg-03 (work in progress), March 2016.
- [RFC2748] Durham, D., Ed., Boyle, J., Cohen, R., Herzog, S., Rajan, R., and A. Sastry, "The COPS (Common Open Policy Service) Protocol", RFC 2748, DOI 10.17487/RFC2748, January 2000, <<http://www.rfc-editor.org/info/rfc2748>>.
- [RFC2940] Smith, A., Partain, D., and J. Seligson, "Definitions of Managed Objects for Common Open Policy Service (COPS) Protocol Clients", RFC 2940, DOI 10.17487/RFC2940, October 2000, <<http://www.rfc-editor.org/info/rfc2940>>.
- [RFC3084] Chan, K., Seligson, J., Durham, D., Gai, S., McCloghrie, K., Herzog, S., Reichmeyer, F., Yavatkar, R., and A. Smith, "COPS Usage for Policy Provisioning (COPS-PR)", RFC 3084, DOI 10.17487/RFC3084, March 2001, <<http://www.rfc-editor.org/info/rfc3084>>.
- [RFC3303] Srisuresh, P., Kuthan, J., Rosenberg, J., Molitor, A., and A. Rayhan, "Middlebox communication architecture and framework", RFC 3303, DOI 10.17487/RFC3303, August 2002, <<http://www.rfc-editor.org/info/rfc3303>>.
- [RFC3304] Swale, R., Mart, P., Sijben, P., Brim, S., and M. Shore, "Middlebox Communications (midcom) Protocol Requirements", RFC 3304, DOI 10.17487/RFC3304, August 2002, <<http://www.rfc-editor.org/info/rfc3304>>.
- [RFC3483] Rawlins, D., Kulkarni, A., Bokaemper, M., and K. Chan, "Framework for Policy Usage Feedback for Common Open Policy Service with Policy Provisioning (COPS-PR)", RFC 3483, DOI 10.17487/RFC3483, March 2003, <<http://www.rfc-editor.org/info/rfc3483>>.
- [RFC3484] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, DOI 10.17487/RFC3484, February 2003, <<http://www.rfc-editor.org/info/rfc3484>>.

- [RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <<http://www.rfc-editor.org/info/rfc3748>>.
- [RFC4080] Hancock, R., Karagiannis, G., Loughney, J., and S. Van den Bosch, "Next Steps in Signaling (NSIS): Framework", RFC 4080, DOI 10.17487/RFC4080, June 2005, <<http://www.rfc-editor.org/info/rfc4080>>.
- [RFC4261] Walker, J. and A. Kulkarni, Ed., "Common Open Policy Service (COPS) Over Transport Layer Security (TLS)", RFC 4261, DOI 10.17487/RFC4261, December 2005, <<http://www.rfc-editor.org/info/rfc4261>>.
- [RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <<http://www.rfc-editor.org/info/rfc4949>>.
- [RFC5189] Stiemerling, M., Quittek, J., and T. Taylor, "Middlebox Communication (MIDCOM) Protocol Semantics", RFC 5189, DOI 10.17487/RFC5189, March 2008, <<http://www.rfc-editor.org/info/rfc5189>>.
- [RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, DOI 10.17487/RFC5277, July 2008, <<http://www.rfc-editor.org/info/rfc5277>>.
- [RFC5539] Badra, M., "NETCONF over Transport Layer Security (TLS)", RFC 5539, DOI 10.17487/RFC5539, May 2009, <<http://www.rfc-editor.org/info/rfc5539>>.
- [RFC5973] Stiemerling, M., Tschofenig, H., Aoun, C., and E. Davies, "NAT/Firewall NSIS Signaling Layer Protocol (NSLP)", RFC 5973, DOI 10.17487/RFC5973, October 2010, <<http://www.rfc-editor.org/info/rfc5973>>.
- [RFC6022] Scott, M. and M. Bjorklund, "YANG Module for NETCONF Monitoring", RFC 6022, DOI 10.17487/RFC6022, October 2010, <<http://www.rfc-editor.org/info/rfc6022>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<http://www.rfc-editor.org/info/rfc6241>>.

- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<http://www.rfc-editor.org/info/rfc6242>>.
- [RFC6243] Bierman, A. and B. Lengyel, "With-defaults Capability for NETCONF", RFC 6243, DOI 10.17487/RFC6243, June 2011, <<http://www.rfc-editor.org/info/rfc6243>>.
- [RFC6436] Amante, S., Carpenter, B., and S. Jiang, "Rationale for Update to the IPv6 Flow Label Specification", RFC 6436, DOI 10.17487/RFC6436, November 2011, <<http://www.rfc-editor.org/info/rfc6436>>.
- [RFC6470] Bierman, A., "Network Configuration Protocol (NETCONF) Base Notifications", RFC 6470, DOI 10.17487/RFC6470, February 2012, <<http://www.rfc-editor.org/info/rfc6470>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<http://www.rfc-editor.org/info/rfc6536>>.
- [RFC6639] King, D., Ed. and M. Venkatesan, Ed., "Multiprotocol Label Switching Transport Profile (MPLS-TP) MIB-Based Management Overview", RFC 6639, DOI 10.17487/RFC6639, June 2012, <<http://www.rfc-editor.org/info/rfc6639>>.
- [RFC6887] Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, DOI 10.17487/RFC6887, April 2013, <<http://www.rfc-editor.org/info/rfc6887>>.
- [RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <<http://www.rfc-editor.org/info/rfc7223>>.
- [RFC7225] Boucadair, M., "Discovering NAT64 IPv6 Prefixes Using the Port Control Protocol (PCP)", RFC 7225, DOI 10.17487/RFC7225, May 2014, <<http://www.rfc-editor.org/info/rfc7225>>.
- [RFC7277] Bjorklund, M., "A YANG Data Model for IP Management", RFC 7277, DOI 10.17487/RFC7277, June 2014, <<http://www.rfc-editor.org/info/rfc7277>>.
- [RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <<http://www.rfc-editor.org/info/rfc7317>>.

Authors' Addresses

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

Email: shares@ndzh.com

Bob Moskowitz
Huawei
Oak Park, MI 48237

Email: rgm@labs.htt-consult.com

Dacheng Zhang
Beijing
China

Email: dacheng.zdc@aliabab-inc.com

I2NSF
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2017

S. Hares
L. Dunbar
Huawei
D. Lopez
Telefonica I+D
M. Zarny
Goldman Sachs
C. Jacquenet
France Telecom
July 8, 2016

I2NSF Problem Statement and Use cases
draft-ietf-i2nsf-problem-and-use-cases-01.txt

Abstract

This document describes the problem statement for Interface to Network Security Functions (I2NSF) as well as some companion use cases.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 9, 2017.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	4
3. Problem Space	4
3.1. Challenges Facing Security Service Providers	5
3.1.1. Diverse Types of Security Functions	5
3.1.2. Diverse Interfaces to Control NSFs	6
3.1.3. Diverse Interface to Monitor the Behavior of NSFs	6
3.1.4. More Distributed NSFs and vNSFs	6
3.1.5. More Demand to Control NSFs Dynamically	7
3.1.6. Demand for Multi-Tenancy to Control and Monitor NSFs	7
3.1.7. Lack of Characterization of NSFs and Capability Exchange	7
3.1.8. Lack of Mechanism for NSFs to Utilize External Profiles	8
3.1.9. Lack of Mechanisms to Accept External Alerts to Trigger Automatic Configuration Changes	8
3.1.10. Lack of Mechanism for Dynamic Key Distribution to NSFs	9
3.2. Challenges Facing Customers	9
3.2.1. NSFs from Heterogeneous Administrative Domains	10
3.2.2. Today's Control Requests are Vendor Specific	10
3.2.3. Difficulty to Monitor the Execution of Desired Policies	12
3.3. Difficulty to Validate Policies across Multiple Domains	12
3.4. Lack of Standard Interface to Inject Feedback to NSF	13
3.5. Lack of Standard Interface for Capability Negotiation	13
4. Use Cases	13
4.1. Basic Framework	13
4.2. Access Networks	15
4.3. Cloud Data Center Scenario	16
4.3.1. On-Demand Virtual Firewall Deployment	17
4.3.2. Firewall Policy Deployment Automation	18
4.3.3. Client-Specific Security Policy in Cloud VPNs	18
4.3.4. Internal Network Monitoring	19
4.4. I2NSF Preventing Distributed DoS in Overlay Networks	19
5. Management Considerations	20
6. IANA Considerations	21
7. Security Considerations	21
8. Contributors	21
9. Contributing Authors	21
10. References	22

10.1. Normative References 22
 10.2. Informative References 22
 Authors' Addresses 23

1. Introduction

This document describes the problem statement for Interface to Network Security Functions (I2NSF) as well as some I2NSF use cases. A summary of the state of the art in the industry and IETF which is relevant to I2NSF work is documented in [I-D.hares-i2nsf-gap-analysis].

The growing challenges and complexity in maintaining a secure infrastructure, complying with regulatory requirements, and controlling costs are enticing enterprises into consuming network security functions hosted by service providers. The hosted security service is especially attractive to small and medium size enterprises who suffer from a lack of security experts to continuously monitor networks, acquire new skills and propose immediate mitigations to ever increasing sets of security attacks.

According to [Gartner-2013], the demand for hosted (or cloud-based) security services is growing. Small and medium-sized businesses (SMBs) are increasingly adopting cloud-based security services to replace on-premises security tools, while larger enterprises are deploying a mix of traditional and cloud-based security services.

To meet the demand, more and more service providers are providing hosted security solutions to deliver cost-effective managed security services to enterprise customers. The hosted security services are primarily targeted at enterprises (especially small/medium ones), but could also be provided to any kind of mass-market customer. As a result, the Network Security Functions (NSFs) are provided and consumed in a large variety of environments. Users of NSFs may consume network security services hosted by one or more providers, which may be their own enterprise, service providers, or a combination of both. This document also briefly describes the following use cases summarized by [I-D.pastor-i2nsf-merged-use-cases]:

- o [I-D.pastor-i2nsf-access-usecases] (I2NSF-Access),
- o [I-D.zarny-i2nsf-data-center-use-cases](I2NSF-DC), and
- o [I-D.qi-i2nsf-access-network-usecase] (I2NSF-Mobile).

2. Terminology

ACL: Access Control List

B2B: Business-to-Business

Bespoke: Something made to fit a particular person, client or company.

Bespoke security management: Security management which is made to fit a particular customer.

DC: Data Center

FW: Firewall

IDS: Intrusion Detection System

IPS: Intrusion Protection System

I2NSF: interface to Network Security Functions.

NSF: Network Security Function. An NSF is a function that detects abnormal activity and blocks/mitigates the effect of such abnormal activity in order to preserve the availability of a network or a service. In addition, the NSF can help in supporting communication stream integrity and confidentiality.

Flow-based NSF: An NSF which inspects network flows according to a security policy. Flow-based security also means that packets are inspected in the order they are received, and without altering packets due to the inspection process (e.g., MAC rewrites, TTL decrement action, or NAT inspection or changes).

Virtual NSF: An NSF which is deployed as a distributed virtual resource.

VNFPool: Pool of Virtual Network Functions.

3. Problem Space

The following sub-section describes the problems and challenges facing customers and security service providers when some or all of the security functions are no longer physically hosted by the customer's administrative domain.

Security service providers can be internal or external to the company. For example, an internal IT Security group within a large

enterprise could act as a security service provider for the enterprise. In contrast, an enterprise could outsource all security services to an external security service provider. In this document, the security service provider function whether it is internal or external, will be denoted as "service provider".

The "Customer-Provider" relationship may be between any two parties. The parties can be in different firms or different domains of the same firm. Contractual agreements may be required in such contexts to formally document the customer's security requirements and the provider's guarantees to fulfill those requirements. Such agreements may detail protection levels, escalation procedures, alarms reporting, etc. There is currently no standard mechanism to capture those requirements.

A service provider may be a customer of another service provider.

3.1. Challenges Facing Security Service Providers

3.1.1. Diverse Types of Security Functions

There are many types of NSFs. NSFs by different vendors can have different features and have different interfaces. NSFs can be deployed in multiple locations in a given network, and perhaps have different roles.

Below are a few examples of security functions and locations or contexts in which they are often deployed:

External Intrusion and Attack Protection: Examples of this function are firewall/ACL authentication, IPS, IDS, and endpoint protection.

Security Functions in a DMZ: Examples of this function are firewall/ACLs, IDS/IPS, authentication and authorization services, NAT, forwarding proxies, application, and AAA services. These functions may be physically on-premise in a server provider's network at the DMZ spots or located in a "virtual" DMZ.

Internal Security Analysis and Reporting: Examples of this function are security logs, event correlation, and forensic analysis.

Internal Data and Content Protection: Examples of this function are encryption, authorization, and public/private key management for internal database.

Given the diversity of security functions, the contexts in which these functions can be deployed, and the constant evolution of these

functions, standardizing all aspects of security functions is challenging, and most probably not feasible. Fortunately, it is not necessary to standardize all aspects. For example, from an I2NSF perspective, there is no need to standardize on how firewall filters are created or applied.

What is needed is a standardized interface to control and monitor the rule sets that NSFs use to treat packets traversing through. And standardizing interfaces will provide an impetus for standardizing established security functions.

3.1.2. Diverse Interfaces to Control NSFs

To provide effective and competitive solutions and services, Security Service Providers may need to utilize multiple security functions from various vendors to enforce the security policies desired by their customers.

Since no widely accepted industry standard security interface exists today, management of NSFs (device and policy provisioning, monitoring, etc.) tends to be bespoke security management offered by product vendors. As a result, automation of such services, if it exists at all, is also bespoke. Thus, even in the traditional way of deploying security features, there is a gap to coordinate among implementations from distinct vendors. This is the main reason why mono-vendor security functions are often deployed and enabled in a particular network segment.

A challenge for monitoring is that an NSF cannot monitor what it cannot view. Therefore, enabling a security function (e.g., firewall [I-D.ietf-opsawg-firewalls]) does not mean that a network is protected. As such, it is necessary to have a mechanism to monitor and provide execution status of NSFs to security and compliance management tools. There exist various network security monitoring vendor-specific interfaces for forensics and troubleshooting.

3.1.3. Diverse Interface to Monitor the Behavior of NSFs

Obviously, enabling a security function (e.g., firewall [I-D.ietf-opsawg-firewalls]) does not mean that a network is protected. Therefore, it is necessary to have a mechanism to monitor the execution status of NSFs.

3.1.4. More Distributed NSFs and vNSFs

The security functions which are invoked to enforce a security policy can be located in different equipment and network locations.

The European Telecommunications Standards Institute (ETSI) Network Function Virtualization (NFV) initiative creates new management challenges for security policies to be enforced by distributed, virtual, and network security functions (vNSF).

A vNSF has higher risk of failure, migrating, and state changes as their hosting VMs are being created, moved, or decommissioned.

3.1.5. More Demand to Control NSFs Dynamically

In the advent of Software-Defined Networking (see [I-D.jeong-i2nsf-sdn-security-services]), more clients, applications or application controllers need to dynamically update their security policies that are enforced by NSFs. The Security Service Providers have to dynamically update their decision-making process (e.g., in terms of NSF resource allocation and invocation) upon receiving requests from their clients.

3.1.6. Demand for Multi-Tenancy to Control and Monitor NSFs

Service providers may need several operational units to control and monitor the NSFs, especially when NSFs become distributed and virtualized.

3.1.7. Lack of Characterization of NSFs and Capability Exchange

To offer effective security services, service providers need to activate various security functions in NSFs or vNSFs manufactured by multiple vendors. Even within one product category (e.g., firewall), security functions provided by different vendors can have different features and capabilities. For example, filters that can be designed and activated by a firewall may or may not support IPv6 depending on the firewall technology.

The service provider's management system (or controller) needs a way to retrieve the capabilities of service functions by different vendors so that it could build an effective security solution. These service function capabilities can be documented in a static manner (e.g., a file) or via an interface which accesses a repository of security function capabilities which the NSF vendors dynamically update.

A dynamic capability registration is useful for automation because security functions may be subject to software and hardware updates. These updates may have implications on the policies enforced by the NSFs.

Today, there is no standard method for vendors to describe the capabilities of their security functions. Without a common technical framework to describe the capabilities of security functions, service providers cannot automate the process of selecting NSF's by different vendors to accommodate customer's requirements.

3.1.8. Lack of Mechanism for NSF's to Utilize External Profiles

Many security functions depend on signature files or profiles to perform (e.g., IPS/IDS signatures, DOTS filters). Different policies might need different signatures or profiles. Today, the construction and use of black databases can be a win-win strategy for all parties involved. There might be Open Source-provided signature/profiles (e.g., by Snort or others) in the future.

There is a need to have a standard envelop (i.e., the format) to allow NSF's to use external profiles.

3.1.9. Lack of Mechanisms to Accept External Alerts to Trigger Automatic Configuration Changes

NSF can ask the I2NSF security controller to alter a specific policy. For example, a DDoS alert could trigger a change to the routing system to send traffic to a traffic scrubbing service to mitigate the DDoS.

The DDoS protection has the following two parts: a) the configuration of signaling of open threats and b) DDoS mitigation. DOTS controller manages the signaling part of DDoS. I2NSF controller(s) would manage the changing to the affected policies (e.g., forwarding and routing, filtering, etc.). By monitoring the network alerts from DDoS, I2NSF can feed an alerts analytics engine that could recognize attacks and the I2NSF can thus enforce the appropriate policies.

DDoS mitigation is enhanced if the provider's network security controller can monitor, analyze, and investigate the abnormal events and provide information to the client or change the network configuration (see section x) for details on the interfaces.

[I-D.zhou-i2nsf-capability-interface-monitoring] provides details on how monitoring aspects of the flow-based Network Security Functions (NSF's) can use the I2NSF interfaces to receive traffic reports and enforce policy.

3.1.10. Lack of Mechanism for Dynamic Key Distribution to NSFs

There is a need for a controller to distribute various keys to distributed NSFs. To distribute various keys, the keys must be created and managed. While there are many key management methods and key derivation functions (KDF), there is a lack of standard interface to provision and manage keys.

The keys may be used for message authentication and integrity in order to protect data flows. In addition, keys may be used to secure the protocol and messages in the core routing infrastructure ([RFC4948])

As of now there is not much focus on an abstraction for keying information that describes the interface between protocols, operators, and automated key management.

The ability to utilize keys when routing protocols send or receive messages will be enhanced by having an abstract key table maintained by a security service. Conceptually, there must be an interface defined for routing/signaling protocols to make requests for automated key management when it is being used, to notify the protocols when keys become available in the key table.

An abstract key service will work under the following conditions:

1. I2NSF needs to design the key table abstraction, the interface between key management protocols and routing/other protocols, and possibly security protocols at other layers.
2. For each routing/other protocol, I2NSF needs to define the mapping between how the protocol represents key material and the protocol-independent key table abstraction. (If protocols share common mechanisms for authentication (e.g., TCP Authentication Option), then the same mapping may be reused.)
3. Automated Key management must support both symmetric keys and group keys via the service provided by items 1 and 2.

3.2. Challenges Facing Customers

When customers invoke hosted security services, their security policies may be enforced by a collection of security functions hosted in different domains. Customers may not have the security skills to express sufficiently precise requirements or security policies. Usually, these customers express the expectations of their security requirements or the intent of their security policies. These expectations can be considered customer level security expectations.

Customers may also desire to express guidelines for security management. Examples of such guidelines include:

- o Which critical communications are to be preserved during critical events (DOTS),
- o Which hosts are to continue service even during severe security attacks (DOTS),
- o Reporting of attacks to CERT (MILE),
- o Managing network connectivity of systems out of compliance (SACM),

3.2.1. NSFs from Heterogeneous Administrative Domains

Many medium and large enterprises have deployed various on-premises security functions which they want to continue to deploy. These enterprises want to combine local security functions with remote hosted security functions to achieve more efficient and immediate counter-measures to both Internet-originated attacks and enterprise network-originated attacks.

Some enterprises may only need the hosted security services for their remote branch offices where minimal security infrastructures/capabilities exist. The security solution will consist of deploying NSFs on customer networks and on service provider networks.

3.2.2. Today's Control Requests are Vendor Specific

Customers may consume NSFs by multiple service providers. Customers need to express their security requirements, guidelines, and expectations to the service providers. In turn, the service providers must translate this customer information into customer security policies and associated configuration tasks for the set of security functions in their network. Without a standard technical characterization or a standard interface, the service provider faces many challenges:

No standard technical characterization and/or APIs : Even for the most common security services there is no standard technical characterization or APIs. Most security services are accessible only through disparate, proprietary interfaces (e.g., portals or APIs) in whatever format vendors choose to offer. The service provider must have the customer's input to manage these widely varying interfaces.

No standard interface: Without standard interfaces it is complex for customers to update security policies or integrate the

extensive preliminary legwork. A standard interface also helps service providers since they could now offer the same firewall/IPS interface to represent firewall/IPS services for utilizing products from many vendors. The result is that the service provider has now abstracted the firewall/IPS services. The standard interface also helps the firewall/IPS vendors to focus on their core security functions or extended features rather than the standard building blocks of a management interface.

3.2.3. Difficulty to Monitor the Execution of Desired Policies

How a policy is translated into technology-specific actions is hidden from the customers. However, customers still need ways to monitor the delivered security service that results from the execution of their desired security requirements, guidelines and expectations.

Today, there is no standard way for customers to get security service assurance of their specified security policies properly enforced by the security functions in the provider domain. The customer also lacks the ability to perform "what-if" scenarios to assess the efficiency of the delivered security service.

3.3. Difficulty to Validate Policies across Multiple Domains

One key aspect of a hosted security service with security functions located at different premises is the ability to express, monitor and verify security policies that combine several distributed security functions. It is crucial to an effective service to be able to take these actions via a standard interface. This standard interface becomes more crucial to the hosted security service when NSFs are instantiated in Virtual Machines which are sometimes widely distributed in the network and sometimes are combined together in one device to perform a set of tasks for delivering a service.

Without standard interfaces and security policy data models, the enforcement of a customer-driven security policy remains challenging because of the inherent complexity created by combining the invocation of several vendor-specific security functions into a multi-vendor, heterogeneous environment. Each vendor-specific function may require specific configuration procedures and operational tasks.

Ensuring the consistent enforcement of the policies at various domains is also challenging. Standard data models are likely to contribute to addressing that issue.

3.4. Lack of Standard Interface to Inject Feedback to NSF

Today, many security functions, such as IPS, IDS, DDoS and Antivirus, depend heavily on the associated profiles. They can perform more effective protection if they have the up-to-date profiles. As more sophisticated threats arise, enterprises, vendors, and service providers have to rely on each other to achieve optimal protection. Cyper Threat Alliance (CA, <http://cyberthreatalliance.org/>) is one of those initiatives that aim at combining efforts conducted by multiple organizations.

Today there is no standard interface to exchange security profiles between organizations.

3.5. Lack of Standard Interface for Capability Negotiation

There could be situations when the selected NSFs cannot perform the policies requested by the Security Controller, due to resource constraints. The customer and security service provider should negotiate the appropriate resource constraints before the security service begins. However, unexpected events somethings happen and the NSF may exhaust those negotiated resources. At this point, the NSF should inform the security controller that the allotted resources have been exhausted. To support the automatic control in the SDN-era, it is necessary to have a set of messages for proper notification (and a response to that notification) between the Security Controller and the NSFs.

4. Use Cases

Standard interfaces for monitoring and controlling the behavior of NSFs are essential building blocks for Security Service Providers and enterprises to automate the use of different NSFs from multiple vendors by their security management entities. I2NSF may be invoked by any (authorized) client. Examples of authorized clients are upstream applications (controllers), orchestration systems, and security portals.

4.1. Basic Framework

Users request security services through specific clients (e.g., a customer application, the NSP BSS/OSS or management platform) and the appropriate NSP network entity will invoke the (v)NSFs according to the user service request. This network entity is denoted as the security controller in this document. The interaction between the entities discussed above (client, security controller, NSF) is shown in Figure 2:

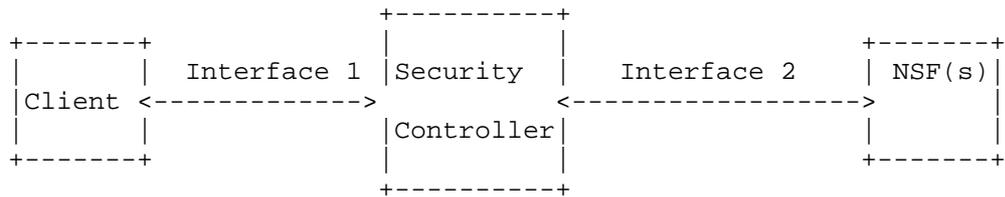


Figure 2: Interaction between Entities

Interface 1 is used for receiving security requirements from client and translating them into commands that NSFs can understand and execute. The security controller also passes back NSF security reports (e.g., statistics) to the client which the control has gathered from NSFs. Interface 2 is used for interacting with NSFs according to commands, and collect status information about NSFs.

Client devices or applications can require the security controller to add, delete or update rules in the security service function for their specific traffic.

When users want to get the executing status of a security service, they can request NSF status from the client. The security controller will collect NSF information through Interface 2, consolidate them, and give feedback to client through Interface 1. This interface can be used to collect not only individual service information, but also aggregated data suitable for tasks like infrastructure security assessment.

Customers may require validating NSF availability, provenance, and correct execution. This validation process, especially relevant for vNSFs, includes at least:

Integrity of the NSF: by ensuring that the NSF is not compromised;

Isolation: by ensuring the execution of the NSF is self-contained for privacy requirements in multi-tenancy scenarios.

Provenance of NSF: Customers may need to be provided with strict guarantees about the origin of the NSF, its status (e.g. available, idle, down, and others), and feedback mechanisms so that a customer may be able to check that a given NSF or set of NSFs properly conform to the the customer's requirements and subsequent configuration tasks.

In order to achieve this, the security controller may collect security measurements and share them with an independent and trusted

third party (via interface 1) in order to allow for attestation of NSF functions using the third party added information.

4.2. Access Networks

This scenario describes use cases for users (e.g. enterprise user, network administrator, and residential user) that request and manage security services hosted in the network service provider (NSP) infrastructure. Given that NSP customers are essentially users of their access networks, the scenario is essentially associated with their characteristics, as well as with the use of vNSFs.

The Virtual CPE described in [NFVUC] use cases #5 and #7 requires a model of access virtualization that includes mobile and residential access where the operator may offload security services from the customer local environment (e.g., device or terminal) to its own infrastructure.

These use cases define the interaction between the operator and the vNSFs through automated interfaces, typically by means of B2B communications.

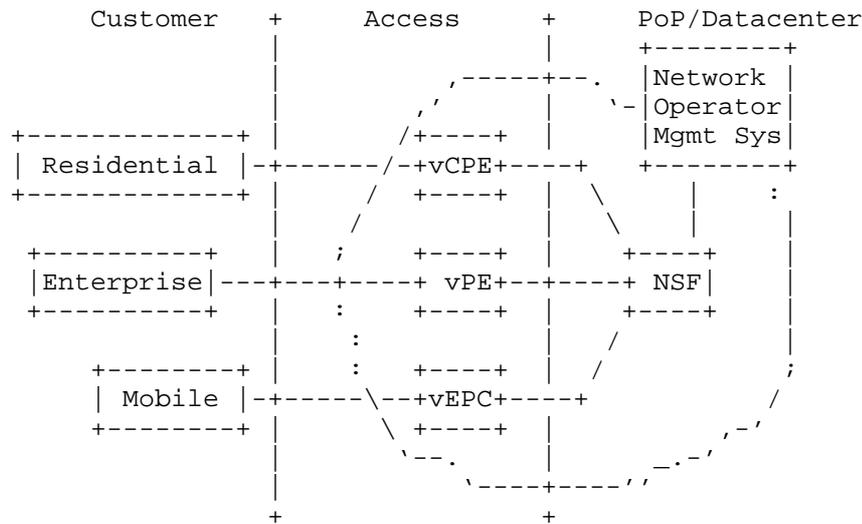


Figure 3: NSF and actors

Different access clients may have different service requests:

Residential: Interface for parental control; Interface for threat management.

Enterprise: Flow Security Policies; Managed security services (Threat management).

Mobile: User experience; content and access management; Threat management for infrastructure, such as Botnet, DDoS, Malware etc.

Some access customers may not care about which NSF's are utilized to achieve the services they requested. In this case, provider network orchestration systems can internally select the NSF's (or vNSF's) to enforce the policies requested by the clients. Other access customers, especially some enterprise customers, may want to get their dedicated NSF's (most likely vNSF's) for direct control purposes. In this case, here are the steps to associate vNSF's to specific customers:

vNSF Deployment: The deployment process consists in instantiating a NSF on a Virtualization Infrastructure (NFVI), within the NSP administrative domain(s) or with other external domain(s). This is a required step before a customer can subscribe to a security service supported in the vNSF.

vNSF Customer Provisioning: Once a vNSF is deployed, any customer can subscribe to it. The provisioning lifecycle includes the following:

- * Customer enrollment and cancellation of the subscription to a vNSF;
- * Configuration of the vNSF, based on specific configurations, or derived from common security policies defined by the NSP.
- * Retrieve and list the vNSF functionalities, extracted from a manifest or a descriptor. The NSP management systems can demand this information to offer detailed information through the commercial channels to the customer.

4.3. Cloud Data Center Scenario

In a data center, network security mechanisms such as firewalls may need to be dynamically added or removed for a number of reasons. These changes may be explicitly requested by the user, or triggered by a pre-agreed upon Service Level Agreement (SLA) between the user and the provider of the service. For example, the service provider may be required to add more firewall capacity within a set timeframe whenever the bandwidth utilization hits a certain threshold for a

specified period. This capacity expansion could result in adding new instances of firewalls on existing machines or provisioning a completely new firewall instance in a different machine.

The on-demand, dynamic nature of security service delivery essentially encourages that the network security "devices" be in software or virtual form factors, rather than in a physical appliance form. This requirement is a provider-side concern. Users of the firewall service are agnostic (as they should) as to whether or not the firewall service is run on a VM or any other form factor. Indeed, they may not even be aware that their traffic traverses firewalls.

Furthermore, new firewall instances need to be placed in the "right zone" (domain). The issue applies not only to multi-tenant environments where getting the tenant in the right domain is of paramount importance, but also in environments owned and operated by a single organization with its own service segregation policies. For example, an enterprise may mandate that firewalls serving Internet traffic and Business-to-Business (B2B) traffic be separated. Another example is that IPS/IDS services for investment banking and non-banking traffic may be separated for regulatory reasons.

4.3.1. On-Demand Virtual Firewall Deployment

A service provider-operated cloud data center could serve tens of thousands of clients. Clients' compute servers are typically hosted on virtual machines (VMs), which could be deployed across different server racks located in different parts of the data center. It is often not technically and/or financially feasible to deploy dedicated physical firewalls to suit each client's security policy requirements, which can be numerous. What is needed is the ability to dynamically deploy virtual firewalls for each client's set of servers based on established security policies and underlying network topologies.

allow clients to retrieve/manage security policies in a consistent manner across different providers.

4.3.4. Internal Network Monitoring

There are many types of internal traffic monitors that may be managed by a security controller. This includes a new class of services referred to as Data Loss Prevention (DLP), or Reputation Protection Services (RPS). Depending on the class of event, alerts may go to internal administrators, or external services.

4.4. I2NSF Preventing Distributed DoS in Overlay Networks

In the internet where everything is connected, preventing unwanted traffic that may cause Denial of Service (DoS) attack or distributed DoS (DDoS) has become a challenge. One place where DDoS can be challenging to prevent or mitigate is in overlay networks. Many networks such as Internet of Things (IoT) networks, Information-Centric Networks (ICN), Content Delivery Networks (CDN), and cloud networks use overlay networks within their paths (or links). The underlay networks that support overlay networks can be attacked by DDoS, thereby saturating access links or links within the network. DoS or DDoS attacks on the access links may also cause the overlay nodes' CPUs or links to be saturated by DoS or DDoS traffic which will prevent these links from being used by legitimate overlay traffic. Overlay security solutions do not address underlay security threats so there is a need for a distributed solution to prevent DDoS attacks from spreading throughout overlay and underlay networks. Such solution may for example rely upon the dynamic, highly-reactive, enforcement of security filtering policies network-wise.

Similar to traditional networks placing a firewall or Intrusion Prevention System (IPS) on the wire to enforce traffic rules, the interface to network security functions (I2NSF) can be used by overlay networks to request underlay networks enforce certain flow-based security rules. Using this mechanism, the overlay network can coordinate with the underlay network to remove unwanted traffic including DoS and DDoS in the underlay network.

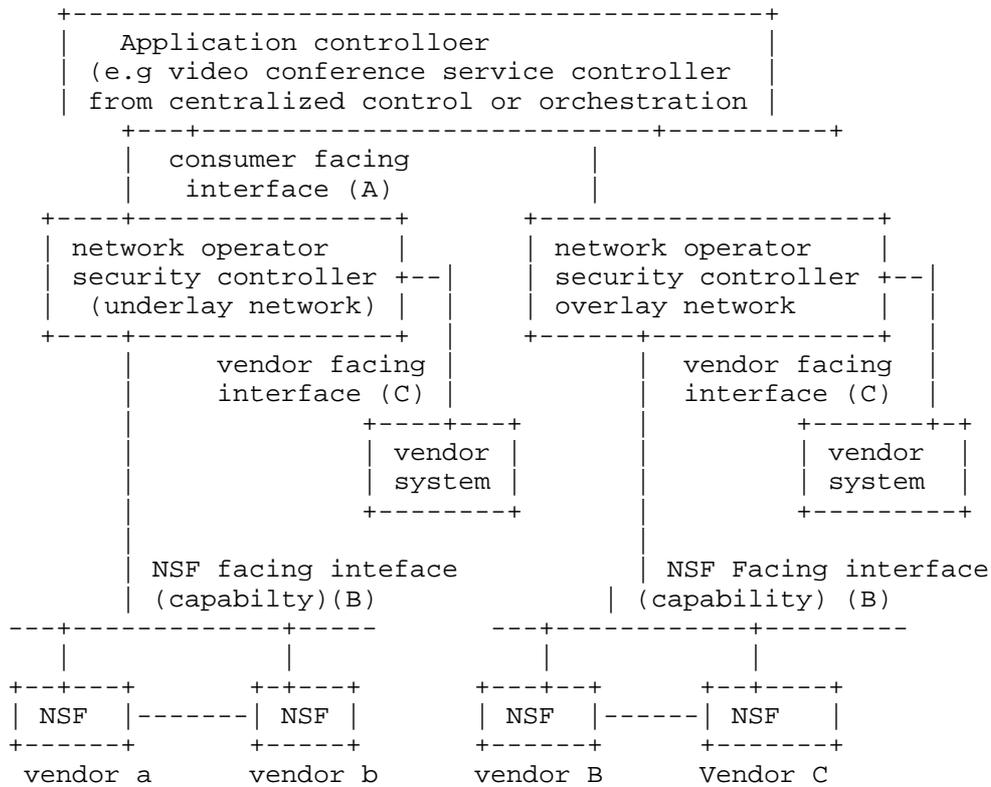


Figure 5: I2NSF Preventing DDoS Attacks in Overlay Networks.

5. Management Considerations

Management of NSFs usually include the following:

- o Lifecycle management and resource management of NSFs,
- o Device configuration, such as address configuration, device internal attributes configuration, etc.;
- o Signaling, and
- o Policy rule provisioning.

I2NSF will only focus on the policy provisioning part of NSF management.

6. IANA Considerations

No IANA considerations exist for this document.

7. Security Considerations

Having a secure access to control and monitor NSFs is crucial for hosted security services. An I2NSF security controller raises new security threats. It needs to be resilient to attacks and quickly recover from attacks. Therefore, proper secure communication channels have to be carefully specified for carrying controlling and monitoring traffic between the NSFs and their management entity (or entities).

In addition, the Flow security policies specified by customers can conflict with providers' internal security policies which may allow unauthorized traffic or unauthorized changes to flow polices (e.g. customers changing flow policies that do not belong to them). Therefore, it is crucial to have proper AAA to authorize access to the network and access to the I2NSF management stream.

8. Contributors

I2NSF is a group effort. The following people actively contributed to the initial use case text: Xiaojun Zhuang (China Mobile), Sumandra Majee (F5), Ed Lopez (Fortinet), and Robert Moskowitz (Huawei).

9. Contributing Authors

I2NSF has had a number of contributing authors. The following are contributing authors:

- o Antonio Pastur (Telefonica I+D),
- o Mohamed Boucadair (France Telecom),
- o Michael Georgiades (Prime Tel),
- o Minpeng Qi (China Mobile),
- o Shaibal Chakrabarty (US Ignite), and
- o Nic Leymann (Deutsche Telekom).

10. References

10.1. 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>>.

10.2. Informative References

- [Gartner-2013] Messmer, E., "Gartner: Cloud-based security as a service set to take off", October 2013.
- [I-D.hares-i2nsf-gap-analysis] Hares, S., Zhang, D., Moskowitz, R., and H. Rafiee, "Analysis of Existing work for I2NSF", draft-hares-i2nsf-gap-analysis-01 (work in progress), December 2015.
- [I-D.ietf-netmod-acl-model] Bogdanovic, D., Koushik, K., Huang, L., and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-06 (work in progress), December 2015.
- [I-D.ietf-opsawg-firewalls] Baker, F. and P. Hoffman, "On Firewalls in Internet Security", draft-ietf-opsawg-firewalls-01 (work in progress), October 2012.
- [I-D.jeong-i2nsf-sdn-security-services] Jeong, J., Kim, H., and P. Jung-Soo, "Requirements for Security Services based on Software-Defined Networking", draft-jeong-i2nsf-sdn-security-services-01 (work in progress), March 2015.
- [I-D.lopez-i2nsf-packet] Ed, E., "Packet-Based Paradigm For Interfaces To NSFs", draft-lopez-i2nsf-packet-00 (work in progress), March 2015.
- [I-D.pastor-i2nsf-access-usecases] Pastor, A. and D. Lopez, "Access Use Cases for an Open OAM Interface to Virtualized Security Services", draft-pastor-i2nsf-access-usecases-00 (work in progress), October 2014.

- [I-D.pastor-i2nsf-merged-use-cases]
Pastor, A., Lopez, D., Wang, K., Zhuang, X., Qi, M., Zarny, M., Majee, S., Leymann, N., Dunbar, L., and M. Georgiades, "Use Cases and Requirements for an Interface to Network Security Functions", draft-pastor-i2nsf-merged-use-cases-00 (work in progress), June 2015.
- [I-D.qi-i2nsf-access-network-usecase]
Wang, K. and X. Zhuang, "Integrated Security with Access Network Use Case", draft-qi-i2nsf-access-network-usecase-02 (work in progress), March 2015.
- [I-D.zarny-i2nsf-data-center-use-cases]
Zarny, M., Leymann, N., and L. Dunbar, "I2NSF Data Center Use Cases", draft-zarny-i2nsf-data-center-use-cases-00 (work in progress), October 2014.
- [I-D.zhou-i2nsf-capability-interface-monitoring]
Zhou, C., Xia, L., Boucadair, M., and J. Xiong, "The Capability Interface for Monitoring Network Security Functions (NSF) in I2NSF", draft-zhou-i2nsf-capability-interface-monitoring-00 (work in progress), October 2015.
- [RFC4948] Andersson, L., Davies, E., and L. Zhang, "Report from the IAB workshop on Unwanted Traffic March 9-10, 2006", RFC 4948, DOI 10.17487/RFC4948, August 2007, <<http://www.rfc-editor.org/info/rfc4948>>.
- [RFC7277] Bjorklund, M., "A YANG Data Model for IP Management", RFC 7277, DOI 10.17487/RFC7277, June 2014, <<http://www.rfc-editor.org/info/rfc7277>>.

Authors' Addresses

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

Phone: +1-734-604-0332
Email: shares@ndzh.com

Linda Dunbar
Huawei
5340 Legacy Drive, Suite 175
Plano, TX 75024
USA

Phone: +1-734-604-0332
Email: ldunbar@huawei.com

Diego R. Lopex
Telefonica I+D
Don Ramon de la Cruz, 82
Madrid 28006
Spain

Email: diego.r.lopez@telefonica.com

Myo Zarny
Goldman Sachs
30 Hudson Street
Jersey City, NJ 07302
USA

Email: myo.zarny@gs.com

Christian Jacquenet
France Telecom
Rennes, 35000
France

Email: Christian.jacquenet@orange.com

I2NSF
Internet-Draft
Intended status: Informational
Expires: January 29, 2017

S. Hares
J. Strassner
Huawei
D. Lopez
Telefonica I+D
L. Xia
Huawei
July 8, 2016

Interface to Network Security Functions (I2NSF) Terminology
draft-ietf-i2nsf-terminology-01.txt

Abstract

This document defines a set of terms that are used for the Interface to Network Security Functions (I2NSF) effort.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 29, 2017.

Copyright Notice

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

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

Table of Contents

1. Introduction 2
 2. Terminology 3
 3. IANA Considerations 10
 4. Security Considerations 10
 5. Contributors 11
 6. References 11
 6.1. Informative References 11
 Authors' Addresses 12

1. Introduction

This document defines the terminology for the Interface to Network Security Functions (I2NSF) effort. This section provides some background on I2NSF; a detailed problem statement can be found in [I-D.ietf-i2nsf-problem-and-use-cases]. Motivation and comparison to previous work can be found in [I-D.ietf-i2nsf-gap-analysis].

Enterprises are now considering using network security functions (NSFs) hosted by service providers due to the growing challenges and complexity in maintaining an up-to-date secure infrastructure that complies with regulatory requirements, while controlling costs. The hosted security service is especially attractive to small- and medium-size enterprises who suffer from a lack of security experts to continuously monitor, acquire new skills and propose immediate mitigations to ever increasing sets of security attacks. Small- and medium-sized businesses (SMBs) are increasingly adopting cloud-based security services to replace on-premises security tools, while larger enterprises are deploying a mix of traditional (hosted) and cloud-based security services.

To meet the demand, more and more service providers are providing hosted security solutions to deliver cost-effective managed security services to enterprise customers. The hosted security services are primarily targeted at enterprises, but could also be provided to mass-market customers as well. NSFs are provided and consumed in increasingly diverse environments. Users of NSFs may consume network security services hosted by one or more providers, which may be their own enterprise, service providers, or a combination of both.

It is out of scope in this document to define an exhaustive list of terms that are used in the security field; the reader is referred to other applicable documents, such as [RFC4949].

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 [RFC2119] significance.

3. Terminology

AAA: Authentication, Authorization, and Accounting. See individual definitions.

Abstraction: The definition of the salient characteristics and behavior of an object that distinguish it from all other types of objects. It manages complexity by exposing common properties between objects and processes while hiding detail that is not relevant.

Access Control: Protection of system resources against unauthorized access; a process by which use of system resources is regulated according to a security policy, and is permitted by only authorized entities (users, programs, processes, or other systems) according to that policy [RFC4949].

Accounting: The act of collecting information on resource usage for the purpose of trend analysis, auditing, billing, or cost allocation ([RFC2975] [RFC3539]).

ACL (Access Control List): This is a mechanism that implements access control for a system resource by enumerating the system entities that are permitted to access the resource and stating, either implicitly or explicitly, the access modes granted to each entity [RFC4949]. A YANG description is defined in [I-D.ietf-netmod-acl-model].

Action: Defines what is to be done when a set of Conditions are met (See I2NSF Action). (from [I-D.ietf-supra-generic-policy-info-model]).

Assertion: Defined by the ITU in [X.1252] as "a statement made by an entity without accompanying evidence of its validity". In the context of I2NSF, an assertion MAY include metadata about all or part of the assertion (e.g., context of the assertion, or about timestamp indicating the point in time the assertion was created). The validity of an assertion cannot be verified. (from [I-D.ietf-sacm-terminology]).

Authentication: Defined in [RFC4949] as "the process of verifying a claim that a system entity or system resource has a certain attribute value." (from [I-D.ietf-sacm-terminology]).

Authorization: Defined in [RFC4949] as "an approval that is granted to a system entity to access a system resource." (from [I-D.ietf-sacm-terminology]).

B2B: Business-to-Business.

Bespoke: Something made to fit a particular person, customer, or company.

Bespoke security management: Security management systems that are make to fit a particular customer.

Boolean Clause: A logical statement that evaluates to either TRUE or FALSE. Also called Boolean Expression.

Capability: Defines a set of features that are available from a managed entity (see also I2NSF Capability). Examples of "managed entities" are NSFs and Controllers, where NSF Capabilities and Controller Capabilities define functionality of an NSF and about Controller, respectively. These functions may, but do not have to, be used. All Capabilities are announced through the Registration Interface.

Capability Interface: An interface dedicated to requesting, receiving, editing, and deleting capability information.

Client: See Consumer. [Editor's note: placeholder for gradually replacing Client with Consumer, since Client is too vague and has other connotations (e.g., client-server)].

Client-Facing Interface: See Consumer-Facing Interface.
See also: Interface, NSF-Facing Interface.

Component: An encapsulation of software that communicates using Interfaces. A Component may be implemented by hardware and/or software, and be represented using a set of classes. In general, a Component encapsulates a set of data structures and a set of algorithms that implement the function(s) that it provides.

Consumer: A Consumer is a Role that is assigned to an I2NSF Component that can receive information from another I2NSF Component. See also: Provider, Role.

Consumer-Facing Interface: An Interface dedicated to communication with Consumers of NSF data and Services. This is typically defined per I2NSF administrative domain. See also: Interface, NSF-Facing Interface.

Condition: A set of attributes, features, and/or values that are to be compared with a set of known attributes, features, and/or values in order to make a decision. A Condition, when used in the context of a Policy Rule, is used to determine whether or not the set of Actions in that Policy Rule can be executed or not. Examples of an I2NSF Condition include matching attributes of a packet or flow, and comparing the internal state of a NSF to a desired state. (from [I-D.ietf-supra-generic-policy-info-model]).

Constraint: A Constraint is a limitation or restriction. Constraints may be associated with any type of object (e.g., Events, Conditions, and Actions in Policy Rules).

Constraint Programming: A type of programming that uses constraints to define relations between variables in order to find a feasible (and not necessarily optimal) solution.

Context: The Context of an Entity is a collection of measured and/or inferred knowledge that describe the state and the environment in which an Entity exists or has existed. (from <http://www.ietf.org/mail-archive/web/i2nsf/current/msg00762.html>).

Controller: A Controller is a management Component that contains control plane functions to manage and facilitate information sharing, as well as execute security functions. This definition is based on that in [I-D.ietf-sacm-terminology].

Control Plane: In the context of I2NSF, the Control Plane is an architectural Component that provides common control functions to all I2NSF Components, including some or all of the following: authentication, authorization, accounting, auditing, and Capability discovery and negotiation. The Control Plane orchestrates the operation of the Data Plane according to guidance and/or input from the Management Plane. I2NSF Components with Interfaces to the Control Plane have knowledge of the Capabilities of other I2NSF Components within a particular I2NSF administrative domain. This definition is based on that in [I-D.ietf-sacm-terminology]. See also: Data Plane, Management Plane.

Customer: A business role of an entity that is involved in the definition and/or consumption of services, and the possible negotiation of a contract to use services from a Provider.

DC: Data Center

Data Model: A representation of concepts of interest to an environment in a form that is dependent on data repository, data definition language, query language, implementation language, and protocol (typically one or more of these). Note the difference between a data ****model**** and a data ****structure****.
[I-D.ietf-supra-generic-policy-info-model].

Data Plane: In the context of I2NSF, the Data Plane is an architectural Component that provides operational functions to enable an I2NSF Component to provide and consume packets and flows. See also: Control Plane, Management Plane.

Data Structure: A low-level building block that is used in programming to implement an algorithm. A data model typically contains multiple types of data structures; however, a data structure does not contain a data model. Note the difference between a data ****model**** and a data ****structure****.

Event: An important occurrence in time of a change in the system being managed, and/or in the environment of the system being managed. Examples of an I2NSF Event include time and user actions (e.g. logon, logoff, and actions that violate an ACL). An Event, when used in the context of a Policy Rule, is used to determine whether the Condition clause of an imperative Policy Rule can be evaluated or not (from [I-D.ietf-supra-generic-policy-info-model]).

ECA: Event - Condition - Action (a type of Policy Rule).

Firewall (FW): A function that restricts data communication traffic to and from one of the connected networks (the one said to be 'inside' the firewall), and thus protects that network's system resources against threats from the other network (the one that is said to be 'outside' the firewall) [RFC4949].
[I-D.ietf-opsawg-firewalls]

Flow-based NSF: A NSF that inspects network flows according to a set of policies intended for enforcing security properties. Flow-based security also means that packets are inspected in the order they are received, and without modification to the packet due to the inspection process.

I2NSF Action: An I2NSF Action is a special type of Action that is used to control and monitor aspects of flow-based Network Security Functions. Examples of I2NSF Actions include providing intrusion detection and/or protection, web and flow filtering, and deep packet inspection for packets and flows. An I2NSF Action, when used in the context of a I2NSF Policy Rule, may be executed when both the Event and the Condition clauses of its owning I2NSF Policy Rule evaluate to true. The execution of this Action may be influenced by applicable metadata. (from [I-D.ietf-supra-generic-policy-info-model]).

I2NSF Agent: A software Component in a device that implements an NSF. It receives provisioning information and requests for operational data (e.g., monitoring data) from an I2NSF Consumer. It is also responsible for enforcing the policies that it receives from an I2NSF Consumer.

I2NSF Capability: A set of features that are available from an NSF Server or an NSF Controller. While both are Capabilities, the former defines functions that are available from an NSF, whereas the latter defines functions that are available from a security Controller or other Management Entity. This definition is based on that in [I-D.ietf-sacm-terminology].

I2NSF Client: See I2NSF Consumer.

I2NSF Component: A Component that provides one or more I2NSF Services. I2NSF Components are managed and communicate with other I2NSF Components using I2NSF Interfaces.

I2NSF Consumer: A software Component that uses the I2NSF framework to read, write, and/or change provisioning and operational aspects of the NSFs that it attaches to.

I2NSF Consumer Interface: An Interface dedicated to requesting and using I2NSF Services. For example, this Interface could be used to request a set of Flow Security policies from an I2NSF Controller or from one or more individual NSFs. The difference is that the former uses more abstract Condition matching (e.g., based on tenant or customer ID), whereas the latter uses more low-level Condition matching (e.g., based on flow state or fields in a flow or packet). See also: Interface, I2NSF Provider Interface, Client-Facing Interface, NSF-Facing Interface.

I2NSF Management System: I2NSF Consumers operate within the scope of a network management system, which serves as a collection and distribution point for I2NSF security provisioning.

I2NSF Policy: A set of Policy Rules that are used to manage and control the changing or maintaining of the state of an instance of an NSF.

I2NSF Policy Rule: A Policy Rule that is adapted for I2NSF usage. The I2NSF Policy Rule is assumed to be in ECA form (i.e., an imperative structure). Other types of programming paradigms (e.g., declarative and functional) are currently out of scope. An example of an I2NSF Policy Rule is, in pseudo-code:

```
IF <event-clause> is TRUE
  IF <condition-clause> is TRUE
    THEN execute <action-clause>
  END-IF
END-IF
```

In the above example, the Event, Condition, and Action portions of a Policy Rule are all ****Boolean Clauses****.

I2NSF Provider Interface: An Interface dedicated to providing I2NSF Services. For example, this could provide Anti-Virus, (D)DoS, or IPS Services. See also: Interface, I2NSF Provider Interface, Client-Facing Interface, NSF-Facing Interface.

I2NSF Registry: A registry that contains I2NSF capability information, which can be controlled by the I2NSF Management System. See also: Registry.

I2NSF Service: A set of functions, provided by an I2NSF Consumer, which are used by zero or more I2NSF Producers. Exemplary I2NSF Services include Anti-Virus, Authentication, Authorization, (D)DoS, Firewall, and IPS Services. See also: Interface, I2NSF Provider Interface, Client-Facing Interface, NSF-Facing Interface.

IDS: Intrusion Detection System (see below).

IPS: Intrusion Protection System (see below).

Information Model: A representation of concepts of interest to an environment in a form that is independent of data repository, data definition language, query language, implementation language, and protocol [I-D.ietf-supra-generic-policy-info-model].

Interface: A set of operations one object knows it can invoke on, and expose to, another object. It is a subset of all operations that a given object implements. The same object may have multiple types of interfaces to serve different purposes. An example of multiple interfaces can be seen by considering the interfaces include a firewall uses; these include:

- * multiple interfaces for data packets to traverse through,
- * an interface for a controller to impose policy, or retrieve the results of execution of a policy rule.

See also: Consumer Interface, I2NSF Interface, Provider Interface

Intrusion Detection System (IDS): A system that detects network intrusions via a variety of filters, monitors, and/or probes. An IDS may be stateful or stateless.

Intrusion Protection System (IPS): A system that protects against network intrusions. An IPS may be stateful or stateless.

Management Plane: In the context of I2NSF, the Management Plane is an architectural Component that provides common functions to define the behavior of I2NSF Components. The primary use of the Management Plane is to transport behavioral commands, and supply OAM data, for making decisions that affect behavior. Examples include modifying the configuration of an I2NSF Component and transporting OAM data. See also: Control Plane, Data Plane.

Metadata: Data that provides information about other data. Examples include IETF network management protocols (e.g. NETCONF, RESTCONF, IPFIX) or IETF routing interfaces (I2RS). The I2NSF security interface may utilize Metadata to describe and/or prescribe characteristics and behavior of the YANG data models.

Middlebox: Any intermediary device performing functions other than the normal, standard functions of an IP router on the datagram path between a source host and destination host [RFC3234].

Network Security Function (NSF): Software that provides a set of security-related services. Examples include detecting unwanted activity and blocking or mitigating the effect of such unwanted activity in order to fulfil service requirements. The NSF can also help in supporting communication stream integrity and confidentiality.

NSF-Facing Interface: An Interface dedicated to communication with a set of NSFs. This is typically defined per I2NSF administrative domain. See also: Interface, Consumer-Facing Interface.

OAM: Operation, Administrative, and Management.

OCL (Object Constraint Language): A constraint programming language that is used to specify constraints (e.g., in UML) (from <http://www.ietf.org/mail-archive/web/i2nsf/current/msg00762.html>)

Policy Rule: A set of rules that are used to manage and control the changing or maintaining of the state of one or more managed objects. Often this is shortened to Rule or Policy (see I2NSF policy rule). (from [I-D.ietf-supra-generic-policy-info-model]).

Profile: A structured representation of information that uses a pre-defined set of capabilities of an object, typically in a specific context. Zero or more Capabilities may be changed at runtime. This may be used to simplify how this object interacts with other objects in its environment.

Producer: A Producer is a Role that is assigned to an I2NSF Component that can send information and/or commands to another I2NSF Component. See also: Consumer, Role.

Registry: A logically centralized location containing data of a particular type; it may optionally contain metadata, relationships, and other aspects of the registered data in order to use those data effectively. An I2NSF registry is used to contain capability information that can be controlled by the controller.

Registration Interface: An interface dedicated to requesting, receiving, editing, and deleting information in a Registry.

Role: An abstraction of a Component that models context-specific views and responsibilities of an object as separate Role objects. Role objects can optionally be attached to, and removed from, the object that the Role object describes at runtime. This provides three important benefits. First, it enables different behavior to be supported by the same Component for different contexts. Second, it enables the behavior of a Component to be adjusted dynamically (i.e., at runtime, in response to changes in context) by using one or more Roles to define the behavior desired for each context. Third, it decouples the Roles of a Component from the Applications use that Component.

Service Interface: An Interface dedicated to enabling Policy Rules to be managed. This is also called the I2NSF Consumer Interface.

Service Provider Security Controller: TBD (Editorial: Place holder for a split between controller and security controller definitions.)

Tenant: A group of users that share common access privileges to the same software. An I2NSF tenant may be physical or virtual, and may run on a variety of systems or servers.

Vendor-Facing Interface: An Interface dedicated to registering and vendor-specific NSFs and Capabilities. It is also used to invoke vendor-specific functionality. This is also called the NSF-Facing Interface.

3. IANA Considerations

No IANA considerations exist for this document.

4. Security Considerations

This is a terminology document with no security considerations.

5. Contributors

The following people contributed to creating this document, and are listed in alphabetical order:

Henk Birkholz

6. References

6.1. Informative References

[I-D.ietf-i2nsf-gap-analysis]

Hares, S., Moskowitz, R., and D. Zhang, "Analysis of Existing work for I2NSF", draft-ietf-i2nsf-gap-analysis-01 (work in progress), April 2016.

[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-01 (work in progress), July 2016.

[I-D.ietf-netmod-acl-model]

Bogdanovic, D., Sreenivasa, K., Huang, L., Blair, D., "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-08 (work in progress), July 2016.

[I-D.ietf-opsawg-firewalls]

Baker, F. and P. Hoffman, "On Firewalls in Internet Security", draft-ietf-opsawg-firewalls-01 (work in progress), October 2012.

[I-D.ietf-sacm-terminology]

Birkholz, H., Lu, J., Cam-Wignet, N., "Secure Automation and Continuous Monitoring (SACM) Terminology", draft-ietf-sacm-terminology-09, March 2016

[I-D.ietf-supra-generic-policy-info-model]

Strassner, J., Halpern, J., and J. Coleman, "Generic Policy Information Model for Simplified Use of Policy Abstractions (SUPA)", draft-ietf-supra-generic-policy-info-model-00 (work in progress), June 2016.

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

[RFC2975] Aboba, B., Arkko, J., and D. Harrington, "Introduction to Accounting Management", RFC 2975, DOI 10.17487/RFC2975, October 2000, <<http://www.rfc-editor.org/info/rfc2975>>.

- [RFC3234] Carpenter, B. and S. Brim, "Middleboxes: Taxonomy and Issues", RFC 3234, DOI 10.17487/RFC3234, February 2002, <<http://www.rfc-editor.org/info/rfc3234>>.
- [RFC3539] Aboba, B. and J. Wood, "Authentication, Authorization and Accounting (AAA) Transport Profile", RFC 3539, DOI 10.17487/RFC3539, June 2003, <<http://www.rfc-editor.org/info/rfc3539>>.
- [RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <<http://www.rfc-editor.org/info/rfc4949>>.
- [X.1252] ITU-T, "Baseline identity management terms and definitions", Recommendation ITU-T X.1252, April 2010

Authors' Addresses

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA
Phone: +1-734-604-0332
Email: shares@ndzh.com

John Strassner
Huawei Technologies
Santa Clara, CA
USA
Email: john.sc.strassner@huawei.com

Diego R. Lopez
Telefonica I+D
Don Ramon de la Cruz, 82
Madrid 28006
Spain
Email: diego.r.lopez@telefonica.com

Liang Xia (Frank)
Huawei
101 Software Avenue, Yuhuatai District
Nanjing , Jiangsu 210012
China
Email: Frank.Xialiang@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 6, 2017

J. Jeong
H. Kim
Sungkyunkwan University
J. Park
ETRI
T. Ahn
S. Lee
Korea Telecom
July 5, 2016

Software-Defined Networking Based Security Services using Interface to
Network Security Functions
draft-jeong-i2nsf-sdn-security-services-05

Abstract

This document describes a framework, objectives, requirements, and use cases for security services based on Software-Defined Networking (SDN) using a common Interface to Network Security Functions (I2NSF). It first proposes the framework of SDN-based security services in the I2NSF framework. It then explains three use cases, such as a centralized firewall system, centralized DDoS-attack mitigation system, and centralized VoIP/VoLTE security system.

Status of This Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-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 January 6, 2017.

Copyright Notice

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

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

Table of Contents

1. Introduction	3
2. Requirements Language	4
3. Terminology	4
4. Overview	5
5. Objectives	7
6. Requirements	8
7. Use Cases	9
7.1. Centralized Firewall System	9
7.2. Centralized DDoS-attack Mitigation System	10
7.3. Centralized VoIP/VoLTE Security System	12
8. Security Considerations	14
9. Acknowledgements	14
10. References	14
10.1. Normative References	14
10.2. Informative References	15
Appendix A. Changes from draft-jeong-i2nsf-sdn-security-services-04	16

1. Introduction

Software-Defined Networking (SDN) is a set of techniques that enables users to directly program, orchestrate, control and manage network resources through software (e.g., SDN applications). It relocates the control of network resources to a dedicated network element, namely SDN controller. The SDN controller uses interfaces to arbitrate the control of network resources in a logically centralized manner. It also manages and configures the distributed network resources, and provides the abstracted view of the network resources to the SDN applications. The SDN applications can customize and automate the operations (including management) of the abstracted network resources in a programmable manner via the interfaces [RFC7149][ITU-T.Y.3300][ONF-OpenFlow][ONF-SDN-Architecture].

Due to the increase of sophisticated network attacks, the legacy security services become difficult to cope with such network attacks in an autonomous manner. SDN has been introduced to make networks more controllable and manageable, and this SDN technology will be promising to autonomously deal with such network attacks in a prompt manner.

This document describes a framework, objectives and requirements to support the protection of network resources through SDN-based security services using a common interface to Network Security Functions (NSF) [i2nsf-framework]. It uses an interface to NSF (I2NSF) for such SDN-based security services that are performed in virtual machines through network functions virtualization [ETSI-NFV].

This document addresses the challenges of the existing systems for security services. As feasible solutions to handle these challenges, this document proposes three use cases of the security services, such as a centralized firewall system, centralized DDoS-attack mitigation system, and centralized VoIP/VoLTE security system.

For the centralized firewall system, this document raises limitations in the legacy firewalls in terms of flexibility and administration costs. Since in many cases, access control management for firewall is manually performed, it is difficult to add the access control policy rules corresponding to new network attacks in a prompt and autonomous manner. Thus, this situation requires expensive administration costs. This document introduces a use case of SDN-based firewall system to overcome these limitations.

For the centralized DDoS-attack mitigation system, this document raises limitations in the legacy DDoS-attack mitigation techniques in terms of flexibility and administration costs. Since in many cases, network configuration for the mitigation is manually performed, it is

difficult to dynamically configure network devices to limit and control suspicious network traffic for DDoS attacks. This document introduces a use case of SDN-based DDoS-attack mitigation system to provide an autonomous and prompt configuration for suspicious network traffic.

For the centralized VoIP/VoLTE security system, this documents raises challenges in the legacy VoIP/VoLTE security system in terms of provisioning time, the granularity of security, cost, and the establishment of policy. This document shows a use case of SDN-based VoIP/VoLTE security system to resolve these challenges along in the I2NSF framework.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. Terminology

This document uses the terminology described in [RFC7149], [ITU-T.Y.3300], [ONF-OpenFlow], [ONF-SDN-Architecture], [ITU-T.X.1252], and [ITU-T.X.800]. In addition, the following terms are defined below:

- o Software-Defined Networking: A set of techniques that enables to directly program, orchestrate, control, and manage network resources, which facilitates the design, delivery and operation of network services in a dynamic and scalable manner [ITU-T.Y.3300].
- o Access Control: A procedure used to determine if an entity should be granted access to resources, facilities, services, or information based on pre-established rules and specific rights or authority associated with the requesting party [ITU-T.X.1252].
- o Access Control Policy: The set of rules that define the conditions under which access may take place [ITU-T.X.800].
- o Access Control Policy Rules: Security policy rules concerning the provision of the access control service [ITU-T.X.800].
- o Network Resources: Network devices that can perform packet forwarding in a network system. The network resources include network switch, router, gateway, WiFi access points, and similar devices.

- o Firewall: A firewall that is a device or service at the junction of two network segments that inspects every packet that attempts to cross the boundary. It also rejects any packet that does not satisfy certain criteria for disallowed port numbers or IP addresses.
- o Centralized Firewall System: A centralized firewall that can establish and distribute access control policy rules into network resources for efficient firewall management. These rules can be managed dynamically by a centralized server for firewall. SDN can work as a network-based firewall system through a standard interface between firewall applications and network resources.
- o Centralized DDoS-attack Mitigation System: A centralized mitigator that can establish and distribute access control policy rules into network resources for efficient DDoS-attack mitigation. These rules can be managed dynamically by a centralized server for DDoS-attack mitigation. SDN can work as a network-based mitigation system through a standard interface between DDoS-attack mitigation applications and network resources.
- o Centralized VoIP/VoLTE Security System: A centralized security system that handles the security issues related to VoIP and VoLTE services. SDN can work as a network-based security system through a standard interface between VoIP/VoLTE security applications and network resources.

4. Overview

This section describes the referenced architecture to support SDN-based security services, such as centralized firewall system and centralized DDoS-attack mitigation system. Also, it describes a framework for SDN-based security services using I2NSF.

As shown in Figure 1, network security functions (NSFs) as security services (e.g., firewall, DDoS-attack mitigation, VoIP/VoLTE, web filter, and deep packet inspection) run on the top of SDN controller [ITU-T.Y.3300] [ONF-SDN-Architecture]. When an administrator enforces security policies for such security services through an application interface, SDN controller generates the corresponding access control policy rules to meet such security policies in an autonomous and prompt manner. According to the generated access control policy rules, the network resources such as switches take an action to mitigate network attacks, for example, dropping packets with suspicious patterns.

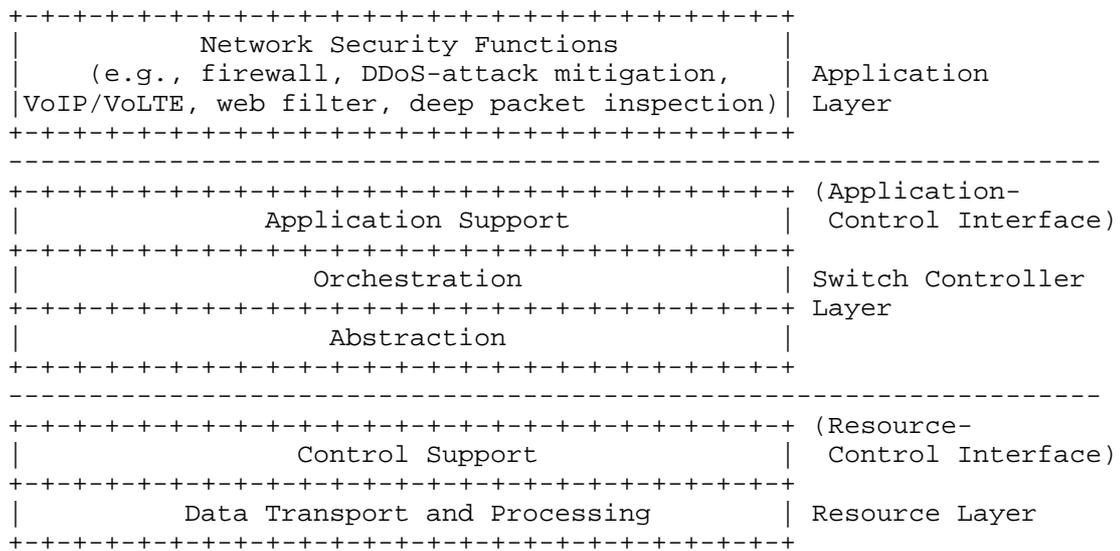


Figure 1: High-level Architecture for SDN-based Security Services

Figure 2 shows a framework to support SDN-based security services using I2NSF [i2nsf-framework]. As shown in Figure 2, I2NSF client can use security services by delivering their high-level security policies to security controller via client facing interface. Security controller asks NSFs to perform function-level security services via NSF facing interface. The NSFs run on top of virtual machines through Network Functions Virtualization (NFV) [ETSI-NFV]. NSFs ask switch controller to perform their required security services on switches under the supervision of switch controller. In addition, security controller uses registration interface to communicate with developer’s management system for registering (or deregistering) the developer’s NSFs into (or from) the NFV system using the I2NSF framework.

NSF facing interface between security controller and NSFs can be implemented by Network Configuration Protocol (NETCONF) [RFC6241] with a data modeling language called YANG [RFC6020] that describes function-level security services. A data model in [i2nsf-cap-interface-yang] can be used for the I2NSF capability interface, which is NSF facing interface.

The proposed framework of SDN-based security services can be combined to a security management architecture in [i2nsf-sec-mgmt-arch] for handling high-level security policies as well as low-level security policies.

Also, the proposed framework can enforce low-level security policies in NSFs by using a service function chaining (SFC) enabled I2NSF architecture in [i2nsf-sfc-enabled-arch].

5. Objectives

- o Prompt reaction to new network attacks: SDN-based security services allow private networks to defend themselves against new sophisticated network attacks.
- o Automatic defense from network attacks: SDN-based security services identify the category of network attack (e.g., malware and DDoS attacks) and take counteraction for the defense without the intervention of network administrators.
- o Network-load-aware resource allocation: SDN-based security services measure the overhead of resources for security services and dynamically select resources considering load balance for the maximum network performance.

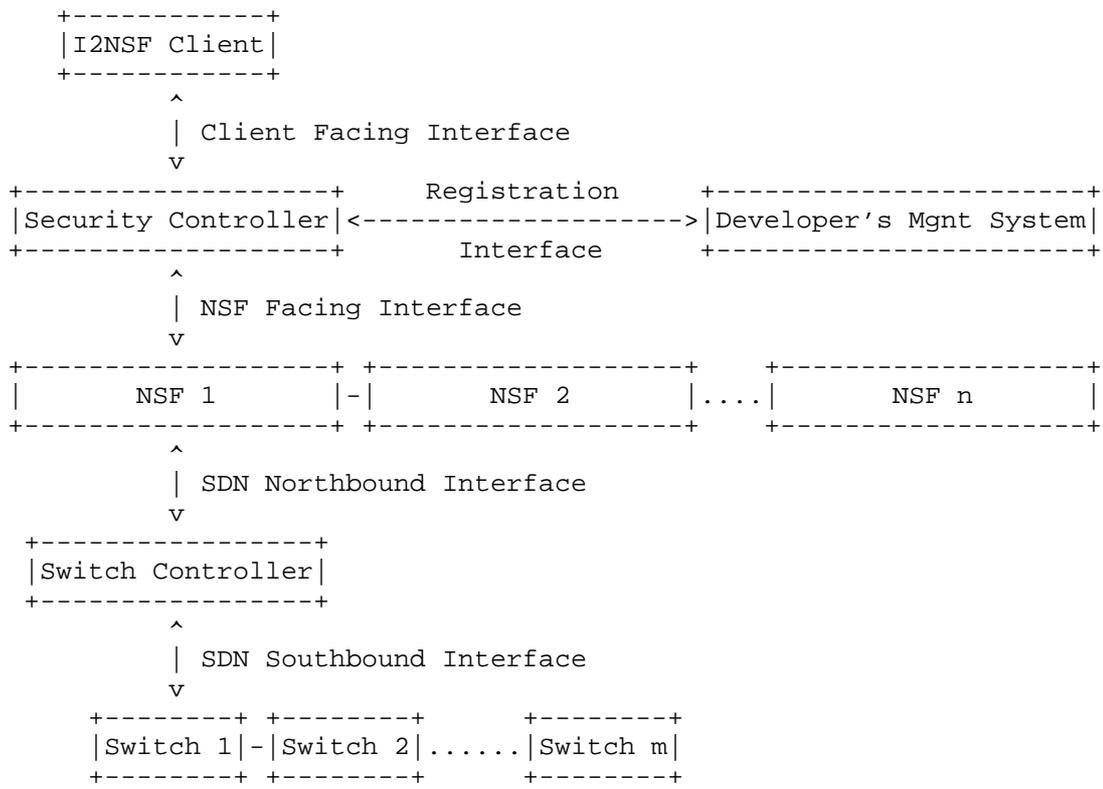


Figure 2: A Framework for SDN-based Security Services using I2NSF

6. Requirements

SDN-based security services provide dynamic and flexible network resource management to mitigate network attacks, such as malware and DDoS attacks. In order to support this capability, the requirements for SDN-based security services are described as follows:

- o SDN-based security services are required to support the programmability of network resources to mitigate network attacks.
- o SDN-based security services are required to support the orchestration of network resources and SDN applications to mitigate network attacks.
- o SDN-based security services are required to provide an application interface allowing the management of access control policies in an autonomous and prompt manner.

- o SDN-based security services are required to provide a resource-control interface for the control of network resources to mitigate network attacks.
- o SDN-based security services are required to provide the logically centralized control of network resources to mitigate network attacks.
- o SDN-based security services are required to support the seamless services to mitigate network attacks.
- o SDN-based security services are required to provide the dynamic control of network resources to mitigate network attacks.

7. Use Cases

This section introduces three use cases for security services based on SDN: (i) centralized firewall system, (ii) centralized DDoS-attack mitigation system, and (iii) centralized VoIP/VoLTE security system.

7.1. Centralized Firewall System

For the centralized firewall system, a centralized network firewall can manage each network resource and firewall rules can be managed flexibly by a centralized server for firewall (called Firewall). The centralized network firewall controls each switch for the network resource management and the firewall rules can be added or deleted dynamically.

The procedure of firewall operations in the centralized firewall system is as follows:

1. Switch forwards an unknown flow's packet to Switch Controller.
2. Switch Controller forwards the unknown flow's packet to an appropriate security service application, such as Firewall.
3. Firewall analyzes the headers and contents of the packet.
4. If Firewall regards the packet as a malware's packet with a suspicious pattern, it reports the malware's packet to Switch Controller.
5. Switch Controller installs new rules (e.g., drop packets with the suspicious pattern) into switches.
6. The malware's packets are dropped by switches.

For the above centralized firewall system, the existing SDN protocols can be used through standard interfaces between the firewall application and switches [RFC7149][ITU-T.Y.3300][ONF-OpenFlow][ONF-SDN-Architecture].

Legacy firewalls have some challenges such as the expensive cost, performance, management of access control, establishment of policy, and packet-based access mechanism. The proposed framework can resolve these challenges through the above centralized firewall system based on SDN as follows:

- o Cost: The cost of adding firewalls to network resources such as routers, gateways, and switches is substantial due to the reason that we need to add firewall on each network resource. To solve this, each network resource can be managed centrally such that a single firewall is manipulated by a centralized server.
- o Performance: The performance of firewalls is often slower than the link speed of network interfaces. Every network resource for firewall needs to check firewall rules according to network conditions. Firewalls can be adaptively deployed among network switches, depending on network conditions in the framework.
- o The management of access control: Since there may be hundreds of network resources in an administered network, the dynamic management of access control for security services like firewall is a challenge. In the framework, firewall rules can be dynamically added for new malware.
- o The establishment of policy: Policy should be established for each network resource. However, it is difficult to describe what flows are permitted or denied for firewall within a specific organization network under management. Thus, a centralized view is helpful to determine security policies for such a network.
- o Packet-based access mechanism: Packet-based access mechanism is not enough for firewall in practice since the basic unit of access control is usually users or applications. Therefore, application level rules can be defined and added to the firewall system through the centralized server.

7.2. Centralized DDoS-attack Mitigation System

For the centralized DDoS-attack mitigation system, a centralized DDoS-attack mitigation can manage each network resource and manipulate rules to each switch through a centralized server for DDoS-attack mitigation (called DDoS-attack Mitigator). The centralized DDoS-attack mitigation system defends servers against

DDoS attacks outside private network, that is, from public network.

Servers are categorized into stateless servers (e.g., DNS servers) and stateful servers (e.g., web servers). For DDoS-attack mitigation, traffic flows in switches are dynamically configured by traffic flow forwarding path management according to the category of servers [AVANT-GUARD]. Such a management should consider the load balance among the switches for the defense against DDoS attacks.

The procedure of DDoS-attack mitigation operations in the centralized DDoS-attack mitigation system is as follows:

1. Switch periodically reports an inter-arrival pattern of a flow's packets to Switch Controller.
2. Switch Controller forwards the flow's inter-arrival pattern to an appropriate security service application, such as DDoS-attack Mitigator.
3. DDoS-attack Mitigator analyzes the reported pattern for the flow.
4. If DDoS-attack Mitigator regards the pattern as a DDoS attack, it computes a packet dropping probability corresponding to suspiciousness level and reports this DDoS-attack flow to Switch Controller.
5. Switch Controller installs new rules into switches (e.g., forward packets with the suspicious inter-arrival pattern with a dropping probability).
6. The suspicious flow's packets are randomly dropped by switches with the dropping probability.

For the above centralized DDoS-attack mitigation system, the existing SDN protocols can be used through standard interfaces between the DDoS-attack mitigator application and switches [RFC7149] [ITU-T.Y.3300][ONF-OpenFlow][ONF-SDN-Architecture].

The centralized DDoS-attack mitigation system has challenges similar to the centralized firewall system. The proposed framework can resolve these challenges through the above centralized DDoS-attack mitigation system based on SDN as follows:

- o Cost: The cost of adding DDoS-attack mitigators to network resources such as routers, gateways, and switches is substantial due to the reason that we need to add DDoS-attack mitigator on each network resource. To solve this, each network resource can be managed centrally such that a single DDoS-attack mitigator is

manipulated by a centralized server.

- o Performance: The performance of DDoS-attack mitigators is often slower than the link speed of network interfaces. The checking of DDoS attacks may reduce the performance of the network interfaces. DDoS-attack mitigators can be adaptively deployed among network switches, depending on network conditions in the framework.
- o The management of network resources: Since there may be hundreds of network resources in an administered network, the dynamic management of network resources for performance (e.g., load balancing) is a challenge for DDoS-attack mitigation. In the framework, as dynamic network resource management, traffic flow forwarding path management can handle the load balancing of network switches [AVANT-GUARD]. With this management, the current and near-future workload can be spread among the network switches for DDoS-attack mitigation. In addition, DDoS-attack mitigation rules can be dynamically added for new DDoS attacks.
- o The establishment of policy: Policy should be established for each network resource. However, it is difficult to describe what flows are permitted or denied for new DDoS-attacks (e.g., DNS reflection attack) within a specific organization network under management. Thus, a centralized view is helpful to determine security policies for such a network.

7.3. Centralized VoIP/VoLTE Security System

For the centralized VoIP/VoLTE security system, a centralized VoIP/VoLTE security system can monitor each VoIP/VoLTE flow and manage VoIP/VoLTE security rules controlled by a centralized server for VoIP/VoLTE security service (called VoIP IPS). The VoIP/VoLTE security system controls each switch for the VoIP/VoLTE call flow management by manipulating the rules that can be added, deleted or modified dynamically.

The procedure of VoIP/VoLTE security operations in the centralized VoIP/VoLTE security system is as follows:

1. A switch forwards an unknown call flow's signal packet (e.g., SIP packet) to Switch Controller. Also, if the packet belongs to a matched flow's packet related to SIP (called matched SIP packet), Switch forwards the packet to Switch Controller so that the packet can be checked by an NSF for VoIP (i.e., VoIP IPS) via Switch Controller, which monitors the behavior of its SIP call.
2. Switch Controller forwards the unknown flow's packet or the matched SIP packet to an appropriate security service function,

such as VoIP IPS.

3. VoIP IPS analyzes the headers and contents of the signal packet, such as IP address, calling number, and session description [RFC4566].
4. If VoIP IPS regards the packet as a spoofed packet by hackers or a scanning packet searching for VoIP/VoLTE devices, it requests the Switch Controller to block that packet and the subsequent packets that have the same call-id.
5. Switch Controller installs new rules (e.g., drop packets) into switches.
6. The illegal packets are dropped by switches.

For the above centralized VoIP/VoLTE security system, the existing SDN protocols can be used through standard interfaces between the VoIP IPS application and switches [RFC7149][ITU-T.Y.3300][ONF-OpenFlow][ONF-SDN-Architecture].

Legacy hardware based VoIP IPSes have some challenges, such as provisioning time, the granularity of security, expensive cost, and the establishment of policy. The proposed framework can resolve these challenges through the above centralized VoIP/VoLTE security system based on SDN as follows:

- o Provisioning: The provisioning time of setting up a legacy VoIP IPS to network is substantial because it takes from some hours to some days. By managing the network resources centrally, VoIP IPS can provide more agility in provisioning both virtual and physical network resources from a central location.
- o The granularity of security: The security rules of a legacy VoIP IPS are compounded considering the granularity of security. The proposed framework can provide more granular security by centralizing security control into a switch controller. The VoIP IPS can effectively manage security rules throughout the network.
- o Cost: The cost of adding VoIP IPS to network resources, such as routers, gateways, and switches is substantial due to the reason that we need to add VoIP IPS on each network resource. To solve this, each network resource can be managed centrally such that a single VoIP IPS is manipulated by a centralized server.
- o The establishment of policy: Policy should be established for each network resource. However, it is difficult to describe what flows are permitted or denied for VoIP IPS within a specific

organization network under management. Thus, a centralized view is helpful to determine security policies for such a network.

So far this document has described the procedure and impact of the three use cases for security services. To support these use cases in the proposed framework, a data model described in [i2nsf-cap-interface-yang] can be used as NSF facing interface along with NETCONF [RFC6241].

8. Security Considerations

The proposed SDN-based framework in this document is derived from the I2NSF framework [i2nsf-framework], so the security considerations of the I2NSF framework should be included in this document. Therefore, proper secure communication channels should be used the delivery of control or management messages among the components in the proposed framework.

This document shares all the security issues of SDN that are specified in the "Security Considerations" section of [ITU-T.Y.3300].

9. Acknowledgements

This document was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) [10041244, Smart TV 2.0 Software Platform] and by MSIP/IITP [R0166-15-1041, Standard Development of Network Security based SDN].

This document has greatly benefited from inputs by Jinyong Kim, Daeyoung Hyun, Mahdi Daghmehchi-Firoozjaei, and Geumhwan Cho.

10. References

10.1. Normative References

- | | |
|-------------------|---|
| [RFC2119] | Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997. |
| [i2nsf-framework] | Lopez, E., Lopez, D., Dunbar, L., Strassner, J., Zhuang, X., Parrott, J., Krishnan, R., and S. Durbha, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-01, June 2016. |

- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

10.2. Informative References

- [i2nsf-cap-interface-yang] Jeong, J., Kim, J., Hyun, D., Park, J., and T. Ahn, "YANG Data Model of Interface to Network Security Functions Capability Interface", draft-jeong-i2nsf-capability-interface-yang-00, July 2016.
- [i2nsf-sec-mgmt-arch] Kim, H., Ko, H., Oh, S., Jeong, J., and S. Lee, "An Architecture for Security Management in I2NSF Framework", draft-kim-i2nsf-security-management-architecture-01, July 2016.
- [i2nsf-sfc-enabled-arch] Hyun, S., Woo, S., Yeo, Y., Jeong, J., and J. Park, "Service Function Chaining-Enabled I2NSF Architecture", draft-hyun-i2nsf-sfc-enabled-i2nsf-00, July 2016.
- [RFC7149] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7149, March 2014.
- [ITU-T.Y.3300] Recommendation ITU-T Y.3300, "Framework of Software-Defined Networking", June 2014.
- [ONF-OpenFlow] ONF, "OpenFlow Switch Specification (Version 1.4.0)", October 2013.
- [ONF-SDN-Architecture] ONF, "SDN Architecture", June 2014.
- [ITU-T.X.1252] Recommendation ITU-T X.1252, "Baseline Identity Management Terms and Definitions", April 2010.

- [ITU-T.X.800] Recommendation ITU-T X.800, "Security Architecture for Open Systems Interconnection for CCITT Applications", March 1991.
- [AVANT-GUARD] Shin, S., Yegneswaran, V., Porras, P., and G. Gu, "AVANT-GUARD: Scalable and Vigilant Switch Flow Management in Software-Defined Networks", ACM CCS, November 2013.
- [ETSI-NFV] ETSI GS NFV 002 V1.1.1, "Network Functions Virtualisation (NFV); Architectural Framework", October 2013.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

Appendix A. Changes from draft-jeong-i2nsf-sdn-security-services-04

The following changes were made from draft-jeong-i2nsf-sdn-security-services-04:

- o According to the change of terminology in the I2NSF framework, the names of the components and interfaces are updated as follows: Application Controller -> I2NSF Client, Security Function (SF) -> Network Security Function (NSF), Vendor System -> Developer's Management System, Service Layer Interface -> Client Facing Interface, Capability Layer Interface -> NSF Facing Interface.
- o Three use cases described in this document can use a data model corresponding to the information model for the I2NSF capability interface.
- o The proposed framework of SDN-based security services can be combined to a security management architecture for handling security policies.
- o The proposed framework can enforce low-level security policies in NSFs by using a service function chaining (SFC) enabled I2NSF architecture.

Authors' Addresses

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

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

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

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

Jung-Soo Park
Electronics and Telecommunications Research Institute
218 Gajeong-Ro, Yuseong-Gu
Daejeon 305-700
Republic of Korea

Phone: +82 42 860 6514
EMail: pjs@etri.re.kr

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

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

Se-Hui Lee
Korea Telecom
70 Yuseong-Ro, Yuseong-Gu
Daejeon 305-811
Republic of Korea

Phone: +82 42 870 8162
EMail: sehuilee@kt.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 5, 2017

H. Kim
H. Ko
S. Oh
J. Jeong
Sungkyunkwan University
S. Lee
Korea Telecom
July 4, 2016

An Architecture for Security Management in I2NSF Framework
draft-kim-i2nsf-security-management-architecture-01

Abstract

This document describes an architecture for security management in the Interface to Network Security Functions (I2NSF) framework. This security management architecture consists of I2NSF Client, Security Management System (i.e., Security Controller and Developer's Management System), and Network Security Functions (NSFs) in the I2NSF framework. I2NSF Client consists of Application Logic, Policy Updater, and Policy Collector. Security Controller consists of Security Policy Manager and NSF Capability Manager. This document explains their missions and the processing of security management in a high level. It also describes representative use cases, such as security management for the list of malware domains and security management for VoIP-VoLTE.

Status of This Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-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 January 5, 2017.

Copyright Notice

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

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

Table of Contents

1. Introduction	3
2. Objectives	3
3. Requirements Language	4
4. Terminology	4
5. Architecture of Security Management	4
5.1. Security Policy Manager	5
5.2. NSF Capability Manager	6
5.3. Developer's Management System	6
5.4. Application Logic	6
5.5. Policy Updater	6
5.6. Policy Collector	6
6. Use Cases	7
6.1. Security Management for the List of Malware Domains	7
6.2. Security Management for VoIP-VoLTE	8
7. Security Considerations	9
8. Acknowledgements	9
9. References	9
9.1. Normative References	9
9.2. Informative References	9

1. Introduction

To enforce a user's high-level security policy into the I2NSF framework [i2nsf-framework], I2NSF Client delivers such a policy to Security Controller via Client Facing Interface. In this document, an architecture for security management is proposed for a given high-level policy in the I2NSF framework. This architecture contains I2NSF Client, Security Management System (i.e., Security Controller and Developer's Management System), and NSFs in the I2NSF framework. I2NSF Client includes Application Logic, Policy Updater, and Policy Collector. Security Controller contains Security Policy Manager and NSF Capability Manager.

Security Policy Manager and NSF Capability Manager in Security Controller are responsible for controlling the updated security policy which will be given by Policy Updater in I2NSF Client via Client Facing Interface. Policy Updater delivers new or updated policies to Security Controller. On the other hand, when an event occurs for NSF to change a low-level policy, Policy Collector receives the correspondingly updated high-level policy via Security Controller. Next, it also updates accordingly the current policies in Application Logic.

In this document, we propose a security management architecture that integrates additional components for security management into the I2NSF framework. Our architecture is designed to support flexible and effective security policies. Application Logic generates the high-level policy and Policy Updater sends it to Security Policy Manager via Client Facing Interface. Security Policy Manager maps the high-level policy into several low-level policies in Security Controller. After mapping into those policies, Security Policy Manager sends them to NSF(s) so that they can be enforced into the NSF(s).

2. Objectives

This document has two main objectives for security management architecture as follows.

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

3. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

4. Terminology

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

- o Application Logic: It is a component in the security management architecture which generates high-level security policies to block or mitigate security attacks.
- o Policy Updater: It is a component which forwards a high-level security policy, which is received from Application Logic, to Security Controller.
- o Security Policy Manager: It maps a high-level security policy received from Policy Updater into low-level security policies, and vice versa.
- o NSF Capability Manager: It is a component which stores the NSF capability registered by Developer's Management System via Registration Interface and shares it to Security Policy Manager to generate the corresponding low-level security policies.
- o Policy Collector: It is a component that forwards the updated high-level security policy to Application Logic.

5. Architecture of Security Management

This section describes a security management architecture in I2NSF and focuses on Security Management System having Security Controller and Developer's Management System. It also explains basic operations of Security Controller. In addition, it describes the details about each component of the architecture.

Figure 1 shows a security management architecture in the I2NSF framework. The architecture is designed to support the enforcement of flexible and effective security policies. Application Logic in I2NSF Client generates a high-level policy in accordance with new security attacks and then Policy Updater in I2NSF Client sends such a policy to Security Policy Manager in Security Controller. Security Policy Manager maps the high-level policy into several low-level policies relevant to NSF capability registered into NSF Capability Manager. After such a mapping into low-level policies, Security

policy happens in NSF, NSF sends the changed low-level policy to Security Policy Manager via NSF Facing Interface. Security Policy Manager maps such changed low-level policy into the high-level policy and sends it to Policy Collector via Client Facing Interface.

5.2. NSF Capability Manager

NSF Capability Manager is a component integrated into Security Controller. It stores the NSF capability registered by Developer's Management System via Registration Interface and shares it to Security Policy Manager so that Security Policy Manager can generate low-level policies relevant to a given NSF capability. Moreover, whenever a new NSF is registered, NSF Capability Manager requests Developer's Management System to register the NSF capability into the management table of NSF Capability Manager via Registration Interface. On the other hand, when the existing NSF is deleted, NSF Capability Manager eliminates the NSF capability from its management table.

5.3. Developer's Management System

Developer's Management System is a component which registers a new NSF's capability to NSF Capability Manager via Registration Interface. Moreover, in the case where there is some update in the registered NSF, such an update will be delivered from Developer's Management System to NSF Capability Manager.

5.4. Application Logic

Application Logic is a component which generates a high-level security policy to block or mitigate security attacks. It sends the generated policies to Policy Updater. However, this component is out of our standardization scope. We explain its detailed operations in two use cases in Section 6.

5.5. Policy Updater

Policy Updater is a component which receives a high-level security policy generated by Application Logic and delivers it to Security Policy Manager via Client Facing Interface.

5.6. Policy Collector

Policy Collector is a component which receives the updated high level security policy from Security Controller via Client Facing Interface. Such an update is required because the corresponding low-level security policy is updated by some event that occurred in an NSF. After receiving it, Policy Collector forwards it to Application Logic

so that Application Logic can update the corresponding high-level security policy received from Security Controller.

6. Use Cases

A generic architecture is designed to react to possible security attacks. This section shows the procedure of the defense for security attacks in the I2NSF framework [i2nsf-framework] for a given list of security attacks in malware domains and VoIP/VoLTE security attacks.

6.1. Security Management for the List of Malware Domains

Malware domain blacklisting maintains and publishes the blacklists of IP addresses of possible attacking hosts, servers, and networks that are suspicious of malicious activities. Figure 2 shows a security management architecture for Malware Domain Blacklisting.

Based on the malware domain blacklisting, the list of malware domains can be updated either manually or automatically by Malware Domain Manager in I2NSF Client. Also, Malware Domain Manager periodically generates a new high-level security policy to prevent the delivery of packets from/to those newly added malware domains and enforce the low-level security policies in NSF. It sends the new high-level security policy to Policy Updater, which forwards it to Security Controller.

An updated low-level policy is sent by an NSF to Security Controller via NSF Facing Interface so that Security Controller can generate the corresponding high-level security policy. Security Controller delivers the high-level security policy to Policy Collector. Policy Collector forwards the policy to Malware Domain Manager as an Application Logic.

VoIP-VoLTE attackers and enforce the low-level security policies in NSF. It sends the new high-level security policy to Policy Updater, which forwards it to Security Controller.

An updated low-level policy for VoIP-VoLTE attacks is sent by an NSF to Security Controller via NSF Facing Interface so that Security Controller can generate the corresponding high-level security policy, such as IP addresses, user-agents, and expire time values that need to be added by Security Controller. Security Controller delivers the high-level security policy to Policy Collector. Policy Collector forwards the policy to VoIP-VoLTE Security Manager as an Application Logic.

7. Security Considerations

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

8. Acknowledgements

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

9. References

9.1. Normative References

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

9.2. Informative References

[i2nsf-framework] Lopez, E., Lopez, D., Dunbar, L., Strassner, J., Zhuang, X., Parrott, J., Krishnan, R., and S. Durbha, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-00 , May 2016.

Authors' Addresses

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

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

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

Phone: +82-31-299-4104
EMail: skoh21@skku.edu

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

Phone: +82-31-299-4104
EMail: osh09@skku.edu

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

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

Se-Hui Lee
Korea Telecom
70 Yuseong-Ro, Yuseong-Gu
Daejeon 305-811
Republic of Korea

Phone: +82 42 870 8162
EMail: sehuilee@kt.com

I2NSF Working Group
Internet-Draft
Intended status: Informational
Expires: January 7, 2017

R. Kumar
A. Lohiya
Juniper Networks
D. Qi
Bloomberg
X. Long
July 6, 2016

Northbound Interfaces for Security Policy Controllers : A Framework and
Information Model
draft-kumar-i2nsf-controller-northbound-framework-00

Abstract

This document provides a framework and information model for the definition of northbound interfaces for a security policy controller. The interfaces are based on user-intent instead of vendor-specific or device-centric approaches that would require deep knowledge of vendor products and their security features. The document identifies the common interfaces needed to enforce the user-intent-based policies onto network security functions (NSFs) irrespective of how those functions are realized. The function may be physical or virtual in nature and may be implemented in networking or dedicated appliances.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 7, 2017.

Copyright Notice

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

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

Table of Contents

1. Introduction	2
2. Conventions Used in this Document	3
3. Security Provisioning Framework	4
3.1. Deployment Models for Implementing Security Policies	5
3.2. Client Perspective on Security Policy Configuration and Management	8
4. Functional Requirements for the Northbound Interface	8
4.1. Multi-Tenancy and RBAC for Policy Management	9
4.2. Policy Lifecycle Management	10
4.3. Policy Endpoint Groups	10
4.4. Policy Rules	11
4.5. Policy Actions	12
4.6. Third-Party Integration	13
4.7. Telemetry Data	13
5. Operational Requirements for the Northbound Interface	13
5.1. API Version	13
5.2. API Extensibility	14
6. IANA Considerations	14
7. Acknowledgements	14
8. Normative References	14
Authors' Addresses	14

1. Introduction

Programming security policies in a network is a fairly complex task and requires very deep knowledge of the vendors' devices in order to implement a security policy. This has been the biggest challenge for both Service Providers and Enterprise, henceforth known as end-customers, to keep up-to-date with the security of their networks and assets. The challenge is amplified due to virtualization because security appliances come in both physical and virtual forms and are supplied by a variety of vendors who have their own proprietary interfaces to manage and implement the security policies on their devices.

Even if an end-customer deploys a single vendor solution across its entire network, it is difficult to manage security policies due to the complexity of network security features available in the devices. The end-customer may use a vendor-provided management system that gives some abstraction in the form of GUI and helps in provisioning and managing security policies. The single vendor approach is highly restrictive in today's network as explained below:

- o The end-customer cannot rely on a single vendor because one vendor may not be able keep up to date with its security needs.
- o The large end-customer may have a presence across different sites and regions and that may mean it is not possible to have a single vendor solution due to technical or business reasons.
- o If and when the end-customer migrates from one vendor to another, it is not possible to migrate security policies from one management system to another without complex manual work.
- o Due to virtualization within data centers, end-customers are using physical and virtual forms of security functions with a wide variety of vendors, including open source, to control their costs.
- o The end-customer might choose various devices in the network (such as routers, switches, firewall devices, and overlay-networks) as enforcement points for security policies for any reason (such as network design simplicity, cost, most-effective place, scale and performance).

In order to provide the end-customer with a solution where they can deploy security policies across different vendors and devices whether physical or virtual, the Interface to Network Security Functions (I2NSF) working group in the IETF is defining a set of northbound interfaces.

This document discusses the requirements for these northbound interfaces and describes a framework and information model so that these interfaces can be easily used by end-customer security administrators without knowledge of specific security devices or features. We refer to this as "user-intent-based interfaces".

2. Conventions Used in this Document

BSS: Business Support System.

CMDB: Configuration Management Database.

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

FW: Firewall.

IDS: Intrusion Detection System.

IPS: Intrusion Protection System.

LDAP: Lightweight Directory Access Protocol.

NSF: Network Security Function, defined by [I-D.ietf-i2nsf-problem-and-use-cases].

OSS: Operation Support System.

RBAC: Role Based Access Control.

SIEM: Security Information and Event Management.

URL: Universal Resource Locator.

vNSF: Refers to NSF being instantiated on Virtual Machines.

3. Security Provisioning Framework

The IETF I2NSF working group has defined a framework for Interfaces to Network Security Functions that defines following terminology:

Client: A client could be a GUI system used by a security administrator, an OSS/BSS system used by an end-customer, or a security controller system or application in the end-customer's management system.

Client-Facing Interface: A client-facing interface is an interface used to configure and manage security a framework across the entire network independent of device-specific interface so that same interface can be used for any device from any vendor.

The "Client Facing Interface" ensures that an end-customer can deploy any device from any vendor and still be able to use same consistent interface. In essence, these interfaces give a framework to manage end-customer's security policies. Henceforth in this document, we "security policy management interface" interchangeably when we refer to these northbound interfaces.

3.1. Deployment Models for Implementing Security Policies

This document describes a framework for security policy management interfaces. This document does not describe a framework for southbound interface: those may be defined in another draft.

Traditionally, medium and larger end-customers deploy management systems to manage their security policies. This approach may not be suitable for modern datacenters that are virtualized and manage their resources using controllers.

There are two different deployment models:

- a. Management without an explicit management system for control of devices and NSFs. In this deployment, the security policy controller acts as a NSF policy management system that takes information passed over the northbound policy interface and translates into data on the I2NSF southbound interface. The I2NSF interfaces are implemented by security device/function vendors. This would usually be done by having an I2NSF agent embedded in the security device or NSF. This deployment model is shown in Figure 1.

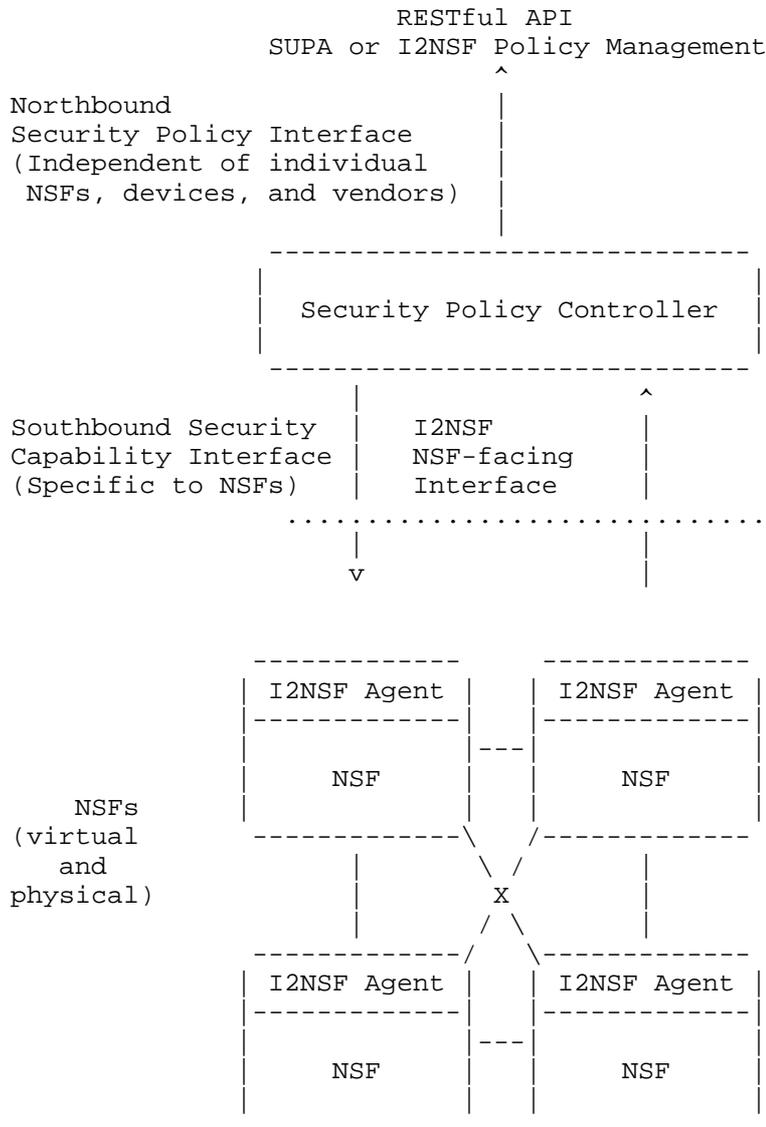


Figure 1: Deployment without Management System

- b. Management with an explicit management system for control of devices and NSFs. This model is similar to the model above except that security policy controller interacts with a dedicated management system which could either proxy I2NSF southbound interfaces or could provide a layer where security devices or

NSFs do not support an I2NSF agent to process I2NSF southbound interfaces. This deployment model is shown in Figure 2.

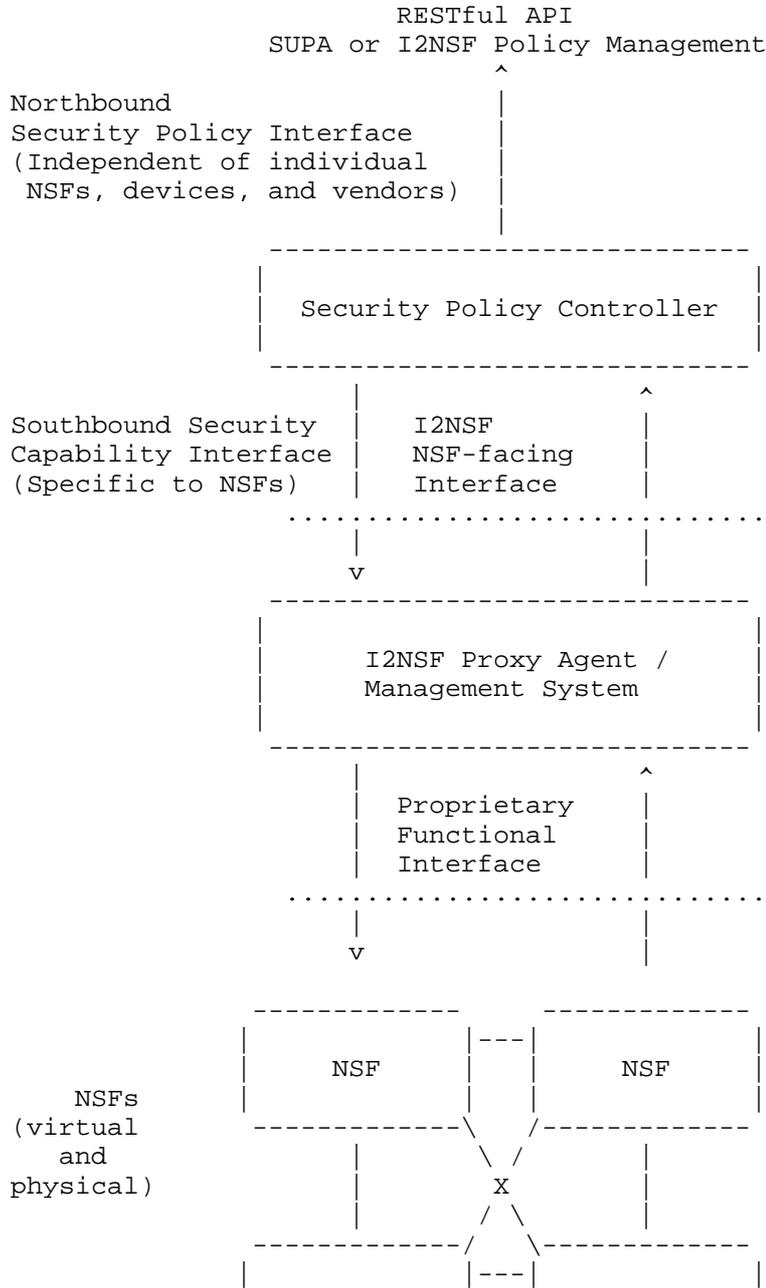




Figure 2: Deployment with Management System or I2NSF Proxy Agent

Although the deployment models discussed here don't necessarily affect the northbound security policy interface, they do give an overall context for defining a security policy interface based on abstraction.

3.2. Client Perspective on Security Policy Configuration and Management

In order to provide I2NSF northbound interface for security policies to client that are not specific to any vendor, device or feature implementation, it is important that security policies shall be configured and managed from a client's perspective. We refer to this as the user-intent based model since it is primarily driven by how security administrators view security policies from the deployment perspective.

The client perspective ensures that policy management is not only easy to understand for them (the actual users), but is also independent of vendor, device, and specific implementation which is the foremost goal for a northbound interface.

4. Functional Requirements for the Northbound Interface

As mentioned earlier, it is important that the northbound interface be primarily driven by user-intent which is what a client understands well. In order to define this interface, we must understand the requirements and framework used by the security administrator.

A security policy that is based on user-intent is completely agnostic of how this policy is enforced in the end-customer's network. The security controller may choose to implement such a policy on any device (router, switch, firewall) in a physical or virtual form factor. The security policy controller's implementation is outside the scope of this document and the I2NSF working group.

At a high level, the objects that are required in order to express and build the security policies fall into the following categories:

- o Multi-tenancy and RBAC for policy management
- o Policy lifecycle management

- o Policy endpoint groups
- o Policy rules
- o Policy actions
- o Third party integration
- o Telemetry data

The above categories are by no means a complete list and may not be sufficient for all use-cases and all end-customers, but should be a good start for a wide variety of use-cases in both Service Provider networks and Enterprise networks.

The following sections provide further details on the above mentioned security policies categories.

4.1. Multi-Tenancy and RBAC for Policy Management

An end-customer that uses security policies may have internal tenants and would like to have a framework wherein each tenant manages its own security policies to provide isolation across different tenants.

An end-customer may be a cloud service provider with multi-tenant deployments where each tenant is a different organization and must allow complete isolation across different tenants.

The RBAC objects and method needed to build such a framework is defined below.

Policy-Tenant: An entity that owns and manages the security policies.

Policy-User: A user within a Policy-Tenant authorized to manage security policies for that tenant.

Policy-Authorization-Role: A role assigned to a Policy-User that determines whether the user has read-write access, read-only access, or no access for certain resources.

Authentication and Authorization Scheme: There must be a scheme for a Policy-User to be authenticated and authorized to use the policy controller.

4.2. Policy Lifecycle Management

In order to provide more sophisticated security framework, there should be a mechanism to express that a policy becomes dynamically active/enforced or inactive based on either security administrator intervention or an event.

One example of dynamic policy management is when the security administrator pre-configures all the security policies, but the policies get activated/enforced or deactivated based on dynamic threats faced by the end-customer. Basically, a threat event may activate certain inactive policies, and once a new event indicates that the threat has gone away, the policies become inactive again.

The northbound interface should support the following mechanisms for policy enforcement:

Admin-Enforced: The policy, once configured, remains active/enforced until removed by the security administrator.

Time-Enforced: The policy configuration specifies the time profile that determines when policy is activated/enforced.

Event-Enforced: The policy configuration specifies the event profile that determines when policy is activated/enforced.

4.3. Policy Endpoint Groups

Typically, when the security administrator configures a security policy, the intention is to apply this policy to certain subsets of the network. The subsets may be identified based on criteria such as users, devices, and applications. We refer to such a subset of the network as a "Policy Endpoint Group".

One of the biggest challenges for a security administrator is how to make sure that security policies remain effective while constant changes are happening to the "Policy Endpoint Group" for various reasons (e.g., organizational changes). If the policy is created based on static information such as user names, application, or network subnets, then every time that this static information changes policies would need to be updated. For example, if a policy is created that allows access to an application only from the group of Human Resource users (the HR-users group), then each time the HR-users group changes, the policy needs to be updated.

Changes to policy could be highly taxing to the end-customer for various operational reasons. The policy management framework must allow "Policy Endpoint Group" to be dynamic in nature so that changes

to the group (HR-users in our example) automatically result in updates to its content.

We call these dynamic Policy Endpoint Groups "Meta-data Driven Groups". The meta-data is a tag associated with endpoint information such as users, applications, and devices. The mapping from meta-data to dynamic content could come either from standards-based or proprietary tools. The security controller could use any available mechanisms to derive this mapping and to make automatic updates to the policy content if the mapping information changes.

The northbound policy interface must support endpoint groups for user-intent based policy management. The following meta-data driven groups are typically used for configuring security polices:

User-Group: This group identifies a set of users based on a tag or on static information. The tag to user information is dynamically derived from systems such as Active Directory or LDAP. For example, an end-customer may have different user-groups, such as HR-users, Finance-users, Engineering-users, to classify a set of users in each department.

Device-Group: This group identifies a set of devices based on a tag or on static information. The tag to device information is dynamically derived from systems such as CMDB. For example, an end-customer may want to classify all machines running one operating system into one group and machines running another operating system into another group.

Application-Group: This group identifies a set of applications based on a tag or on static information. The tag to application information is dynamically derived from systems such as CMDB. For example, an end-customer may want to classify all applications running in the Legal department into one group and all applications running under a specific operating system into another group.

Location-Group: This group identifies a set of locations based on a tag or on static information. The tag to location information is dynamically derived from systems such as CMDB. For example, an end-customer may want to classify all sites/locations in a geographic region as one group.

4.4. Policy Rules

The security policy rules can be as simple as specifying a match for the user or application specified through "Policy Endpoint Group" and take one of the "Policy Actions" or more complicated rules that

specify how two different "Policy Endpoint Groups" interact with each other. The northbound interface must support mechanisms to allow the following rule matches.

Policy Endpoint Groups: The rule must allow a way to match either a single or a member of a list of "Policy Endpoint Groups".

There must also be a way to express whether a group is a source or a destination so that the security administrator can apply the rule in only one direction of a communication.

There must also be a way to express a match between two "Policy Endpoint Groups" so that a policy can be effective for communication between two groups.

Direction: The rule must allow a way to express whether the security administrator wants to match the "Policy Endpoint Group" as the source or destination. The default should be to match both directions if the direction rule is not specified in the policy.

Threats: The rule should allow the security administrator to express a match for threats that come either in the form of feeds (such as botnet feeds, GeoIP feeds, URL feeds, or feeds from a SIEM) or speciality security appliances.

The threat could be from malware and this requires a way to match for virus signatures or file hashes.

4.5. Policy Actions

The security administrator must be able to configure a variety of actions within a security policy. Typically, security policy specifies a simple action of "deny" or "permit" if a particular rule is matched. Although this may be enough for most of the simple policies, the I2NSF northbound interface must also provide a more comprehensive set of actions so that the interface can be used effectively across various security functions.

Permit: This action means continue processing the next rule or allow the packet to pass if this is the last rule.

Deny: This action means stop further rule processing and drop the packet.

Drop connection: This action means stop further rule processing, drop the packet, and drop connection (for example, by sending a TCP reset).

Log: This action means create a log entry whenever a rule is matched.

Authenticate connection: This action means that whenever a new connection is established it should be authenticated.

Quarantine/Redirect: This action may be relevant for event driven policy where certain events would activate a configured policy that quarantines or redirects certain packet flows.

4.6. Third-Party Integration

The security policies in the end-customer's network may require the use of specialty devices such as honeypots, behavioral analytics, or SIEM in the network, and may also involve threat feeds, virus signatures, and malicious file hashes as part of comprehensive security policies.

The northbound interface must allow the security administrator to configure these threat sources and any other information to provide integration and fold this into policy management.

4.7. Telemetry Data

One of the most important aspect of security is to have visibility into the networks. As threats become more sophisticated, the security administrator must be able to gather different types of telemetry data from various devices in the network. The collected data could simply be logged or sent to security analysis engines for behavioral analysis and for threat detection.

The northbound interface must allow the security administrator to collect various kinds of data from NSFs. The data source could be syslog, flow records, policy violation records, and other available data.

5. Operational Requirements for the Northbound Interface

5.1. API Version

The northbound interface must support a version number for each RESTful API. This is very important because the client application and the controller application will most likely come from different vendors. Even if the vendor is same, it is hard to imagine that two different applications would be released in lock step.

Without API versioning, it hard to debug and figure out issues if application breaks. Although API versioning does not guarantee that

applications will always work, it helps in debugging if the problem is caused by an API mismatch.

5.2. API Extensibility

Abstraction and standardization of the northbound interface is of tremendous value to end-customers as it gives them the flexibility of deploying any vendors' NSF. However this might also look like as an obstacle to innovation.

If an NSF vendor comes up with new feature or functionality that can't be expressed through the currently defined northbound interface, there must be a way to extend existing APIs or to create a new API that is relevant for that NSF vendor only.

6. IANA Considerations

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

7. Acknowledgements

The editors would like to thank Adrian Farrel for helpful discussions and advice.

8. Normative 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-00 (work in progress), February 2016.

Authors' Addresses

Rakesh Kumar
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
US

Email: rkkumar@juniper.net

Anil Lohiya
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
US

Email: alohiya@juniper.net

Dave Qi
Bloomberg
731 Lexington Avenue
New York, NY 10022
US

Email: DQI@bloomberg.net

Xiaobo Long
4 Cottonwood Lane
Warren, NJ 07059
US

Email: long.xiaobo@gmail.com

RFC Beautification Working Group
Internet-Draft
Intended status: Informational
Expires: December 30, 2016

D. Migault
A. Ranjbar
Ericsson
June 28, 2016

Collaboration Agreement for Security Service Function
draft-mglt-i2nsf-ssf-collaboration-00.txt

Abstract

This document specifies a collaboration agreement protocol. The collaboration agreement makes possible individual security services functions (SSF) to collaborate with each other. The collaboration is mostly intended for SSF located in different administrative domains, in which case the collaboration cannot be performed by a shared orchestrator.

The collaboration between SSF in different domains assumes the traffic is steered through the two domains.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 30, 2016.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Requirements notation	2
2. Introduction	2
3. Collaboration Agreement	3
4. Collaboration Agreement Protocol	4
5. Collaboration Agreement Management operations	6
6. Error Message handling	6
7. Payload Format	7
7.1. Collaboration Agreement Objects	7
7.2. Collaboration Agreement Protocol	11
7.2.1. Collaboration Agreement Protocol Request	11
7.2.2. Collaboration Agreement Protocol Response	12
7.3. Collaboration Agreement Protocol Additional Operations	12
8. Security Considerations	13
9. IANA Considerations	13
10. Acknowledgements	13
11. Normative References	13
Authors' Addresses	13

1. Requirements notation

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].

2. Introduction

Security Service Function (SSF) has been deployed to mitigate and detect malicious traffic and security threats in networks.

A typical use case would consider today's cloud-based services where a data flow is forwarded from the Internet Service Provider to the cloud which hosts the destination service or any on-path services. The services deployed in the cloud are at least partly implemented using a combination of one or more SSF. Similarly, the ISP may also implement a set of on path SSF. The purpose of the collaboration is to enable a SSF running in the cloud administrative domain to take advantage of specific SSF running in the ISP administrative domain. The SSF may be of same type or of different type.

As the SSFs belong to different administrative domains, collaboration between these two SSFs is unlikely to happen through a common shared

orchestrator. To enable the collaboration between individual SSFs, a collaboration agreement protocol is proposed in this document. This protocol is expected to provide: better detection by exchanging real-time information about the detected attacks between SSFs, better mitigation by enforcing mitigation strategies on more effective network segments (e.g. cloud vs ISP), and better resource usage by eliminating the need for frequent deployment of similar service functions and by spreading the tasks among different SSFs.

3. Collaboration Agreement

The SSFs initiating and accepting the collaboration are called, respectively, 'initiator' and 'provider'. The initiator sends to potential providers a Collaboration Agreement (CA), which defines the necessary attributes involved in the collaboration. Such attributes are expected not to be SSF specific. However attributes that characterize the SSF, such as the SFF type, input and output flows, may be part of the CA simply to allow collaborators to define whether or not they are eligible to provide the corresponding services.

For management purposes, the collaboration agreement should also include an 'agreement ID' and a 'duration' indicating its lifespan. It is the responsibility of the 'initiator' to renew the agreement before it expires, although the 'provider' should also be able to notify the former that the agreement needs to be revised or interrupted earlier due to some unexpected event.

Two collaboration modes are envisioned:

- 1) 'Resilient', in which the provider is expected to handle the whole load of that traffic; and
- 2) 'Best Effort', which indicates that the provider supplies the service for a fraction of the load. When the Best Effort mode is chosen, an 'alternate path' indicates where non-treated traffic is forwarded, and 'resource' indicates the resources allocated for the service. The use of 'alternate path' enhances the collaboration between SSFs by allowing the provider to temporarily assign specific amount of resources for handling the packets and send the non-treated traffic through the alternate path to be processed by the initiator. The resources assigned in the Best Effort mode can be expressed in specific ways, such as a combination of various computational resources e.g., CPU, I/O, bandwidth, packet rate, or maximum latency. The manner by which such resources are controlled is left for the provider's implementation (e.g., by leveraging containers and micro services technologies).

Here are the parameters associated to the collaboration agreement:

- o `ca_id`: identifies the collaboration agreement and it can be used later to refer to a specific collaboration agreement.
- o `initiator`: designates the locators (e.g. IP address or FQDN) as well as the authentication credentials associated to the initiator.
- o `provider`: see `initiator`
- o `collaboration type`: indicates the type of collaboration (Best Effort and Resilient).
- o `resource`: designates the resources agreed on between the initiator and the provider. Note that this parameter is optional as resources are only negotiated when collaboration is in a best effort mode.
- o `SSF type`: designates the type of the security instances running.
- o `expiration time`: designates the expiration of the collaboration.
- o `interconnections`: defines how the interconnection between the initiator and the provider is performed. This includes the definition of the alternate path.
- o `direction`: defines if the provider is expected to be in front of the initiator or behind it.

4. Collaboration Agreement Protocol

The purpose of the collaboration agreement protocol is to negotiate between the initiator and provider and make an agreement for cooperation between SSFs placed in a single or multiple domains. Currently, the collaboration agreement protocol is always originated from the initiator. In other words, the provider is not initiating the exchanges as to announce what it can provide.

The collaboration agreement protocol should include the following attributes:

- o `ca_id`: this is the collaboration agreement identifier. In a case the value is not acceptable, an `ERROR_UNACCEPTABLE_CA_ID` MUST be returned. There are, however, little reasons such a collision occurs. If such a collision occurs, the negotiation is aborted and must be restarted with a new `ca_id`.

- o initiator: it includes information that the initiator offers to the provider. Upon receiving the request for collaboration, the provider may reject the collaboration agreement by sending a `ERROR_UNACCEPTABLE_INITIATOR`.
- o provider: it consists of information about the provider to be verified by the initiator. Upon receiving this information, the initiator may abort the negotiation with `ERROR_UNACCEPTABLE_PROVIDER`. The reason for refusing the provider, may be that the provider is not in a white list or that the provider has been explicitly banned by the initiator.
- o resource: it represents allocated resources by the provider for collaboration with the initiator. When the collaboration mode is set to Resilient, the resource is not expected to be provided by the provider. For the best effort mode, the resource provided by the provider may consider the indication provided by the initiator or not. Given the resource provided by the provider, the initiator is likely to close the collaboration or to accept it.

In order to define a flexible framework, the negotiation steps between the initiator and provider is designed as mentioned below:

1. The initiator provides a list of proposals to the provider
2. A proposal may contain multiple proposition for a given attribute. For example, let P1 be a proposal offered by the initiator. In this case, the initiator may be willing to make an agreement with the providers either in Best Effort or Resilient modes. In this case, the initiator will set P1 with an object of collaboration type set to Best Effort AND an object of collaboration type set to Resilient.
3. When multiple proposals are received by the provider, the provider is expected to choose a single proposal. The chosen proposal is the one that contains the attributes that fits the provider.
4. When a proposal is chosen, the provider must select for each attribute the preferred value. More especially, when multiple values for a same attribute type are available, the provider selects the preferred value for that attribute. Also, the chosen proposal must have the same amount of attribute types which means the provider is not allowed to remove some attributes or selectively reject attributes.

5. The provider may send an acceptable proposal to the initiator. If none of the proposals are acceptable by the provider, the provider returns a `ERROR_UNACCEPTABLE_PROPOSALS`.

5. Collaboration Agreement Management operations

Once the collaboration agreement has been agreed between the initiator and provider, the following actions need to be considered during its life cycle.

- o `END_AGREEMENT`: This action can be performed by either peers, that is to say the initiator or the provider. This action requires the `ca_id` and credentials to identify peers in the agreement.
- o `UPDATE_EXPIRATION_DATE`: This action is initiated by either peers. It intends to update the expiration date. The expiration date can be extended or advanced. The input parameters are the `ca_id` and the new expiration date. The possible responses are to accept or reject this request. In case of rejection, `ERROR_UNACCEPTABLE_NEW_EXPIRATION_DATE` is sent to the requested peer.
- o `UPDATE_RESOURCE`: This action is expected to be triggered by the provider. It indicates the amount of resources the provider offers for collaboration. This is an informative message. It may be useful for the initiator to know how much resource will be dedicated to the collaboration by the provider so it can adjust its strategy.
- o `REDIRECT_SSF`: This action is triggered when peers change their location. This action is initiated by either peers. It may result in changes in Alternate Path in case of Best Effort mode.

6. Error Message handling

The following Error message have been considered so far:

```
ERROR_UNACCEPTABLE_CA_ID
ERROR_UNACCEPTABLE_PROVIDER
ERROR_UNACCEPTABLE_INITIATOR
ERROR_UNACCEPTABLE_PROPOSALS
ERROR_UNACCEPTABLE_NEW_EXPIRATION_DATE
```

7. Payload Format

7.1. Collaboration Agreement Objects

This section represents the Collaboration Agreement object. The collaboration agreement is an object with properties. Some of these properties are object themselves. In order to enrich the object definition, the Collaboration is defined on different objects including 'peer' and 'resource' objects.

'Peer' object represents the necessary information associated to a peer. A peer can be either the initiator or provider. The description of a peer object is as follows:

```
{
  "peer":{
    "type": "object",
    "description": "provides different elements associated to the
                  initiator or the provider. This includes
                  location as well as authentication credentials",
    "properties": {
      "rsakey": {
        "type": "string",
        "description": "RSA public key used to identify the
                      initiator"
      },
      "cert": {
        "type": "array",
        "description": "list of certificates to authenticate the
                      initiator"
      },
      "fqdn": {
        "type": "string",
        "description": " FQDN associated to the initiator"
      },
      "ipv4": {
        "type": "string",
        "description": "IPv4 address used to reach the initiator"
      },
      "ipv6": {
        "type": "string",
        "description": "IPv6 address used to reach the initiator"
      }
    }
  }
}
```

The following object designates the resources agreed between the initiator and the provider.

```
{
  "resource": {
    "type": "object",
    "description": "resource engaged into the collaboration",
    "properties": {
      "cpu": {
        "type": "number",
        "description": "cpu limit"
      },
      "memory": {
        "type": "number",
        "description": "memory limit"
      },
      "net": {
        "type": "number",
        "description": "net limit"
      },
      "blkio": {
        "type": "number",
        "description": "block limit"
      }
    }
  }
}
```

The collaboration type is defined as follow:

TYPE	CODE
Resilient	0
Best Effort	1

SSF instance types can be extended to any number of available services. We do not limit SSF types and we expect to extend this number in future. Some example SSFs can be defined as follows:

TYPE	CODE
Rate limiting	0
DNSoverTCP	1
PacketDropper	2

The following object is a Collaboration Agreement object which includes several properties to define an agreement between the provider and initiator.

```
{
  "type": "Collaboration Agreement",
  "description": "This object designates the Collaboration
                 Agreement properties",
  "properties": {
    "ca_id" : {
      "type": "number",
      "description" : "unique identifier of the Collaboration
                     agreement"
    },
    "initiator": {
      "type": "peer",
      "description": "provides the different elements associated
                     to the initiator. This includes location
                     as well as authentication credentials"
    },
    "provider": {
      "type": "peer",
      "description": "provides the different elements associated
                     to the provider. This includes location as
                     well as authentication credentials"
    },
    "collaboration_type": {
      "type": "number",
      "description": "defines whether the type of the
                     collaboration"
    },
    "security_service_instance_type": {
      "type": "number",
      "description": "the type of security service instance"
    },
    "interconnections":{
      "type": "interconnections",
      "description": "the type of security service instance"
    },
    "resource":{
      "type": "resource",
      "description": "the type of security service instance"
    },
    "direction":{
      "type": "direction",
      "description": "indicates whether the provider MUST
                     be placed downstream or upstream"
    }
  }
}
```

7.2. Collaboration Agreement Protocol

The collaboration agreement protocol can be defined as request or response objects.

7.2.1. Collaboration Agreement Protocol Request

The following object defines the request object which includes information about initiator, resources and a set of proposal objects.

```
{
  "type": "collaboration protocol agreement request",
  "description": "object",
  "properties": {
    "ca_id" : {
      "type": "number",
      "description" : "unique identifier for collaboration
                    agreement"
    },
    "initiator":{
      "type": "peer",
      "description": "provides different elements associated
                    to the initiator. This includes location
                    as well as authentication credentials"
    },
    "informative resource requested":{
      "type": "resource",
      "description": "the type of security service instance"
    },
    "proposals":{
      "type": "Array",
      "description": "Array of proposals offered by the initiator"
    }
  }
}
```

A proposal object can also be defined as follows:

```

{
  "type": "object",
  "description": "A single proposal with a set of attributes.
                 The expected attribute types are collaboration
                 type, security service instance type and
                 interconnections",
  "properties": {
    "proposal_id": {
      "type": "number"
    }
    "proposed-attribute": {
      "type": "object"
      "properties": {
        "attribute-type": {
          type: string
        }
        "attribute-values": {
          "type": "array"
          "items": {
            attribute-value
          }
        }
      }
    }
  }
}

```

7.2.2. Collaboration Agreement Protocol Response

The response object is similar to the request object except that:

- o The response must include a provider object.
- o The proposed list of attribute values must be of size one with the chosen value.

7.3. Collaboration Agreement Protocol Additional Operations

When the initiator and provider are placed in different domains, additional orchestration operations might be needed between domains to make an agreement. Moreover, in case of Best Effort mode, additional operations is needed to establish an alternate path and separate the treated traffic from non-treated traffic e.g. by deploying classifiers on the path.

8. Security Considerations
9. IANA Considerations
10. Acknowledgements
11. 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>>.

Authors' Addresses

Daniel Migault
Ericsson
8400 boulevard Decarie
Montreal, QC H4P 2N2
Canada

Phone: +1 514-452-2160
Email: daniel.migault@ericsson.com

Alireza Ranjbar
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Phone: +358-442992904
Email: alireza.ranjbar@ericsson.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: January 5, 2017

A. Pastor
D. Lopez
Telefonica I+D
A. Shaw
Hewlett Packard Labs
July 4, 2016

Remote Attestation Procedures for Network Security Functions (NSFs)
through the I2NSF Security Controller
draft-pastor-i2nsf-vnsf-attestation-03

Abstract

This document describes the procedures a client can follow to assess the trust on an external NSF platform and its client-defined configuration through the I2NSF Security Controller. The procedure to assess trustworthiness is based on a remote attestation of the platform and the NSFs running on it performed through a Trusted Platform Module (TPM) invoked by the Security Controller.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 5, 2017.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

- 1. Introduction 3
- 2. Requirements Language 3
- 3. Establishing Client Trust 4
 - 3.1. First Step: Client-Agnostic Attestation 4
 - 3.2. Second Step: Client-Specific Attestation 4
 - 3.3. Trusted Computing 5
- 4. NSF Attestation Principles 7
 - 4.1. Requirements for a Trusted NSF Platform 8
 - 4.1.1. Trusted Boot 8
 - 4.1.2. Remote Attestation Service 9
 - 4.1.3. Secure Boot 10
- 5. Remote Attestation Procedures 10
 - 5.1. Trusted Channel with the Security Controller 11
 - 5.2. Security Controller Attestation 13
 - 5.3. Platform Attestation 14
- 6. Security Considerations 14
- 7. IANA Considerations 14
- 8. References 15
 - 8.1. Normative References 15
 - 8.2. Informative References 15
- Authors' Addresses 15

1. Introduction

As described in [I-D.pastor-i2nsf-merged-use-cases], the use of externally provided NSF implies several additional concerns in security. The most relevant threats associated with a externalized virtual platform are detailed in [I-D.ietf-i2nsf-framework]. As stated there, mutual authentication between the user and the NSF environment and, what is more important, the attestation of the elements in this environment by clients could address these threats to an acceptable level of risk. In particular:

- o Any impersonation attempt (of the client or the NSF environment) will be minimized by mutual authentication, and since appropriate records of such authentications will be made available, events will be suitable for auditing in the case of an incident.
- o Attestation of the NSF environment, especially when performed periodically, will allow clients to detect the alteration of the processing elements, or the installation of malformed elements, and mutual authentication will provide again an audit trail.
- o Attestation relying on independent Trusted Third Parties will alleviate the impact of malicious activity on the side of the provider by issuing the appropriate alarms in the event of any NSF environment manipulation.
- o While it is true that any environment is vulnerable to malicious activity with full physical access (and this is obviously beyond the scope of this document), the application of attestation mechanisms raises the degree of physical control necessary to perform an untraceable malicious modification of the environment.

The client can have a proof that their NSFs and policies are correctly (from the client point of view) enforced by the Security Controller. Taking into account the threats identified in [I-D.ietf-i2nsf-framework], this document first identifies the user expectations regarding remote trust establishment, briefly analyzes Trusted Computing techniques, and finally describes the proposed mechanisms for remote establishment of trust through the Security Controller.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [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.

3. Establishing Client Trust

From a high-level standpoint, in any I2NSF platform, the client connects and authenticates to the Security Controller, which then initialises the client's NSFs and policies. Afterwards, user traffic from the client domain goes through the NSF platform which hosts the corresponding NSFs. The user's expectations of the platform behavior are thus twofold:

- o The user traffic will be treated according to the client-specified NSFs and policies, and no other processing will be performed by the Security Controller or the platform itself (e.g. traffic eavesdropping).
- o Each NSF (and its corresponding policies) behaves as configured by the client.

We will refer to the attestation of these two expectations as the "client-agnostic attestation" and the "client-specific attestation". Trusted Computing techniques play a key role in addressing this expectations.

3.1. First Step: Client-Agnostic Attestation

This is the first interaction between a client and a Security Controller: the client wants an attestation that proves it is connected to a genuine Security Controller before continuing with the authentication. In this context, two properties characterise the genuineness of the Security Controller:

1. That the identity of the Security Controller is correct
2. That it will process the client credentials and set up the client NSFs and policies properly.

Once these two properties are proven to the client, the client knows that their credentials will only be used by the Security Controller to set up the execution of their NSFs.

3.2. Second Step: Client-Specific Attestation

From the security enforcement point of view, the client agnostic attestation focuses on the initialization of the execution platform

for the vNSFs. This second step aims to prove to clients that their security is enforced accordingly with their choices (i.e. NSFs and policies). The attestation can be performed at the initialization of the NSFs, before any user traffic is processed by the NSFs, or during the execution of the NSFs.

Support of static NSF attestation is REQUIRED for a Security Controller managing NSFs, and MUST be performed before any user traffic is redirected through any set of NSFs. The Security Controller MUST provide a proof to the client that the instantiated NSFs and policies are the ones chosen.

Additionally to the NSF attestation at the moment of their instantiation, a continuous attestation of the NSF execution (based on the generation of periodic TPM integrity measurements) MAY be required by a client to ensure their security.

3.3. Trusted Computing

In a nutshell, Trusted Computing (TC) aims at answering the following question: "As a user or administrator, how can I have some assurance that a computing system is behaving as it should?". The major enterprise level TC initiative is the Trusted Computing Group [TCG], which has been established for more than a decade, that primarily focuses on developing TC for commodity computers (servers, desktops, laptops, etc.).

The overall scheme proposed by TCG for using Trusted Computing is based on a step-by-step extension of trust, called a Chain of Trust. It uses a transitive mechanism: if a user can trust the first execution step and each step correctly attests the next executable software for trustworthiness, then a user can trust the system.

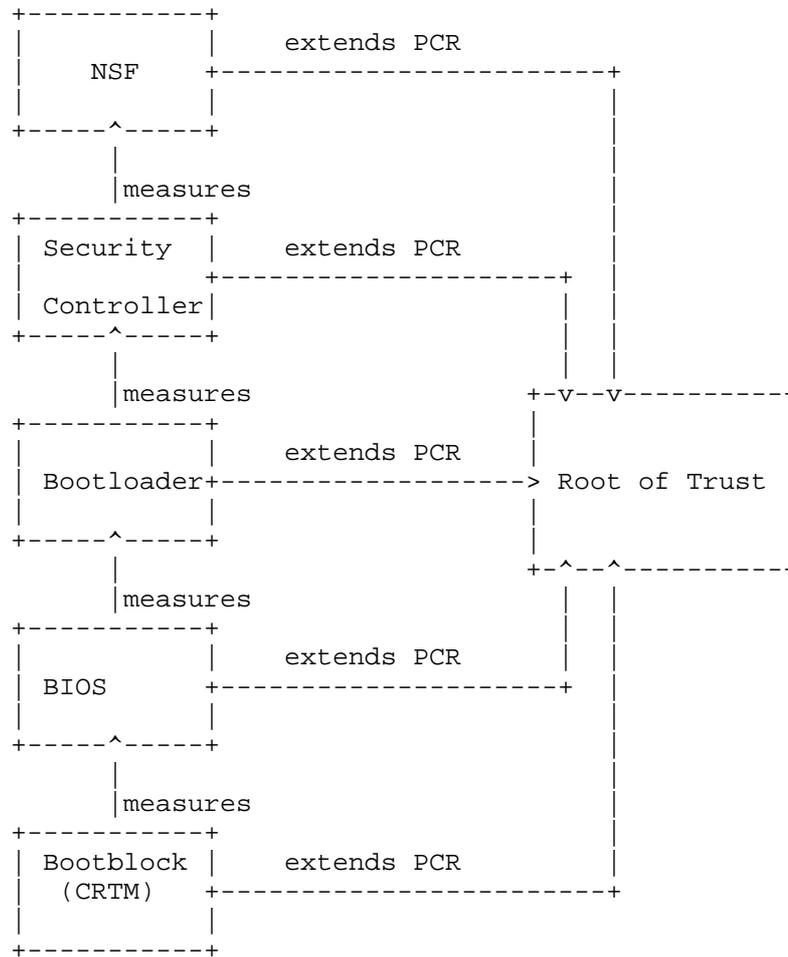


Figure 1: Applying Trusted Computing

Effectively, during the loading of each piece of software, the integrity of each piece of software is measured and stored inside a log that reflects the different boot stages, as illustrated in the figure above. Later, at the request of a user, the platform can present this log (signed with the unique identity of the platform), which can be checked to prove the platform identity and attest the state of the system. The base element for the extension of the Chain of Trust is called the Core Root of Trust.

The TCG has created a standard for the the design and usage of a secure cryptoprocessor to address the storage of keys, general

secrets, identities and platform integrity measurements: the Trusted Platform Module (TPM). When using a TPM as a root of trust, measurements of the software stack are stored in special on-board Platform Configuration Registers (PCRs) on a discrete TPM. There are normally a small number of PCRs that can be used for storing measurements, however it is not possible to directly write to a PCR; instead measurements must be stored using a process called Extending PCRs.

The extend operation can update a PCR by producing a global hash of the concatenated values of the previous PCR value with the new measurement value. The Extend operation allows for an unlimited number of measurements to be captured in a single PCR, since the size of the value is always the same and it retains a verifiable ordered chain of all the previous measurements.

Attestation of the virtualization platform will thus rely on a process of measuring the booted software and storing a chained log of measurements, typically referred to as Trusted Boot. The user will either validate the signed set of measurements with a trusted third party verifier who will assess whether the software configuration is trusted, or the user can check for themselves against their own set of reference digest values (measurements) that they have obtained a priori, and having already known the public endorsement key of the remote Root of Trust.

Trusted Boot should not be confused with a different mechanism known as "Secure Boot", as they both are designed to solve different problems. Secure Boot is a mechanism for a platform owner to lock a platform to only execute particular software. Software components that do not match the configuration digests will not be loaded or executed. This mechanism is particularly useful in preventing bootkits from successfully infecting a platform on reboot. A common standard for implementing Secure Boot is described in [UEFI]. Secure Boot only enforces a particular configuration of software, it does not allow a user to attest or quote for a series of measurements.

4. NSF Attestation Principles

Following the general requirements described in [I-D.ietf-i2nsf-framework] the Security Controller will become the essential element to implement the measurements described above, relaying on a TPM for the Root of Trust.

A mutual authentication of clients and the Security Controller MUST be performed, establishing the desired level of assurance. This level of assurance will determine how stringent are the requirements

for authentication (in both directions), and how detailed the collected measurements and their verification will be. Furthermore, the NSF platform MUST run a TPM, able to collect measurements of the platform itself, the Security Controller, and the NSFs being executed. The Security Controller MUST make the attestation measurements available to the client, directly or by means of a Trusted Third Party.

As described in [I-D.ietf-i2nsf-framework], a trusted connection between the client and the Security Controller MUST be established and all traffic to and from the NSF environment MUST flow through this connection

NOTE: The reference to results from WGs such as NEA and SACM is currently under consideration and will be included here.

4.1. Requirements for a Trusted NSF Platform

Although a discrete hardware TPM is RECOMMENDED, relaxed alternatives (such as embedded CPU TPMs, or memory and execution isolation mechanisms) MAY also be applied when the required level of assurance is lower. This reduced level of assurance MUST be communicated to the user by the Security Controller during the initial mutual authentication phase.

4.1.1. Trusted Boot

NOTE: This section is derived from the original version of the document, focused on virtual NSFs. Although it seems to be applicable to any modern physical appliance, we must be sure all these considerations are 100% applicable to physical NSFs as well, and provide exceptions when that is not the case. Support from expert in physical node attestation is required here.

All clients who interact with a Security Controller MUST be able to:

- a. Identify the Security Controller based on the public key of a Root of Trust.
- b. Retrieve a set of measurements of all the base software the Security Controller has booted (i.e. the NSF platform).

This requires that firmware and software MUST be measured before loading, with the resulting value being used to extend the appropriate PCR register. The general usage of PCRs by each software component SHOULD conform to open standards, in order to make verifying attestation reports interoperable, as it is the case of TCG Generic Server Specification [TCGGSS].

As well as for providing a signed audit log of boot measurements, the PCR values can also be used as an identity for dynamically decrypting encrypted blobs on the platform (such as encryption keys or configurations that belong to operating system components). Software can choose to submit pieces of data to be encrypted by the Root of Trust (which has its own private asymmetric key and PCR registers) and only have it decrypted based on a criteria. This criteria can be that the platform booted into a particular state (e.g. a set of PCR values). Once the desired criteria is described and the sensitive data is encrypted by the root of trust, the data has been sealed to that platform state. The sealed data will only be decrypted when the platform measurements held in the root of trust match the particular state.

Trusted Boot requires the use of a root of trust for safely storing measurements and secrets. Since the Root of Trust is self-contained and isolated from all the software that is measured, it is able to produce a signed set of platform measurements to a local or remote user. Trusted Boot however does not provide enforcement of a configuration, since the root of trust is a passive component not in the execution path, and is solely used for safe independent storage of secrets and platform measurements. It will respond to attestation requests with the exact measurements that were made during the software boot process. Sealing and unsealing of sensitive data is also a strong advantage of Trusted Boot, since it prevents leakage of secrets in the event of an untrusted software configuration.

4.1.2. Remote Attestation Service

A service MUST be present for providing signed attestation report (e.g. the measurements) from the Root of Trust (RoT) to the client. In case of failure to communicate with the service, the client MUST assume the service cannot be trusted and seek an alternative Security Controller.

Since some forms of RoT require serialised access (i.e. due to slow access to hardware), latency of getting an attestation report could increase with simultaneous requests. Simultaneous requests could occur if multiple Trusted Third Parties (TTP) request for attestation reports at the same time. This MAY be improved through batching of requests, in a special manner. In a typical remote attestation protocol, the client sends a random number ("nonce") to the RoT in order to detect any replay attacks. Therefore, caching of an attestation report does not work, since there is the possibility that it may not be a fresh report. The solution is to batch the nonce for each requestor until the RoT is ready for creating the attestation report. The report will be signed by the embedded identity of the RoT to provide data integrity and authenticity, and the report will

include all the nonces of the requestors. Regardless of the number of the number of nonces included, the requestor verifying the attestation report MUST check to see if the requestor's nonce was included in order to detect replay attacks. In addition to the attestation report containing PCRs, an additional report known as an SML (Secure Measurement Log) can be returned to the requestor to provide more information on how to verify the report (e.g. how to reproduce the PCR values). The integrity of the SML is protected by a PCR measurement in the RoT. An example of an open standard for responses is [TCGIRSS]. Further details are discussed in Section 5.2.

As part of initial contact, the Security Controller MAY present a list of external TTPs that the client can use to verify it. However, the client MUST assess whether these external verifiers can be trusted. The client can also choose to ignore or discard the presented verifiers.

Finally, to prevent malicious relaying of attestation reports from a different host, the authentication material of the secure channel (e.g. TLS, IPSec, etc.) SHOULD be bound to the RoT and verified by the connected client, unless the lowest levels of assurance have been chosen and an explicit warning issued. This is also addressed in Section 5.1.

4.1.3. Secure Boot

Using a mechanism such as Secure Boot helps provide strong prevention of software attacks. Furthermore, in combination with a hardware-based TPM, Secure Boot can provide some resilience to physical attacks (e.g. preventing a class of offline attacks and unauthorised system replacement). For NSF providers, it is RECOMMENDED that Secure Boot is employed wherever possible with an appropriate firmware update mechanism, due to the possible threat of software/firmware modifications in either public places or privately with inside attackers.

5. Remote Attestation Procedures

The establishment of trust with the Security Controller and the NSF platform consists of three main phases, which need to be coordinated by the client:

1. Trusted channel with the Security Controller. During this phase, the client securely connects to the Security Controller to avoid that any data can be tampered with or modified by an attacker if the network cannot be considered trusted. The establishment of

the trusted channel is completed after the next step.

2. Security Controller attestation. During this phase, the client verifies that the Security Controller components responsible for handling the credentials and for the isolation with respect to other potential clients are behaving correctly. Furthermore, it is verified that the identity of the platform attested is the same of the one presented by the Security Controller during the establishment of the secure connection.
3. Platform attestation. During this step, that can be repeated periodically until the connection is terminated, the Security Controller verifies the integrity of the elements composing the NSF platform. The components responsible for this task have been already attested during the previous phase.

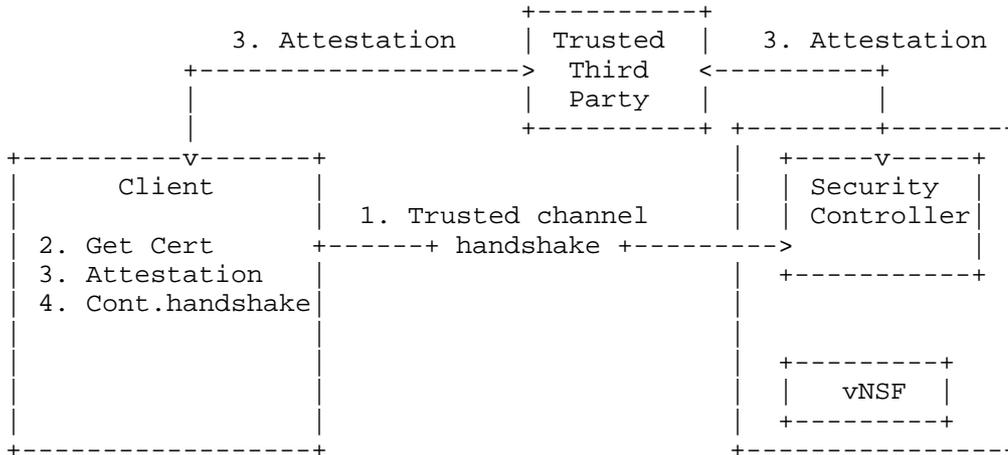


Figure 2: Steps for remote attestation

In the following each step, as depicted in the above figure, is discussed in more detail.

5.1. Trusted Channel with the Security Controller

A trusted channel is an enhanced version of the secured channel that, differently from the latter, requires the integrity verification of the contacted endpoint by the other peer during the initial handshake. However, simply transmitting the integrity measurements

over the channel does not guarantee that the platform verified is the channel endpoint. The public key or the certificate for the secure communication MUST be included as part of the measurements presented by the contacted endpoint during the remote attestation. This way, a malicious platform cannot relay the attestation to another platform as its certificate will not be present in the measurements list of the genuine platform.

In addition, the problem of a potential loss of control of the private key must be addressed (a malicious endpoint could prove the identity of the genuine endpoint). This is done by defining a long-lived Platform Property Certificate. Since this certificate connects the platform identity to the AIK public key, an attacker cannot use a stolen private key without revealing his identity, as it may use the certificate of the genuine endpoint but cannot create a quote with the AIK of the other platform.

Finally, since the platform identity can be verified from the Platform Property Certificate, the information in the certificate to be presented during the establishment of a secure communication is redundant. This allows for the use of self-signed certificates, what would simplify operational procedures in many environments, especially when they are multi-tenant. Thus, in place of certificates signed by trusted CAs, the use of self-signed certificates (which still need to be included in the measurements list) is RECOMMENDED.

The steps required for the establishment of a trusted channel with the Security Controller are as follows:

1. The client begins the trusted channel handshake with the selected Security Controller.
2. The certificate of the Security Controller is collected and used for verifying the binding of the attestation result to the contacted endpoint.
3. The client performs the remote attestation protocol with the Security Controller, either directly or with the help of a Trusted Third Party. The Trusted Third Party MAY perform the verification of attestation quotes on behalf of multiple clients.
4. If the result of the attestation is positive, the application continues the handshake and establishes the trusted channel. Otherwise, it closes the connection.

5.2. Security Controller Attestation

During the establishment of the trusted channel, the client attests the Security Controller by verifying the identity of the contacted endpoint and its integrity. Initially the Security Controller measures all the hardware and software components involved in the boot process of the vNSF platform, in order to build the chain of trust.

Since a client may not have enough capabilities to perform the integrity verification of a Security Controller the client MAY request the status of a Security Controller to a Trusted Third Party (TTP), which is in charge of communicating with it. This choice has the additional advantage of preventing an attacker from easily determining the software running at the Security Controller.

If the client directly performs the remote attestation it performs the following steps:

1. Ask the Security Controller to generate an integrity report with the format defined in [TCGIRSS].
2. The Security Controller retrieves the measurements and asks the TPM to sign the PCRs with an Attestation Identity Key (AIK). This signature provides the client with the evidence that the measurements received belong to the Security Controller being attested.
3. Once the integrity report has been generated it is sent back to the client.
4. The client first checks if the integrity report is valid by verifying the quote and the certificate associated to the AIK, and then determines if the Security Controller is behaving as expected, i.e. its software has not been compromised and isolation among the clients connected to it is enforced. As part of the verification, the client also checks that the digest of the certificate, received during the trusted channel handshake, is present among measurements.

If the client has limited computation resources, or requires an independent external element whom he can trust the measurements from, it may contact a TTP it may contact a TTP which, in turn, attests the Security Controller and returns the result of the integrity evaluation to the client, following the same steps depicted above.

5.3. Platform Attestation

The main outcome of the Security Controller attestation is to detect whether or not it is correctly configuring the operational environment for NSFs to be managed by the connecting client (the NSF platform, or just platform) in a way that any user traffic is processed only by these NSFs within the platform. Platform attestation, instead, evaluates the integrity of the NSFs running within the platform.

Platform attestation does not imply a validation of the mechanisms the Security Controller can apply to select the appropriate NSFs to enforce the Service Policies applicable to specific flows. The selection of these NSFs is supposed to happen independently of the attestation procedures, and trust on the selection process and the translation of policies into function capabilities has to be based on the trust clients have on the Security Controller being attested as the one it was intended to be used. An attestation of the selection and policy mapping procedures constitute an interesting research matter, but it is out of the scope of this document.

The procedures are essentially similar to the ones described in the previous section. This step MAY be applied periodically if the level of assurance selected by the user requires it.

Attesting NSFs, especially if they are running as virtual machines, can become a rather costly operation, especially if periodic monitoring is required by the requested level of assurance, and there are several proposals to make them feasible, from the proposal of virtual TPMs in [VTPM] to the application of Virtual Machine Introspection through an integrity monitor described by [VMIA].

6. Security Considerations

This document is specifically oriented to security and it is considered along the whole text.

7. IANA Considerations

This document requires no IANA actions.

8. References

8.1. Normative References

- [I-D.ietf-i2nsf-framework]
elopez@fortinet.com, e., Lopez, D., Dunbar, L., Strassner, J., Zhuang, X., Parrott, J., Krishnan, R., and S. Durbha, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-02 (work in progress), July 2016.
- [I-D.pastor-i2nsf-merged-use-cases]
Pastor, A., Lopez, D., Wang, K., Zhuang, X., Qi, M., Zarny, M., Majee, S., Leymann, N., Dunbar, L., and M. Georgiades, "Use Cases and Requirements for an Interface to Network Security Functions", draft-pastor-i2nsf-merged-use-cases-00 (work in progress), June 2015.
- [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>>.
- [TCG] "Trusted Computing Group (TCG)", <<https://www.trustedcomputinggroup.org/>>.
- [TCGGSS] "TCG Generic Server Specification, Version 1.0", <<http://www.trustedcomputinggroup.org/>>.
- [TCGIRSS] "Infrastructure Work Group Integrity Report Schema Specification, Version 1.0", <<https://www.trustedcomputinggroup.org/>>.

8.2. Informative References

- [UEFI] "UEFI Specification Version 2.2 (Errata D), Tech. Rep.".
- [VMIA] Schiffman, J., Vijayakumar, H., and T. Jaeger, "Verifying System Integrity by Proxy", <<http://dl.acm.org/citation.cfm?id=2368379>>.
- [VTPM] "vTPM:Virtualizing the Trusted Platform Module", <<https://www.usenix.org/legacy/events/sec06/tech/berger.html>>.

Authors' Addresses

Antonio Pastor
Telefonica I+D
Zurbaran, 12
Madrid, 28010
Spain

Phone: +34 913 128 778
Email: antonio.pastorperales@telefonica.com

Diego R. Lopez
Telefonica I+D
Zurbaran, 12
Madrid, 28010
Spain

Phone: +34 913 129 041
Email: diego.r.lopez@telefonica.com

Adrian L. Shaw
Hewlett Packard Labs
Long Down Avenue
Bristol, BS34 8QZ
UK

Phone: +44 117 316 2877
Email: als@hpe.com

I2NSF
Internet Draft
Intended status: Standard Track

L. Xia
J. Strassner
Huawei
K. Li
D.Zhang
Alibaba
E. Lopez
Fortinet
N. BOUTHORS
Qosmos
Luyuan Fang
Microsoft

Expires: December 2016

June 29, 2016

Information Model of Interface to Network Security Functions
Capability Interface
draft-xia-i2nsf-capability-interface-im-06.txt

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-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 December 29, 2016.

Copyright Notice

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

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

Abstract

This draft is focused on the capability interface of NSFs (Network Security Functions) and proposes its information model for managing the various network security functions.

Table of Contents

1. Introduction	4
2. Conventions used in this document	5
2.1. Terminology	5
3. Overall Analysis of Security Capability	6
3.1. Network Security	7
3.2. Content Security	9
3.3. Attack Mitigation	11
4. Information Model Design	11
4.1. Overall Structure	11
4.2. Information Sub-Model for Network Security Capabilities	14
4.3. Information Sub-Model for Network Security	14
4.3.1. Network Security Policy Rule Extensions	15
4.3.1.1. AuthenticationECAPolicyRule Class Definition	17
4.3.1.2. AuthorizationECAPolicyRuleClass Definition ..	19
4.3.1.3. AccountingECAPolicyRuleClass Definition	21
4.3.1.4. TrafficInspectionECAPolicyRuleClass Definition	23
4.3.1.5. ApplyProfileECAPolicyRuleClass Definition ...	25
4.3.1.6. ApplySignatureECAPolicyRuleClass Definition .	27
4.3.2. Network Security Policy Rule Operation	29
4.3.3. Network Security Event Sub-Model	30
4.3.3.1. UserSecurityEvent Class Description	32
4.3.3.1.1. The usrSecEventContent Attribute	32
4.3.3.1.2. The usrSecEventFormat Attribute	32
4.3.3.1.3. The usrSecEventType Attribute	33
4.3.3.2. DeviceSecurityEvent Class Description	33
4.3.3.2.1. The devSecEventContent Attribute	33
4.3.3.2.2. The devSecEventFormat Attribute	34
4.3.3.2.3. The devSecEventType Attribute	34

4.3.3.2.4.	The devSecEventTypeInfo[0..n] Attribute	34
4.3.3.2.5.	The devSecEventTypeSeverity Attribute	.. 35
4.3.3.3.	SystemSecurityEvent Class Description 35
4.3.3.3.1.	The sysSecEventContent Attribute 35
4.3.3.3.2.	The sysSecEventFormat Attribute 36
4.3.3.3.3.	The sysSecEventType Attribute 36
4.3.3.4.	TimeSecurityEvent Class Description 36
4.3.3.4.1.	The timeSecEventPeriodBegin Attribute	.. 37
4.3.3.4.2.	The timeSecEventPeriodEnd Attribute	... 37
4.3.3.4.3.	The timeSecEventTimeZone Attribute 37
4.3.4.	Network Security Condition Sub-Model 37
4.3.4.1.	PacketSecurityCondition 39
4.3.4.1.1.	PacketSecurityMACCondition 39
4.3.4.1.1.1.	The pktSecCondMACDest Attribute	... 40
4.3.4.1.1.2.	The pktSecCondMACSrc Attribute	... 40
4.3.4.1.1.3.	The pktSecCondMAC8021Q Attribute	.. 40
4.3.4.1.1.4.	The pktSecCondMACEtherType Attribute	40
4.3.4.1.1.5.	The pktSecCondMACTCI Attribute	... 40
4.3.4.1.2.	PacketSecurityIPv4Condition 40
4.3.4.1.2.1.	The pktSecCondIPv4SrcAddr Attribute	40
4.3.4.1.2.2.	The pktSecCondIPv4DestAddr Attribute	40
4.3.4.1.2.3.	The pktSecCondIPv4ProtocolUsed Attribute 41
4.3.4.1.2.4.	The pktSecCondIPv4DSCP Attribute	.. 41
4.3.4.1.2.5.	The pktSecCondIPv4ECN Attribute	... 41
4.3.4.1.2.6.	The pktSecCondIPv4TotalLength Attribute 41
4.3.4.1.2.7.	The pktSecCondIPv4TTL Attribute	... 41
4.3.4.1.3.	PacketSecurityIPv6Condition 41
4.3.4.1.3.1.	The pktSecCondIPv6SrcAddr Attribute	41
4.3.4.1.3.2.	The pktSecCondIPv6DestAddr Attribute	41
4.3.4.1.3.3.	The pktSecCondIPv6DSCP Attribute	.. 41
4.3.4.1.3.4.	The pktSecCondIPv6ECN Attribute	... 42
4.3.4.1.3.5.	The pktSecCondIPv6FlowLabel Attribute	42
4.3.4.1.3.6.	The pktSecCondIPv6PayloadLength Attribute 42
4.3.4.1.3.7.	The pktSecCondIPv6NextHeader Attribute	42
4.3.4.1.3.8.	The pktSecCondIPv6HopLimit Attribute	42
4.3.4.1.4.	PacketSecurityTCPCondition 42
4.3.4.1.4.1.	The pktSecCondTPCSrcPort Attribute	42
4.3.4.1.4.2.	The pktSecCondTPCDestPort Attribute	42
4.3.4.1.4.3.	The pktSecCondTCPSeqNum Attribute	. 43
4.3.4.1.4.4.	The pktSecCondTCPFlags Attribute	.. 43
4.3.4.1.5.	PacketSecurityUDPCondition 43
4.3.4.1.5.1.	The pktSecCondUDPSrcPort Attribute	43
4.3.4.1.5.2.	The pktSecCondUDPDestPort Attribute	43
4.3.4.1.5.3.	The pktSecCondUDPLength Attribute	. 43

4.3.4.2. PacketPayloadSecurityCondition	43
4.3.4.3. TargetSecurityCondition	43
4.3.4.4. UserSecurityCondition	44
4.3.4.5. SecurityContextCondition	44
4.3.4.6. GenericContextSecurityCondition	44
4.3.5. Network Security Action Sub-Model	45
4.3.5.1. IngressAction	46
4.3.5.2. EgressAction	46
4.3.5.3. ApplyProfileAction	46
4.3.5.4. ApplySignatureAction	46
4.4. Information Model for Content Security Control	46
4.5. Information Model for Attack Mitigation Control	47
5. Security Considerations	48
6. IANA Considerations	48
7. References	49
7.1. Normative References	49
7.2. Informative References	49
8. Acknowledgments	49
Appendix A.	50

1. Introduction

The rapid development of cloud computing, along with the demand of cloud-based security services, requires advanced security protection in various scenarios. Examples include network devices in an enterprise network, User Equipment (UE) in a mobile network, devices in the Internet of Things (IoT), or residential access users [I-D.draft-ietf-i2nsf-problem-and-use-cases].

According to [I-D.draft-ietf-i2nsf-framework], there are two types of I2NSF interfaces available for security rules provisioning:

- o Interface between I2NSF clients and a security controller: This is a service-oriented interface, whose main objective is to define a communication channel over which information defining security services can be requested. This enables security information to be exchanged between various applications (e.g., OpenStack, or various BSS/OSS components) and other components (e.g., security controllers). The design goal of the service interface is to decouple the security service in the application layer from various kinds of security devices and their device-specific security functions.

- o Interface between NSFs (e.g., firewall, intrusion prevention, or anti-virus) and a security controller. This interface is independent of how the NSFs are implemented (e.g., run in Virtual Machines (VMs) or physical appliances). In this document, this type of interface is also referred to as the "capability interface". Capabilities are functions that NSFs can perform. This interface is used to advertise, select, and activate capabilities of selected NSFs in a vendor-independent manner.

The capability interface is used to decouple the security management scheme from the set of NSFs that implement this scheme, and through this interface, an NSF can advertise its security functions to its controller.

The information model proposed in this draft is about the functions of an NSF, but is limited to managing part of the capability interface. Note that the monitoring of security functions is out of scope.

This document is organized as follows: Section 3 is an analysis of security capability for the I2NSF capability interface. Section 4 presents the detailed structure and content of the information model. Section 4 specifies the information model of security policy in Routing Backus-Naur Form [RFC5511].

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].

This document references to [I-D.draft-ietf-i2nsf-terminology] for more specific security related and I2NSF scoped terminology definitions.

2.1. Terminology

AAA -Access control, Authorization, Authentication

ACL - Access Control List

AD - Active Directory

ANSI - American National Standards Institute

DDoS - Distributed Deny of Services

FW - Firewall

I2NSF - Interface to Network Security Functions

INCITS - International Committee for Information Technology Standards

IoT - Internet of Things

IPS - Intrusion Prevention System

LDAP - Lightweight Directory Access Protocol

NAT - Network Address Translation

NBI - North-bound Interface

NIST - National Institute of Standard Technology

NSF - Network Security Function

RBAC - Role Based Access Control

UE - User Equipment

URL - Uniform/Universal Resource Locator

VM - Virtual Machine

WAF - Web Application Firewall

3. Overall Analysis of Security Capability

At present, a variety of NSFs produced by multiple security vendors provide various security capabilities to customers. Multiple NSFs can be combined together to provide security services over the given network traffic, regardless of whether the NSFs are implemented as physical or virtual functions.

Most of today's security capabilities fall into several common categories, including network security control, content security control, and attack mitigation control. Each category further covers more specific security capabilities, which are described below.

3.1. Network Security

Network security is a category that describes the inspecting and processing of network traffic based on pre-defined security policies.

The inspecting portion may be thought of as a packet-processing engine that inspects packets traversing networks, either directly or in context to flows with which the packet is associated. From the perspective of packet-processing, implementations differ in the depths of packet headers and/or payloads they can inspect, the various flow and context states they can maintain, and the actions that can be applied to the packets or flows.

The "Event-Condition-Action" (ECA) policy rule set in [I-D.draft-ietf-i2nsf-framework] is used here as the basis for the security rule design:

- o Event: An Event is defined as any important occurrence in time of a change in the system being managed, and/or in the environment of the system being managed. When used in the context of policy rules for I2NSF, it is used to determine whether the Condition clause of the Policy Rule can be evaluated or not. Examples of an I2NSF Event include time and user actions (e.g., logon, logoff, and actions that violate an ACL);
- o Condition: A set of attributes, features, and/or values that are to be compared with a set of known attributes, features, and/or values in order to make a decision. When used in the context of policy rules for I2NSF, it is used to determine whether or not the set of Actions in that Policy Rule can be executed or not. The following are exemplary types of conditions:
 - Packet content values: Refer to the kind of information or attributes acquired directly from the packet headers or payloads that can be used in the security policy. It can be any fields or attributes in the packet L2/L3/L4 header, or special segment of bytes in the packet payload;
 - Context values: Refer to the context information for the received packets. It can be (and not limited to):

- * User: The user (or user group) information to which a network flow is associated. A user has many attributes, such as name, id, password, authentication mode, and so on. The combination of name and id (where id could be a password, a certificate, or other means of identifying the user) is often used in the security policy to identify the user. For example, if an NSF is aware of the IP (or MAC) address associated with the user, the NSF can use a pre-defined or dynamically learned name-address association to enforce the security functions for this given user (or user group);
 - * Schedule: Time or time range when packet or flow is received;
 - * Region: The geographic location where network traffic is received;
 - * Target: The target indicates the entity to which the security services are applied. This can be a service, application, or device. A service is identified by the protocol type and/or port number. An application is a computer program for a specific task or purpose. It provides additional semantics (e.g., dependencies between services) for matching traffic. A device is a managed entity that is connected to the network. The attributes that can identify a device include type (e.g., router, switch, pc) and operating system (e.g., Windows, Linux, or Android), as well as the device's owner;
 - * State: It refers to various states to which the network flow is associated. It can be either the TCP session state (e.g., new, established, related, invalid, or untracked), the session AAA state (e.g., authenticated but not authorized), or the access mode of the device (e.g., wireline, wireless, or cellular; these could be augmented with additional attributes, such as the type of VPN that is being used);
 - * Direction: the direction of the network flow.
- o Action: NSFs provide security functions by executing various Actions, which at least includes:
 - Ingress actions, such as pass, drop, mirroring, etc;

- Egress actions, 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. The functional profile or signature file defines the security capabilities for content security control and/or attack mitigation control; these will be described in sections 3.2 and 3.3, respectively. It is one of the key properties that determine the effectiveness of the NSF, and is mostly vendor-specific today. One goal of I2NSF is to standardize the form and functional interface of those security capabilities while supporting vendor-specific implementations of each.

The above ECA ruleset is very general and easily extensible, thus can avoid any potential constraints which could limit the implementation of the network security control capability.

3.2. Content Security

Content security is another category of security capabilities applied to application layer. Through detecting the contents carried over the traffic in application layer, these capabilities can realize various security functions, such as defending against intrusion, inspecting virus, filtering malicious URL or junk email, blocking illegal web access or malicious data retrieval.

Generally, each type of threat in the application layer has a set of unique characteristics, and requires handling with a set of specific methods. Thus, it can be thought of as a logically independent security capability. Since there are a large number of types of threats in the application layer, as well as new types of threats that occur quickly, there will be a large number of security capabilities. Therefore, some basic principles for security capability management and utilization need to be considered:

- o Flexibility: each security capability should be an independent function, with minimum overlap or dependency to other capabilities. This enables each security capability to be utilized and assembled together freely. More importantly, changes to one capability will not affect other capabilities;

- o High level of abstraction: this enables each capability to have a unified interface to make it programmable; this in turn provides a standardized ability to describe and report its processing results and corresponding statistics information. Furthermore, it facilitates the multi-vendor interoperability;
- o Scalability: The system must have the capability to scale up/down or scale in/out. Thus, it can meet various performance requirements derived from changeable network traffic or service requests. In addition, the security capability must support reporting statistics to the security controller to assist its decision on whether it needs to invoke scaling or not;
- o Automation: The system must have the ability to auto-discover, auto-negotiate, and auto-update security capabilities. These features are especially useful for the management of a large number of NSFs.

Based on the above principles, a set of abstract and vendor-neutral capabilities with standard interfaces is needed. The security controller can compare the requirements of clients to the set of capabilities that are currently available in order to choose which NSFs are needed to meet those requirements. Note that this choice is independent of vendor, and instead relies specifically on the capabilities (i.e., the description) of the functions provided. This also facilitates the customization of the functionality of the selected NSFs by setting the parameters of their interfaces. This category of security capability abstracts security as a black box that has selectable features compared with current network security control mechanisms.

Furthermore, when an unknown threat (e.g., zero-day exploits, unknown malware, and APTs) is reported by a network security device, new capabilities may be created, and/or existing capabilities may be updated (e.g., signature and algorithm), to correspond to the new functionality provided by the NSF to handle the threat. The new capabilities are provided from different vendors after their analysis of the new threats and subsequent installation of the functions required to report on (and possibly mitigate) the threat. New capabilities may be sent to and stored in a centralized repository, or stored separately in a local repository. In either case, a standard interface is needed during this automated update process.

3.3. Attack Mitigation

This category of security capabilities is used to detect and mitigate various types of network attacks. Today's common network attacks can be classified into the following sets, and each set further consists of a number of specific attacks:

- o DDoS attacks:

- Network layer DDoS attacks: Examples include SYN flood, UDP flood, ICMP flood, IP fragment flood, IPv6 Routing header attack, and IPv6 duplicate address detection attack;

- Application layer DDoS attacks: Examples include http flood, https flood, cache-bypass http floods, WordPress XML RPC floods, ssl DDoS.

- o Single-packet attack:

- Scanning and sniffing attacks: IP sweep, port scanning, etc

- malformed packet attacks: Ping of Death, Teardrop, etc

- special packet attacks: Oversized ICMP, Tracert, IP timestamp option packets, etc

Each type of network attack has its own network behaviors and packet/flow characteristics. Therefore, each type of attack needs a special security function, which is advertised as a capability, for detection and mitigation.

Overall, the implementation and management of this category of security capabilities of attack mitigation control is very similar to content security control. A standard interface, through which the security controller can choose and customize the given security capabilities according to specific requirements, is essential.

4. Information Model Design

4.1. Overall Structure

The I2NSF capability interface is in charge of controlling and monitoring the NSFs. This is done using the following approach:

- 1) User of the capability interface selects the set of capabilities required to meet the needs of the application;

- 2) A management entity uses the information model to match chosen capabilities to NSFs, independent of vendor;
- 3) A management entity takes the above information and creates or uses vendor-specific data models to install the NSFs identified by the chosen capabilities;
- 4) Control and monitoring can then begin.

Based on the analysis above, the information model should consist of at least four sections: capability, network security, content security and attack mitigation. This assumes that an external model, or set of models, is used to define the concept of an ECA Policy Rule and its components (e.g., Event, Condition, and Action objects).

Since Capabilities are determined by the management system, and are not inherent characteristics that differentiate objects, it is also assumed that an external model (or set of models) will define a generic metadata concept. Capabilities are then sub-classed from an appropriate class in the external metadata model.

The capability interface is used for advertising, creating, selecting and managing a set of specific security capabilities independent of the type and vendor of device that contains the NSF. That is, the user of the capability interface does not care whether the NSF is virtualized or hosted in a physical device, the vendor of the NSF, and which set of entities the NSF is communicating with (e.g., a firewall or an IPS). Instead, the user only cares about the set of capabilities that the NSF has, such as packet filtering or deep packet inspection. The overall structure is illustrated in the figure below:

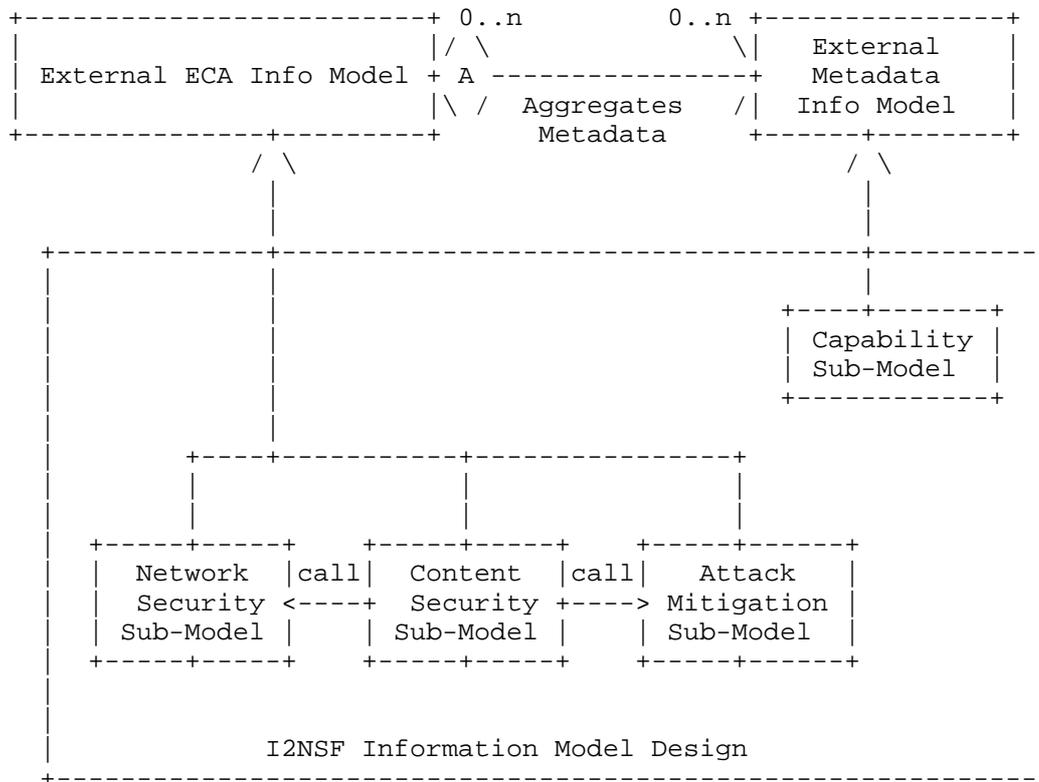


Figure 1. The Overall I2NSF Information Model Design

As illustrated in Figure 1, the network security function is the key. It usually runs as the first step to handle traffic (e.g., packet/flow detection and filtering, etc.) over the network layer. The framework portion of the information model ensures that each of the three domain sub-models (content security, network security, and attack mitigation) can function in collaboration or independently.

The content security and attack mitigation sub-models can be enforced on demand (i.e., once or recursively based on the results of network security function).

This draft defines the four sub-models inside the I2NSF information model shown in Figure 1. This model assumes that another, generic, information model for defining ECA policy rules exists outside of I2NSF. Hence, the Network Security, Content Security, and Attack Mitigation Sub-Models each extend the generic external ECA model to form security policy rules.

It also assumes that Capabilities are modeled as metadata, since a Capability is something that describes and/or prescribes functionality about an object, but is not an inherent part of that object. Hence, the Security Capability Sub-Model extends the generic external metadata model.

Both of these external models could, but do not have to, draw from the SUPA model [I-D.draft-ietf-sup-a-generic-policy-info-model].

The external ECA Information Model supplies at least a set of objects that represent a generic ECA Policy Rule, and a set of objects that represent Events, Conditions, and Actions that can be aggregated by the generic ECA Policy Rule. This enables I2NSF to reuse this generic model for different purposes.

It is assumed that the external ECA Information Model has the ability to aggregate metadata. Capabilities are then subclassed from an appropriate class in the external Metadata Information Model; this enables the ECA objects to use the existing aggregation between them and Metadata to add Metadata to appropriate ECA objects. Referring to Figure 1, this means that each of Network Security, Content Security, and Attack Mitigation Sub-Models can aggregate zero or more metadata objects to describe and/or prescribe their behavior.

Detailed descriptions of each portion of the information model are given in the following sections.

4.2. Information Sub-Model for Network Security Capabilities

The purpose of the Capability Framework Information Sub-Model is to define the concept of a Capability from an external metadata model, and enable Capabilities to be aggregated to appropriate objects in the Network Security, Content Security, and Attack Mitigation models.

4.3. Information Sub-Model for Network Security

The purpose of the Network Security Information Sub-Model is to define how network traffic is defined and determine if one or more network security features need to be applied to the traffic or not. Its basic structure is shown in the following figure:

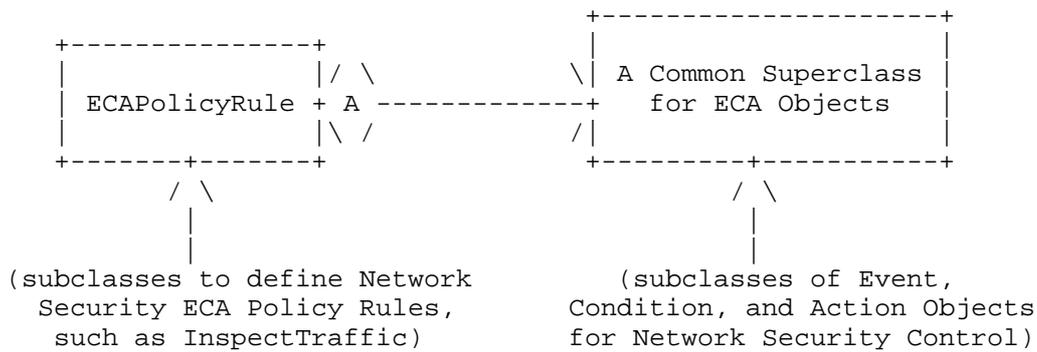


Figure 2. Network Security Information Sub-Model Overview

In the above figure, the ECAPolicyRule, along with the Event, Condition, and Action Objects, are defined in the external ECA Info Model. The Network Security Sub-Model extends both to define security-specific ECA policy rules, as well as Events, Conditions, and Actions.

An I2NSF Policy Rule is a special type of Policy Rule that is in event-condition-action (ECA) form. It consists of the Policy Rule, components of a Policy Rule (e.g., events, conditions, and actions), and optionally, metadata. It can be applied to both uni-directional and bi-directional traffic across the NSF.

Each rule is triggered by one or more events. If the set of events evaluates to true, then a set of conditions are evaluated and, if true, enable a set of actions to be executed.

An example of an I2NSF Policy Rule is, in pseudo-code:

```

IF <event-clause> is TRUE
  IF <condition-clause> is TRUE
    THEN execute <action-clause>
  END-IF
END-IF
    
```

In the above example, the Event, Condition, and Action portions of a Policy Rule are all ****Boolean Clauses****.

4.3.1. Network Security Policy Rule Extensions

Figure 3 shows a more detailed design of the ECA Policy Rule subclasses that are contained in the Network Security Information Sub-Model.

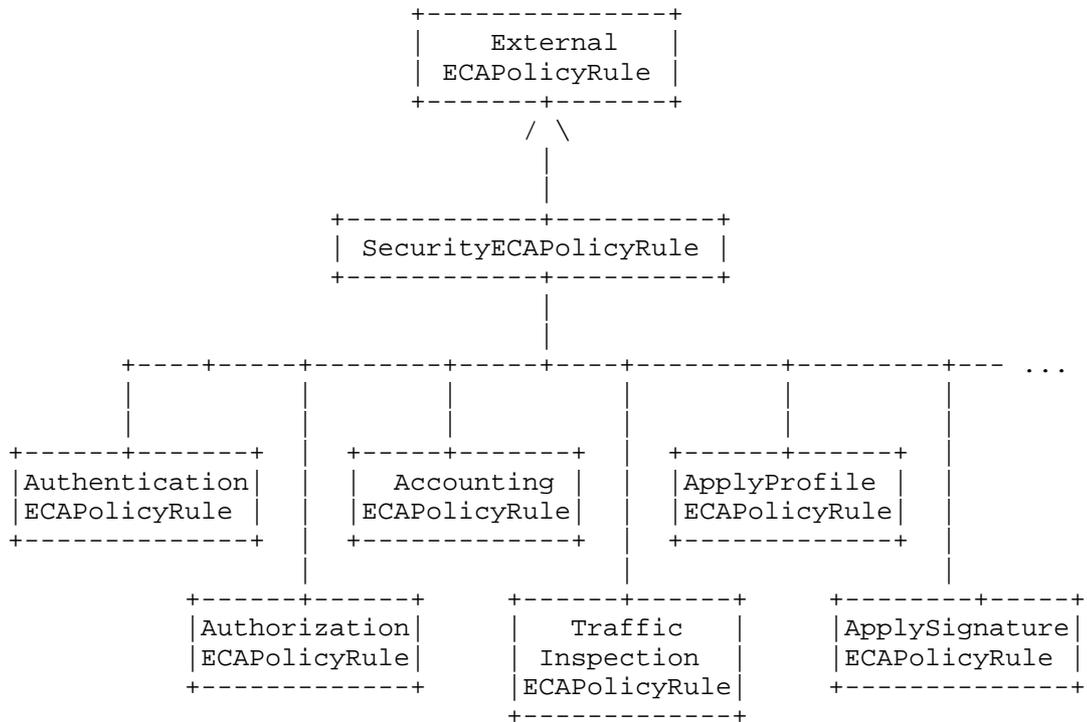


Figure 3. Network Security Info Sub-Model ECAPolicyRule Extensions

The SecurityECAPolicyRule is the top of the I2NSF ECA Policy Rule hierarchy. It inherits from the (external) generic ECA Policy Rule to define Security ECA Policy Rules. The SecurityECAPolicyRule contains all of the attributes, methods, and relationships defined in its superclass, and adds additional concepts that are required for Network Security (these will be defined in the next version of this draft). The six SecurityECAPolicyRule subclasses extend the SecurityECAPolicyRule class to represent six different types of Network Security ECA Policy Rules. It is assumed that the (external) generic ECAPolicyRule class defines basic information in the form of attributes, such as an unique object ID, as well as a description and other basic, but necessary, information.

It is assumed that the (external) generic ECA Policy Rule is abstract; the SecurityECAPolicyRule is also abstract. This enables data model optimizations to be made while making this information model detailed but flexible and extensible.

The SecurityECAPolicyRule defines network security policy as a container that aggregates Event, Condition, and Action objects, which are described in Section 4.4, 4.5, and 4.6, respectively. Events, Conditions, and Actions can be generic or security-specific. Section 4.6 defines the concept of default security Actions.

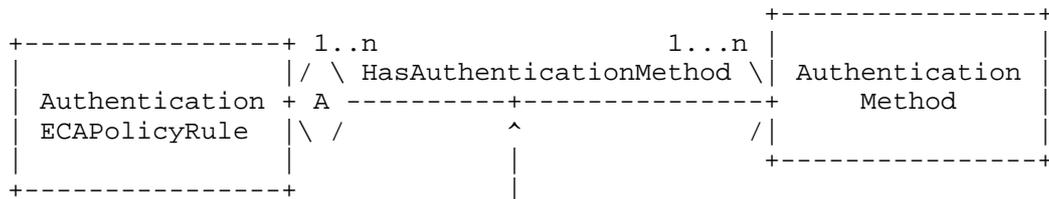
Brief class descriptions of these six ECA Policy Rules are provided in the following sub-sections. Note that there is a common pattern that defines how these ECAPolicyRules operate; this simplifies their implementation. All of these six ECA Policy Rules are concrete classes.

In addition, none of these six subclasses define attributes. This enables them to be viewed as simple object containers, and hence, applicable to a wide variety of content. It also means that the content of the function (e.g., how an entity is authenticated, what specific traffic is inspected, or which particular signature is applied) is defined solely by the set of events, conditions, and actions that are contained by the particular subclass. This enables the policy rule, with its aggregated set of events, conditions, and actions, to be treated as a reusable object.

4.3.1.1. AuthenticationECAPolicyRule Class Definition

The purpose of an AuthenticationECAPolicyRule is to define an ECA Policy Rule that can verify whether an entity has an attribute of a specific value.

This class does NOT define the authentication method used. This is because this would effectively "enclose" this information within the AuthenticationECAPolicyRule. This has two drawbacks. First, other entities that need to use information from the Authentication class(es) could not; they would have to associate with the AuthenticationECAPolicyRule class, and those other classes would not likely be interested in the AuthenticationECAPolicyRule. Second, the evolution of new authentication methods should be independent of the AuthenticationECAPolicyRule; this cannot happen if the Authentication class(es) are embedded in the AuthenticationECAPolicyRule. Hence, this document recommends the following design:



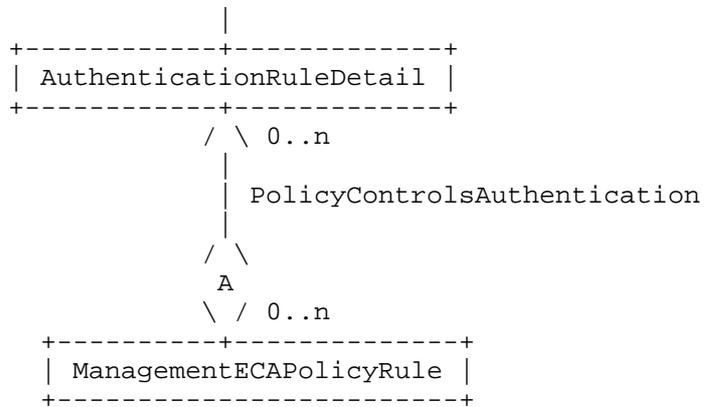


Figure 4. Modeling Authentication Mechanisms

This document only defines the AuthenticationECAPolicyRule; all other classes, and the aggregations, are defined in an external model. For completeness, descriptions of how the two aggregations are used are below.

Figure 4 defines an aggregation between the AuthenticationECAPolicyRule and an external AuthenticationMethod class (which is likely a superclass for different types of authentication mechanisms). This decouples the implementation of authentication mechanisms from how authentication mechanisms are used.

Since different AuthenticationECAPolicyRules can use different authentication mechanisms in different ways, the aggregation is realized as an association class. This enables the attributes and methods of the association class (i.e., AuthenticationRuleDetail) to be used to define how a given AuthenticationMethod is used by a particular AuthenticationECAPolicyRule.

Similarly, the PolicyControlsAuthentication aggregation defines policies to control the configuration of the AuthenticationRuleDetail association class. This enables the entire authentication process to be managed by ECAPolicyRules.

Note: a data model MAY choose to collapse this design into a more efficient implementation. For example, a data model could define two attributes for the AuthenticationECAPolicyRule class, called (for example) authenticationMethodCurrent and authenticationMethodSupported, to represent the HasAuthenticationMethod aggregation and its association class. The

former is a string attribute that defines the current authentication method used by this AuthenticationECAPolicyRule, while the latter defines a set of authentication methods, in the form of an authentication capability, which this AuthenticationECAPolicyRule can advertise.

4.3.1.2. AuthorizationECAPolicyRuleClass Definition

The purpose of an AuthorizationECAPolicyRule is to define an ECA Policy Rule that can determine whether access to a resource should be given and, if so, what permissions should be granted to the entity that is accessing the resource.

This class does NOT define the authorization method(s) used. This is because this would effectively "enclose" this information within the AuthorizationECAPolicyRule. This has two drawbacks. First, other entities that need to use information from the Authorization class(es) could not; they would have to associate with the AuthorizationECAPolicyRule class, and those other classes would not likely be interested in the AuthorizationECAPolicyRule. Second, the evolution of new authorization methods should be independent of the AuthorizationECAPolicyRule; this cannot happen if the Authorization class(es) are embedded in the AuthorizationECAPolicyRule. Hence, this document recommends the following design:

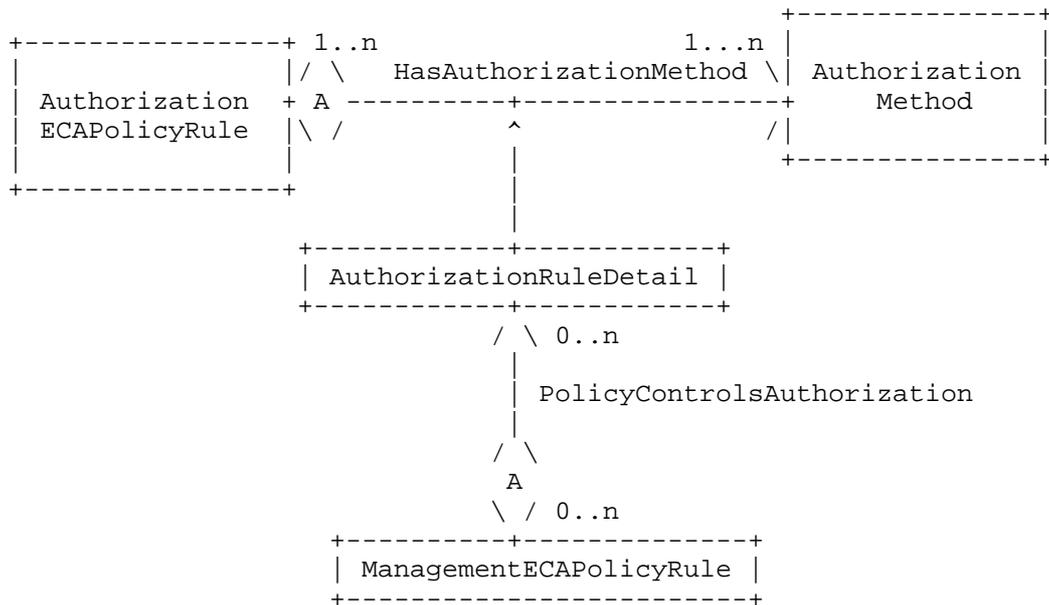


Figure 5. Modeling Authorization Mechanisms

This document only defines the AuthorizationECAPolicyRule; all other classes, and the aggregations, are defined in an external model. For completeness, descriptions of how the two aggregations are used are below.

Figure 5 defines an aggregation between the AuthorizationECAPolicyRule and an external AuthorizationMethod class (which is likely a superclass for different types of authorization mechanisms). This decouples the implementation of authorization mechanisms from how authorization mechanisms are used.

Since different AuthorizationECAPolicyRules can use different authorization mechanisms in different ways, the aggregation is realized as an association class. This enables the attributes and methods of the association class (i.e., AuthorizationRuleDetail) to be used to define how a given AuthorizationMethod is used by a particular AuthorizationECAPolicyRule.

Similarly, the PolicyControlsAuthorization aggregation defines policies to control the configuration of the AuthorizationRuleDetail association class. This enables the entire authorization process to be managed by ECAPolicyRules.

Note: a data model MAY choose to collapse this design into a more efficient implementation. For example, a data model could define two attributes for the AuthorizationECAPolicyRule class, called (for example) authorizationMethodCurrent and authorizationMethodSupported, to represent the HasAuthorizationMethod aggregation and its association class. The former is a string attribute that defines the current authorization method used by this AuthorizationECAPolicyRule, while the latter defines a set of authorization methods, in the form of an authorization capability, which this AuthorizationECAPolicyRule can advertise.

4.3.1.3. AccountingECAPolicyRuleClass Definition

The purpose of an AccountingECAPolicyRule is to define an ECA Policy Rule that can determine which information to collect, and how to collect that information, from which set of resources for the purpose of trend analysis, auditing, billing, or cost allocation [RFC2975] [RFC3539].

This class does NOT define the accounting method(s) used. This is because this would effectively "enclose" this information within the AccountingECAPolicyRule. This has two drawbacks. First, other entities that need to use information from the Accounting class(es) could not; they would have to associate with the AccountingECAPolicyRule class, and those other classes would not likely be interested in the AccountingECAPolicyRule. Second, the evolution of new accounting methods should be independent of the AccountingECAPolicyRule; this cannot happen if the Accounting class(es) are embedded in the AccountingECAPolicyRule. Hence, this document recommends the following design:

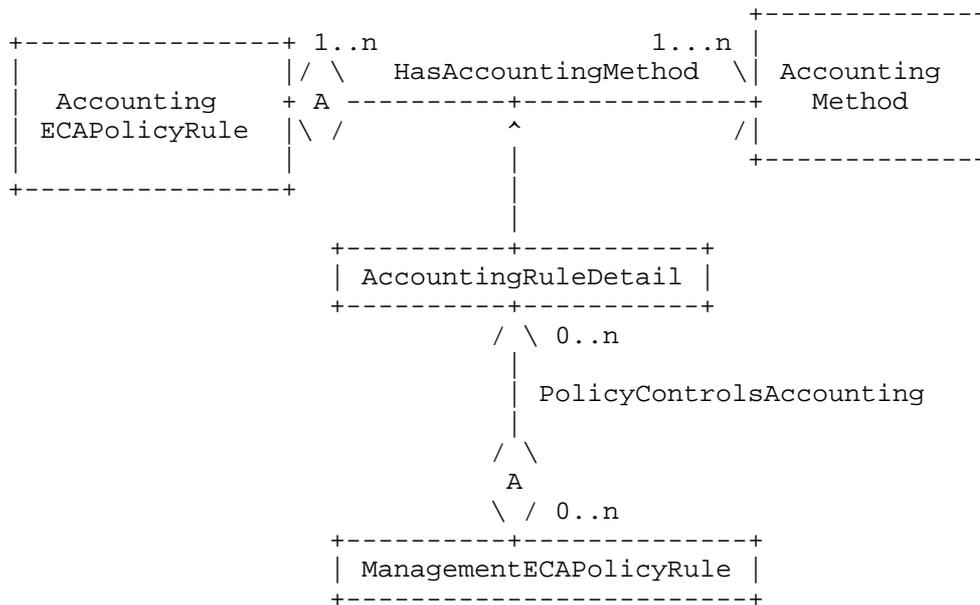


Figure 6. Modeling Accounting Mechanisms

This document only defines the AccountingECAPolicyRule; all other classes, and the aggregations, are defined in an external model. For completeness, descriptions of how the two aggregations are used are below.

Figure 6 defines an aggregation between the AccountingECAPolicyRule and an external AccountingMethod class (which is likely a superclass for different types of accounting mechanisms). This decouples the implementation of accounting mechanisms from how accounting mechanisms are used.

Since different AccountingECAPolicyRules can use different accounting mechanisms in different ways, the aggregation is realized as an association class. This enables the attributes and methods of the association class (i.e., AccountingRuleDetail) to be used to define how a given AccountingMethod is used by a particular AccountingECAPolicyRule.

Similarly, the PolicyControlsAccounting aggregation defines policies to control the configuration of the AccountingRuleDetail association class. This enables the entire accounting process to be managed by ECAPolicyRules.

Note: a data model MAY choose to collapse this design into a more efficient implementation. For example, a data model could define two attributes for the AccountingECAPolicyRule class, called (for example) accountingMethodCurrent and accountingMethodSupported, to represent the HasAccountingMethod aggregation and its association class. The former is a string attribute that defines the current accounting method used by this AccountingECAPolicyRule, while the latter defines a set of accounting methods, in the form of an authorization capability, which this AccountingECAPolicyRule can advertise.

4.3.1.4. TrafficInspectionECAPolicyRuleClass Definition

The purpose of a TrafficInspectionECAPolicyRule is to define an ECA Policy Rule that, based on a given context, can determine which traffic to examine on which devices, which information to collect from those devices, and how to collect that information.

This class does NOT define the traffic inspection method(s) used. This is because this would effectively "enclose" this information within the TrafficInspectionECAPolicyRule. This has two drawbacks. First, other entities that need to use information from the TrafficInspection class(es) could not; they would have to associate with the TrafficInspectionECAPolicyRule class, and those other classes would not likely be interested in the TrafficInspectionECAPolicyRule. Second, the evolution of new traffic inspection methods should be independent of the TrafficInspectionECAPolicyRule; this cannot happen if the TrafficInspection class(es) are embedded in the TrafficInspectionECAPolicyRule. Hence, this document recommends the following design:

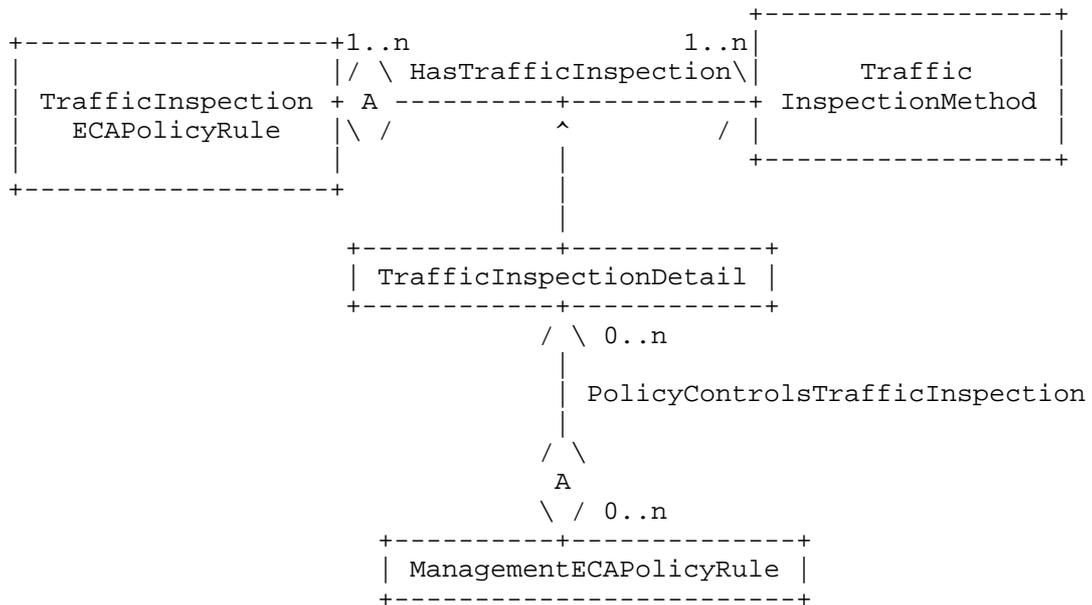


Figure 7. Modeling Traffic Inspection Mechanisms

This document only defines the TrafficInspectionECAPolicyRule; all other classes, and the aggregations, are defined in an external model. For completeness, descriptions of how the two aggregations are used are below.

Figure 7 defines an aggregation between the TrafficInspectionECAPolicyRule and an external TrafficInspection class (which is likely a superclass for different types of traffic inspection mechanisms). This decouples the implementation of traffic inspection mechanisms from how traffic inspection mechanisms are used.

Since different TrafficInspectionECAPolicyRules can use different traffic inspection mechanisms in different ways, the aggregation is realized as an association class. This enables the attributes and methods of the association class (i.e., TrafficInspectionDetail) to be used to define how a given TrafficInspectionMethod is used by a particular TrafficInspectionECAPolicyRule.

Similarly, the PolicyControlsTrafficInspection aggregation defines policies to control the configuration of the TrafficInspectionDetail association class. This enables the entire traffic inspection process to be managed by ECAPolicyRules.

Note: a data model MAY choose to collapse this design into a more efficient implementation. For example, a data model could define two attributes for the TrafficInspectionECAPolicyRule class, called (for example) trafficInspectionMethodCurrent and trafficInspectionMethodSupported, to represent the HasTrafficInspectionMethod aggregation and its association class. The former is a string attribute that defines the current traffic inspection method used by this TrafficInspectionECAPolicyRule, while the latter defines a set of traffic inspection methods, in the form of a traffic inspection capability, which this TrafficInspectionECAPolicyRule can advertise.

4.3.1.5. ApplyProfileECAPolicyRuleClass Definition

The purpose of an ApplyProfileECAPolicyRule is to define an ECA Policy Rule that, based on a given context, can apply a particular profile to specific traffic. The profile defines the security capabilities for content security control and/or attack mitigation control; these will be described in sections 4.4 and 4.5, respectively.

This class does NOT define the set of Profiles used. This is because this would effectively "enclose" this information within the ApplyProfileECAPolicyRule. This has two drawbacks. First, other entities that need to use information from the Profile class(es) could not; they would have to associate with the ApplyProfileECAPolicyRule class, and those other classes would not likely be interested in the ApplyProfileECAPolicyRule. Second, the evolution of new Profile classes should be independent of the ApplyProfileECAPolicyRule; this cannot happen if the Profile class(es) are embedded in the ApplyProfileECAPolicyRule. Hence, this document recommends the following design:

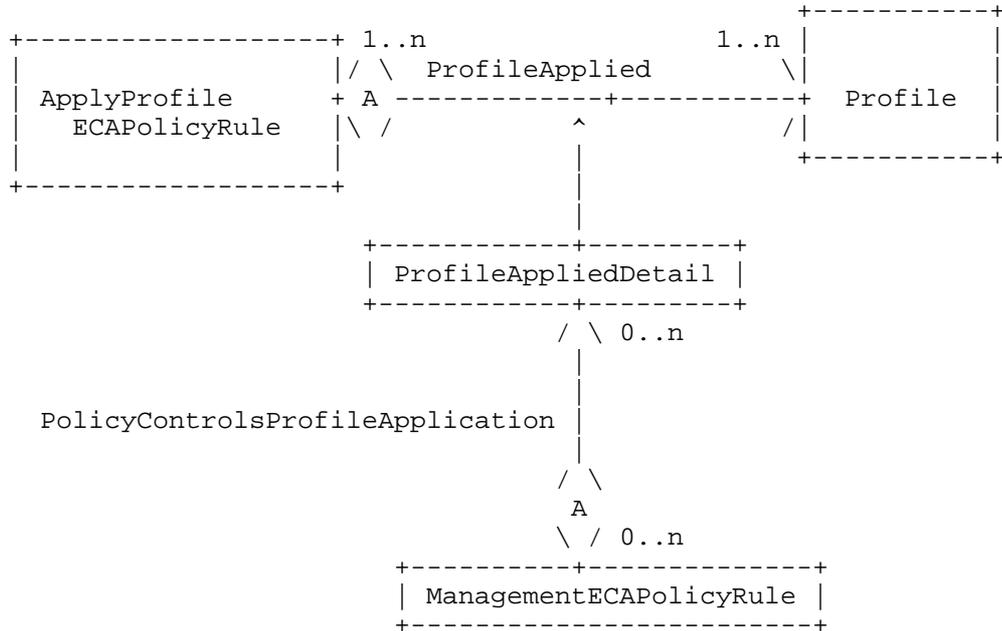


Figure 8. Modeling Profile ApplicationMechanisms

This document only defines the `ApplyProfileECAPolicyRule`; all other classes, and the aggregations, are defined in an external model. For completeness, descriptions of how the two aggregations are used are below.

Figure 8 defines an aggregation between the `ApplyProfileECAPolicyRule` and an external `Profile` class (which is likely a superclass for different types of Profiles). This decouples the implementation of Profiles from how Profiles are used.

Since different `ApplyProfileECAPolicyRules` can use different Profiles in different ways, the aggregation is realized as an association class. This enables the attributes and methods of the association class (i.e., `ProfileAppliedDetail`) to be used to define how a given Profile is used by a particular `ApplyProfileECAPolicyRule`.

Similarly, the `PolicyControlsProfileApplication` aggregation defines policies to control the configuration of the `ProfileAppliedDetail` association class. This enables the application of Profiles to be managed by `ECAPolicyRules`.

Note: a data model MAY choose to collapse this design into a more efficient implementation. For example, a data model could define two attributes for the `ApplyProfileECAPolicyRule` class, called (for example) `profileAppliedCurrent` and `profileAppliedSupported`, to represent the `ProfileApplied` aggregation and its association class. The former is a string attribute that defines the current Profile used by this `ApplyProfileECAPolicyRule`, while the latter defines a set of Profiles, in the form of a Profile capability, which this `ApplyProfileECAPolicyRule` can advertise.

4.3.1.6. `ApplySignatureECAPolicyRule` Class Definition

The purpose of an `ApplySignatureECAPolicyRule` is to define an ECA Policy Rule that, based on a given context, can determine which Signature object (e.g., an anti-virus file, or a URL filtering file, or a script) to apply to which traffic. The Signature object defines the security capabilities for content security control and/or attack mitigation control; these will be described in sections 4.4 and 4.5, respectively.

This class does NOT define the set of Signature objects used. This is because this would effectively "enclose" this information within the `ApplySignatureECAPolicyRule`. This has two drawbacks. First, other entities that need to use information from the Signature object class(es) could not; they would have to associate with the `ApplySignatureECAPolicyRule` class, and those other classes would not likely be interested in the `ApplySignatureECAPolicyRule`. Second, the evolution of new Signature object classes should be independent of the `ApplySignatureECAPolicyRule`; this cannot happen if the Signature object class(es) are embedded in the `ApplySignatureECAPolicyRule`. Hence, this document recommends the following design:

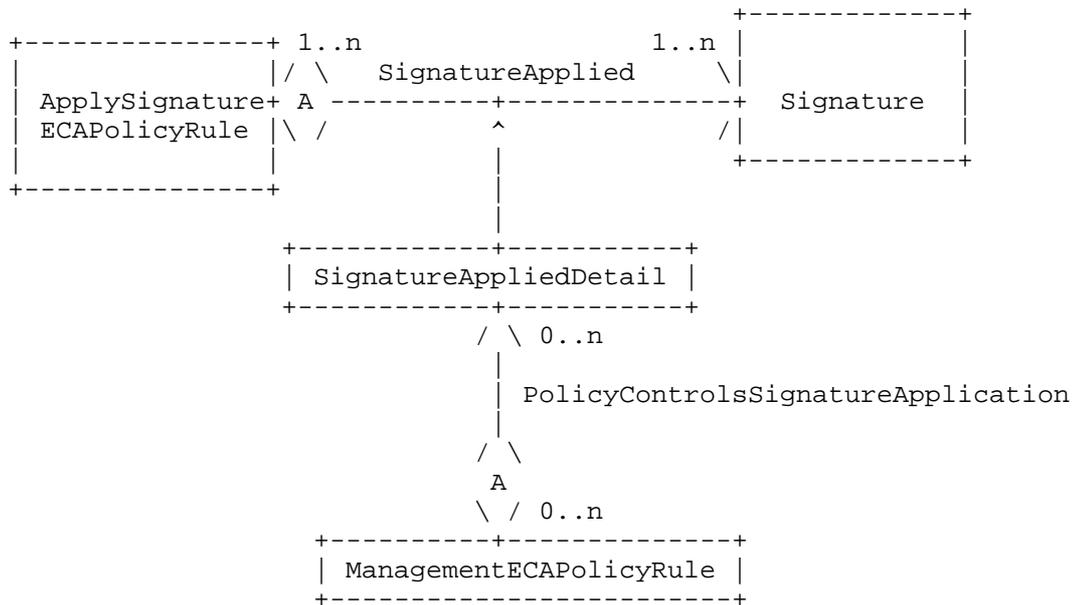


Figure 9. Modeling Sginature Application Mechanisms

This document only defines the `ApplySignatureECAPolicyRule`; all other classes, and the aggregations, are defined in an external model. For completeness, descriptions of how the two aggregations are used are below.

Figure 9 defines an aggregation between the `ApplySignatureECAPolicyRule` and an external `Signature` object class (which is likely a superclass for different types of `Signature` objects). This decouples the implementation of signature objects from how `Signature` objects are used.

Since different `ApplySignatureECAPolicyRules` can use different `Signature` objects in different ways, the aggregation is realized as an association class. This enables the attributes and methods of the association class (i.e., `SignatureAppliedDetail`) to be used to define how a given `Signature` object is used by a particular `ApplySignatureECAPolicyRule`.

Similarly, the `PolicyControlsSignatureApplication` aggregation defines policies to control the configuration of the

SignatureAppliedDetail association class. This enables the application of the Signature object to be managed by policy.

Note: a data model MAY choose to collapse this design into a more efficient implementation. For example, a data model could define two attributes for the ApplySignatureECAPolicyRule class, called (for example) signature signatureAppliedCurrent and signatureAppliedSupported, to represent the SignatureApplied aggregation and its association class. The former is a string attribute that defines the current Signature object used by this ApplySignatureECAPolicyRule, while the latter defines a set of Signature objects, in the form of a Signature capability, which this ApplySignatureECAPolicyRule can advertise.

4.3.2. Network Security Policy Rule Operation

Network security policy consists of a number of more granular ECA Policy Rules formed from the information model described above. In simpler cases, where the Event and Condition clauses remain unchanged, then network security control may be performed by calling additional network security actions. Network security policy examines and performs basic processing of the traffic as follows:

1. For a given SecurityECAPolicyRule (which can be generic or specific to security, such as those in Figure 3), the NSF evaluates the Event clause. It may use security Event objects to do all or part of this evaluation, which are defined in section 4.3.3. If the Event clause evaluates to TRUE, then the Condition clause of this SecurityECAPolicyRule is evaluated; otherwise, execution of this SecurityECAPolicyRule is stopped, and the next SecurityECAPolicyRule (if one exists) is evaluated;
2. The Condition clause is then evaluated. It may use security Condition objects to do all or part of this evaluation, which are defined in section 4.3.4. If the Condition clause evaluates to TRUE, then the set of Actions in this SecurityECAPolicyRule MUST be executed. This is defined as "matching" the SecurityECAPolicyRule; otherwise, execution of this SecurityECAPolicyRule is stopped, and the next SecurityECAPolicyRule (if one exists) is evaluated;
3. If none of the SecurityECAPolicyRules are matched, then the NSF denies the traffic by default;

4. If the traffic matches a rule, the NSF performs the defined Actions on the traffic. It may use security Action objects to do all or part of this execution, which are defined in section 4.3.5. If the action is "deny", the NSF blocks the traffic. If the basic action is permit or mirror, the NSF firstly performs that function, and then checks whether certain other security capabilities are referenced in the rule. If yes, go to step 5. If no, the traffic is permitted;
5. If other security capabilities (e.g., Anti-virus or IPS) are referenced in the SecurityECAPolicyRule, and the Action defined in the rule is permit or mirror, the NSF performs the referenced security capabilities.

Metadata attached to the SecurityECAPolicyRule MAY be used to control how the SecurityECAPolicyRule is evaluated. This is called a Policy Rule Evaluation Strategy. For example, one strategy is to match and execute the first SecurityECAPolicyRule, and then exit without executing any other SecurityECAPolicyRules (even if they matched). In contrast, a second strategy is to first collect all SecurityECAPolicyRules that matched, and then execute them according to a pre-defined order (e.g., the priority of each SecurityECAPolicyRule).

One policy or rule can be applied multiple times to different managed objects (e.g., links, devices, networks, VPNS). This not only guarantees consistent policy enforcement, but also decreases the configuration workload.

4.3.3. Network Security Event Sub-Model

Figure 10 shows a more detailed design of the Event subclasses that are contained in the Network Security Information Sub-Model.

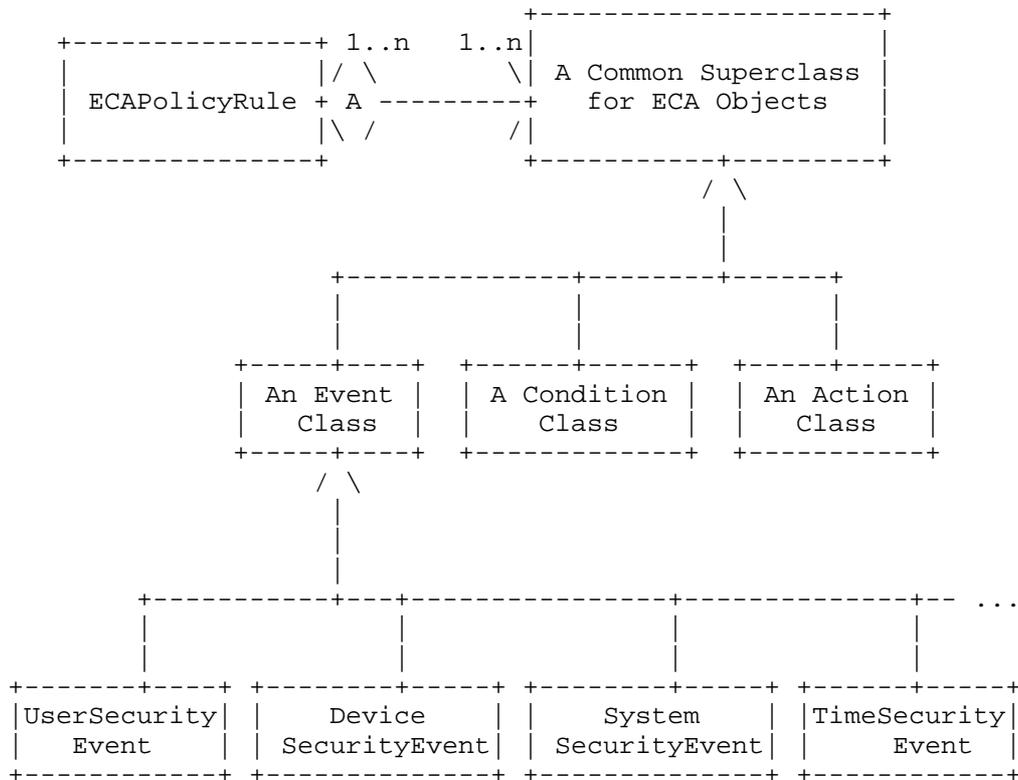


Figure 10. Network Security Info Sub-Model Event Class Extensions

The four Event classes shown in Figure 10 extend the (external) generic Event class to represent Events that are of interest to Network Security. It is assumed that the (external) generic Event class defines basic Event information in the form of attributes, such as a unique event ID, a description, as well as the date and time that the event occurred.

The following are assumptions that define the functionality of the generic Event class. If desired, these could be defined as attributes in a SecurityEvent class (which would be a subclass of the generic Event class, and a superclass of the four Event classes shown in Figure 10). However, this makes it harder to use any generic Event model with the I2NSF events. Assumptions are:

- The generic Event class is abstract
- All four SecurityEvent subclasses are concrete
- The generic Event class uses the composite pattern, so individual Events as well as hierarchies of Events are available (the four subclasses in Figure 10 would be subclasses of the Atomic Event)
- The generic Event class has a mechanism to uniquely identify the source of the Event
- The generic Event class has a mechanism to separate header information from its payload
- The generic Event class has a mechanism to attach zero or more metadata objects to it

Brief class descriptions are provided in the following sub-sections.

4.3.3.1. UserSecurityEvent Class Description

The purpose of this class is to represent Events that are initiated by a user, such as logon and logoff Events. Information in this Event may be used as part of a test to determine if the Condition clause in this ECA Policy Rule should be evaluated or not. Examples include user identification data and the type of connection used by the user.

The UserSecurityEvent class defines the following attributes:

4.3.3.1.1. The usrSecEventContent Attribute

This is a mandatory string that contains the content of the UserSecurityEvent. The format of the content is specified in the usrSecEventFormat class attribute, and the type of Event is defined in the usrSecEventType class attribute. An example of the usrSecEventContent attribute is the string "hrAdmin", with the usrSecEventFormat set to 1 (GUID) and the usrSecEventType attribute set to 5 (new logon).

4.3.3.1.2. The usrSecEventFormat Attribute

This is a mandatory non-negative enumerated integer, which is used to specify the data type of the usrSecEventContent attribute. The content is specified in the usrSecEventContent class attribute, and the type of Event is defined in the usrSecEventType class attribute. An example of the usrSecEventContent attribute is the string "hrAdmin", with the usrSecEventFormat attribute set to 1 (GUID) and the usrSecEventType attribute set to 5 (new logon). Values include:

- 0: unknown
- 1: GUID (Generic Unique Identifier)
- 2: UUID (Universal Unique Identifier)
- 3: URI (Uniform Resource Identifier)
- 4: FQDN (Fully Qualified Domain Name)
- 5: FQPN (Fully Qualified Path Name)

4.3.3.1.3. The usrSecEventType Attribute

This is a mandatory non-negative enumerated integer, which is used to specify the type of Event that involves this user. The content and format are specified in the `usrSecEventContent` and `usrSecEventFormat` class attributes, respectively. An example of the `usrSecEventContent` attribute is the string "hrAdmin", with the `usrSecEventFormat` attribute set to 1 (GUID) and the `usrSecEventType` attribute set to 5 (new logon). Values include:

- 0: unknown
- 1: new user created
- 2: new user group created
- 3: user deleted
- 4: user group deleted
- 5: user logon
- 6: user logoff
- 7: user access request
- 8: user access granted
- 9: user access violation

4.3.3.2. DeviceSecurityEvent Class Description

The purpose of a `DeviceSecurityEvent` is to represent Events that provide information from the Device that are important to I2NSF Security. Information in this Event may be used as part of a test to determine if the Condition clause in this ECA Policy Rule should be evaluated or not. Examples include alarms and various device statistics (e.g., a type of threshold that was exceeded), which may signal the need for further action.

The `DeviceSecurityEvent` class defines the following attributes:

4.3.3.2.1. The devSecEventContent Attribute

This is a mandatory string that contains the content of the `DeviceSecurityEvent`. The format of the content is specified in the `devSecEventFormat` class attribute, and the type of Event is defined

in the devSecEventType class attribute. An example of the devSecEventContent attribute is "alarm", with the devSecEventFormat attribute set to 1 (GUID), the devSecEventType attribute set to 5 (new logon).

4.3.3.2.2. The devSecEventFormat Attribute

This is a mandatory non-negative enumerated integer, which is used to specify the data type of the devSecEventContent attribute. Values include:

- 0: unknown
- 1: GUID (Generic Unique Identifier)
- 2: UUID (Universal Unique Identifier)
- 3: URI (Uniform Resource Identifier)
- 4: FQDN (Fully Qualified Domain Name)
- 5: FQPN (Fully Qualified Path Name)

4.3.3.2.3. The devSecEventType Attribute

This is a mandatory non-negative enumerated integer, which is used to specify the type of Event that was generated by this device. Values include:

- 0: unknown
- 1: communications alarm
- 2: quality of service alarm
- 3: processing error alarm
- 4: equipment error alarm
- 5: environmental error alarm

Values 1-5 are defined in X.733. Additional types of errors may also be defined.

4.3.3.2.4. The devSecEventTypeInfo[0..n] Attribute

This is an optional array of strings, which is used to provide additional information describing the specifics of the Event generated by this Device. For example, this attribute could contain probable cause information in the first array, trend information in the second array, proposed repair actions in the third array, and additional information in the fourth array.

4.3.3.2.5. The devSecEventTypeSeverity Attribute

This is a mandatory non-negative enumerated integer, which is used to specify the perceived severity of the Event generated by this Device. Values include:

- 0: unknown
- 1: cleared
- 2: indeterminate
- 3: critical
- 4: major
- 5: minor
- 6: warning

Values 1-6 are from X.733.

4.3.3.3. SystemSecurityEvent Class Description

The purpose of a SystemSecurityEvent is to represent Events that are detected by the management system, instead of Events that are generated by a user or a device. Information in this Event may be used as part of a test to determine if the Condition clause in this ECA Policy Rule should be evaluated or not. Examples include an event issued by an analytics system that warns against a particular pattern of unknown user accesses, or an Event issued by a management system that represents a set of correlated and/or filtered Events.

The SystemSecurityEvent class defines the following attributes:

4.3.3.3.1. The sysSecEventContent Attribute

This is a mandatory string that contains the content of the SystemSecurityEvent. The format of the content is specified in the sysSecEventFormat class attribute, and the type of Event is defined in the sysSecEventType class attribute. An example of the sysSecEventContent attribute is the string "sysadmin3", with the sysSecEventFormat attribute set to 1 (GUID), and the sysSecEventType attribute set to 2 (audit log cleared).

4.3.3.3.2. The sysSecEventFormat Attribute

This is a mandatory non-negative enumerated integer, which is used to specify the data type of the sysSecEventContent attribute. Values include:

- 0: unknown
- 1: GUID (Generic Unique Identifier)
- 2: UUID (Universal Unique Identifier)
- 3: URI (Uniform Resource Identifier)
- 4: FQDN (Fully Qualified Domain Name)
- 5: FQPN (Fully Qualified Path Name)

4.3.3.3.3. The sysSecEventType Attribute

This is a mandatory non-negative enumerated integer, which is used to specify the type of Event that involves this device. Values include:

- 0: unknown
- 1: audit log written to
- 2: audit log cleared
- 3: policy created
- 4: policy edited
- 5: policy deleted
- 6: policy executed

4.3.3.4. TimeSecurityEvent Class Description

The purpose of a TimeSecurityEvent is to represent Events that are temporal in nature (e.g., the start or end of a period of time). Time events signify an individual occurrence, or a time period, in which a significant event happened. Information in this Event may be used as part of a test to determine if the Condition clause in this ECA Policy Rule should be evaluated or not. Examples include issuing an Event at a specific time to indicate that a particular resource should not be accessed, or that different authentication and authorization mechanisms should now be used (e.g., because it is now past regular business hours).

The TimeSecurityEvent class defines the following attributes:

4.3.3.4.1. The timeSecEventPeriodBegin Attribute

This is a mandatory DateTime attribute, and represents the beginning of a time period. It has a value that has a date and/or a time component (as in the Java or Python libraries).

4.3.3.4.2. The timeSecEventPeriodEnd Attribute

This is a mandatory DateTime attribute, and represents the end of a time period. It has a value that has a date and/or a time component (as in the Java or Python libraries). If this is a single Event occurrence, and not a time period when the Event can occur, then the timeSecEventPeriodEnd attribute may be ignored.

4.3.3.4.3. The timeSecEventTimeZone Attribute

This is a mandatory string attribute, and defines the time zone that this Event occurred in using the format specified in ISO8601.

4.3.4. Network Security Condition Sub-Model

Figure 11 shows a more detailed design of the Condition subclasses that are contained in the Network Security Information Sub-Model.

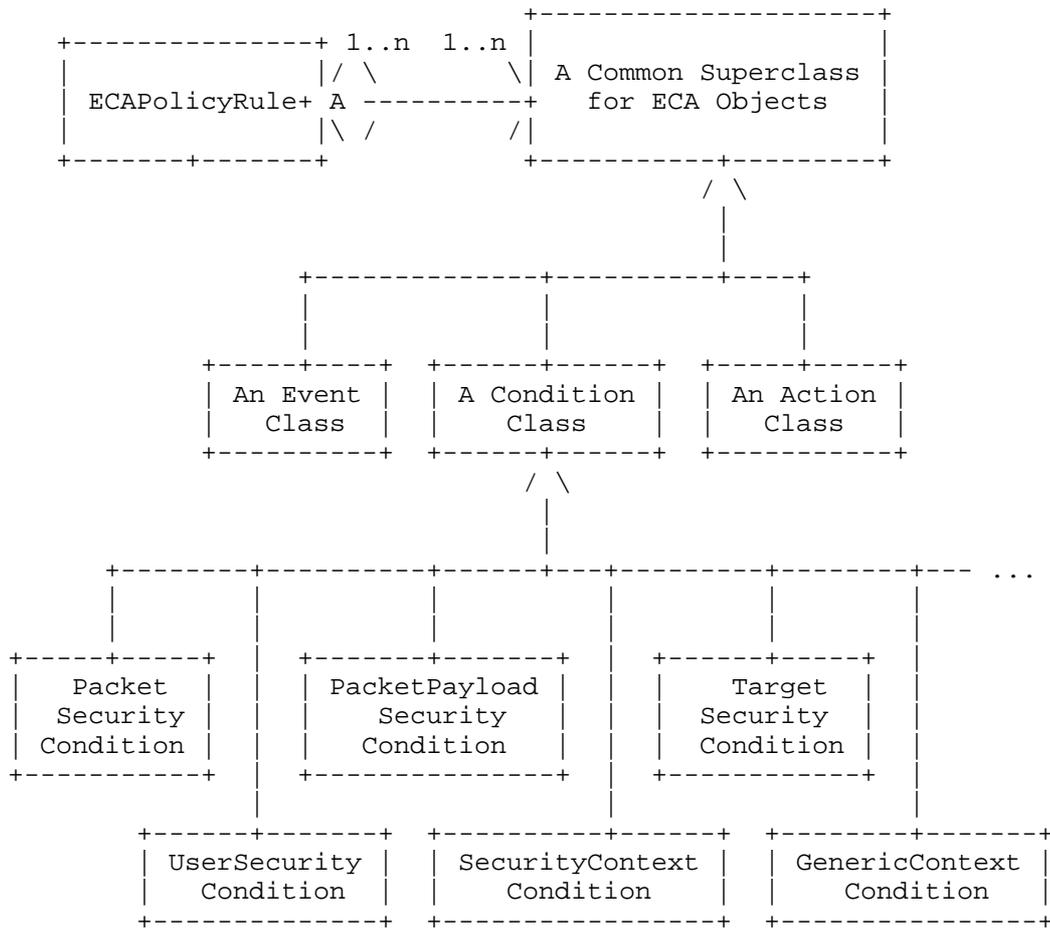


Figure 11. Network Security Info Sub-Model Condition Class Extensions

The six Condition classes shown in Figure 11 extend the (external) generic Condition class to represent Conditions that are of interest to Network Security. It is assumed that the (external) generic Condition class is abstract, so that data model optimizations may be defined. It is also assumed that the generic Condition class defines basic Condition information in the form of attributes, such as a unique object ID, a description, as well as a mechanism to attach zero or more metadata objects to it. While this could be defined as attributes in a SecurityCondition class (which would be a subclass

of the generic Condition class, and a superclass of the six Condition classes shown in Figure 11), this makes it harder to use any generic Condition model with the I2NSF conditions.

Brief class descriptions are provided in the following sub-sections.

4.3.4.1. PacketSecurityCondition

The purpose of this Class is to represent packet header information that can be used as part of a test to determine if the set of Policy Actions in this ECA Policy Rule should be executed or not. This class is abstract, and serves as the superclass of more detailed conditions that involve different types of packet formats. Its subclasses are shown in Figure 12, and are defined in the following sections.

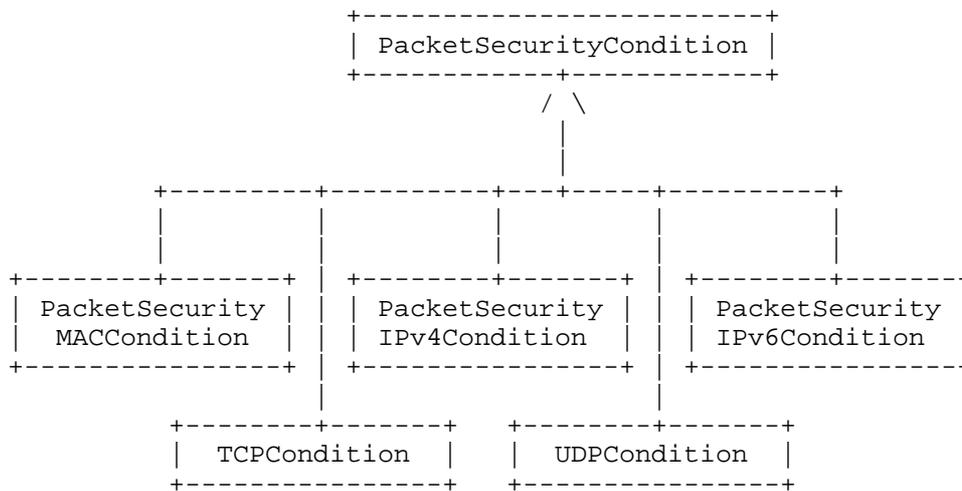


Figure 12. Network Security Info Sub-Model PacketSecurityCondition Class Extensions

4.3.4.1.1. PacketSecurityMACCondition

The purpose of this Class is to represent packet MAC packet header information that can be used as part of a test to determine if the set of Policy Actions in this ECA Policy Rule should be executed or not. This class is concrete, and defines the following attributes:

4.3.4.1.1.1. The pktSecCondMACDest Attribute

This is a mandatory string attribute, and defines the MAC destination address (6 octets long).

4.3.4.1.1.2. The pktSecCondMACSrc Attribute

This is a mandatory string attribute, and defines the MAC source address (6 octets long).

4.3.4.1.1.3. The pktSecCondMAC8021Q Attribute

This is an optional string attribute, and defines the 802.1Q tag value (2 octets long). This defines VLAN membership and 802.1p priority values.

4.3.4.1.1.4. The pktSecCondMACEtherType Attribute

This is a mandatory string attribute, and defines the EtherType field (2 octets long). Values up to and including 1500 indicate the size of the payload in octets; values of 1536 and above define which protocol is encapsulated in the payload of the frame.

4.3.4.1.1.5. The pktSecCondMACTCI Attribute

This is an optional string attribute, and defines the Tag Control Information. This consists of a 3 bit user priority field, a drop eligible indicator (1 bit), and a VLAN identifier (12 bits).

4.3.4.1.2. PacketSecurityIPv4Condition

The purpose of this Class is to represent packet IPv4 packet header information that can be used as part of a test to determine if the set of Policy Actions in this ECA Policy Rule should be executed or not. This class is concrete, and defines the following attributes:

4.3.4.1.2.1. The pktSecCondIPv4SrcAddr Attribute

This is a mandatory string attribute, and defines the IPv4 Source Address (32 bits).

4.3.4.1.2.2. The pktSecCondIPv4DestAddr Attribute

This is a mandatory string attribute, and defines the IPv4 Destination Address (32 bits).

4.3.4.1.2.3. The pktSecCondIPv4ProtocolUsed Attribute

This is a mandatory string attribute, and defines the protocol used in the data portion of the IP datagram (8 bits).

4.3.4.1.2.4. The pktSecCondIPv4DSCP Attribute

This is a mandatory string attribute, and defines the Differentiated Services Code Point field (6 bits).

4.3.4.1.2.5. The pktSecCondIPv4ECN Attribute

This is an optional string attribute, and defines the Explicit Congestion Notification field (2 bits).

4.3.4.1.2.6. The pktSecCondIPv4TotalLength Attribute

This is a mandatory string attribute, and defines the total length of the packet (including header and data) in bytes (16 bits).

4.3.4.1.2.7. The pktSecCondIPv4TTL Attribute

This is a mandatory string attribute, and defines the Time To Live in seconds (8 bits).

4.3.4.1.3. PacketSecurityIPv6Condition

The purpose of this Class is to represent packet IPv6 packet header information that can be used as part of a test to determine if the set of Policy Actions in this ECA Policy Rule should be executed or not. This class is concrete, and defines the following attributes:

4.3.4.1.3.1. The pktSecCondIPv6SrcAddr Attribute

This is a mandatory string attribute, and defines the IPv6 Source Address (128 bits).

4.3.4.1.3.2. The pktSecCondIPv6DestAddr Attribute

This is a mandatory string attribute, and defines the IPv6 Destination Address (128 bits).

4.3.4.1.3.3. The pktSecCondIPv6DSCP Attribute

This is a mandatory string attribute, and defines the Differentiated Services Code Point field (6 bits). It consists of the six most significant bits of the Traffic Class field in the IPv6 header.

4.3.4.1.3.4. The pktSecCondIPv6ECN Attribute

This is a mandatory string attribute, and defines the Explicit Congestion Notification field (2 bits). It consists of the two least significant bits of the Traffic Class field in the IPv6 header.

4.3.4.1.3.5. The pktSecCondIPv6FlowLabel Attribute

This is a mandatory string attribute, and defines an IPv6 flow label. This, in combination with the Source and Destination Address fields, enables efficient IPv6 flow classification by using only the IPv6 main header fields (20 bits).

4.3.4.1.3.6. The pktSecCondIPv6PayloadLength Attribute

This is a mandatory string attribute, and defines the total length of the packet (including the fixed and any extension headers, and data) in bytes (16 bits).

4.3.4.1.3.7. The pktSecCondIPv6NextHeader Attribute

This is a mandatory string attribute, and defines the type of the next header (e.g., which extension header to use) (8 bits).

4.3.4.1.3.8. The pktSecCondIPv6HopLimit Attribute

This is a mandatory string attribute, and defines the maximum number of hops that this packet can traverse (8 bits).

4.3.4.1.4. PacketSecurityTCPCondition

The purpose of this Class is to represent packet TCP packet header information that can be used as part of a test to determine if the set of Policy Actions in this ECA Policy Rule should be executed or not. This class is concrete, and defines the following attributes:

4.3.4.1.4.1. The pktSecCondTPCSrcPort Attribute

This is a mandatory string attribute, and defines the Source Port (16 bits).

4.3.4.1.4.2. The pktSecCondTPCDestPort Attribute

This is a mandatory string attribute, and defines the Destination Port (16 bits).

4.3.4.1.4.3. The pktSecCondTPCSeqNum Attribute

This is a mandatory string attribute, and defines the sequence number (32 bits).

4.3.4.1.4.4. The pktSecCondTPCFlags Attribute

This is a mandatory string attribute, and defines the nine Control bit flags (9 bits).

4.3.4.1.5. PacketSecurityUDPCondition

The purpose of this Class is to represent packet UDP packet header information that can be used as part of a test to determine if the set of Policy Actions in this ECA Policy Rule should be executed or not. This class is concrete, and defines the following attributes:

4.3.4.1.5.1. The pktSecCondUDPSrcPort Attribute

This is a mandatory string attribute, and defines the UDP Source Port (16 bits).

4.3.4.1.5.2. The pktSecCondUDPDestPort Attribute

This is a mandatory string attribute, and defines the UDP Destination Port (16 bits).

4.3.4.1.5.3. The pktSecCondUDPLength Attribute

This is a mandatory string attribute, and defines the length in bytes of the UDP header and data (16 bits).

4.3.4.2. PacketPayloadSecurityCondition

The purpose of this Class is to represent packet payload data that can be used as part of a test to determine if the set of Policy Actions in this ECA Policy Rule should be executed or not. Examples include a specific set of bytes in the packet payload.

4.3.4.3. TargetSecurityCondition

The purpose of this Class is to represent information about different targets of this policy (i.e., entities to which this policy rule should be applied), which can be used as part of a test to determine if the set of Policy Actions in this ECA Policy Rule

should be executed or not. Examples include whether the targeted entities are playing the same role, or whether each device is administered by the same set of users, groups, or roles.

This Class has several important subclasses, including:

- a. ServiceSecurityContextCondition is the superclass for all information that can be used in an ECA Policy Rule that specifies data about the type of service to be analyzed (e.g., the protocol type and port number)
- b. ApplicationSecurityContextCondition is the superclass for all information that can be used in a ECA Policy Rule that specifies data that identifies a particular application (including metadata, such as risk level)
- c. DeviceSecurityContextCondition is the superclass for all information that can be used in a ECA Policy Rule that specifies data about a device type and/or device OS that is being used

4.3.4.4. UserSecurityCondition

The purpose of this Class is to represent data about the user or group referenced in this ECA Policy Rule that can be used as part of a test to determine if the set of Policy Actions in this ECA Policy Rule should be evaluated or not. Examples include the user or group id used, the type of connection used, whether a given user or group is playing a particular role, or whether a given user or group has failed to login a particular number of times.

4.3.4.5. SecurityContextCondition

The purpose of this Class is to represent security conditions that are part of a specific context, which can be used as part of a test to determine if the set of Policy Actions in this ECA Policy Rule should be evaluated or not. Examples include testing to determine if a particular pattern of security-related data have occurred, or if the current session state matches the expected session state.

4.3.4.6. GenericContextSecurityCondition

The purpose of this Class is to represent generic contextual information in which this ECA Policy Rule is being executed, which can be used as part of a test to determine if the set of Policy Actions in this ECA Policy Rule should be evaluated or not. Examples include geographic location and temporal information.

4.3.5. Network Security Action Sub-Model

Figure 13 shows a more detailed design of the Action subclasses that are contained in the Network Security Information Sub-Model.

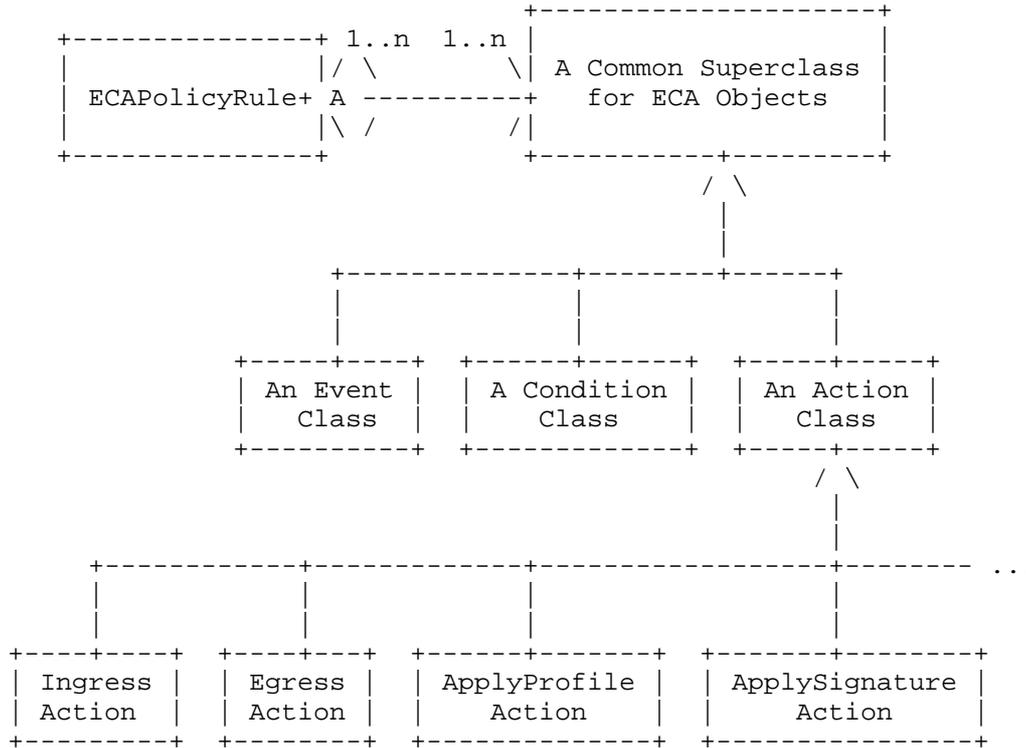


Figure 13. Network Security Info Sub-Model Action Extensions

The four Action classes shown in Figure 13 extend the (external) generic Action class to represent Actions that perform a Network Security Control function. Brief class descriptions are provided in the following sub-sections.

4.3.5.1. IngressAction

The purpose of this Class is to represent actions performed on packets that enter an NSF. Examples include pass, drop, mirror traffic.

4.3.5.2. EgressAction

The purpose of this Class is to represent actions performed on packets that exit an NSF. Examples include pass, drop, mirror traffic, signal, encapsulate.

4.3.5.3. ApplyProfileAction

The purpose of this Class is to represent applying a profile to packets to perform content security and/or attack mitigation control.

4.3.5.4. ApplySignatureAction

The purpose of this Class is to represent applying a signature file to packets to perform content security and/or attack mitigation control.

4.4. Information Model for Content Security Control

The block for content security control is composed of a number of security capabilities, while each one aims for protecting against a specific type of threat in the application layer.

Following figure shows a basic structure of the information model:

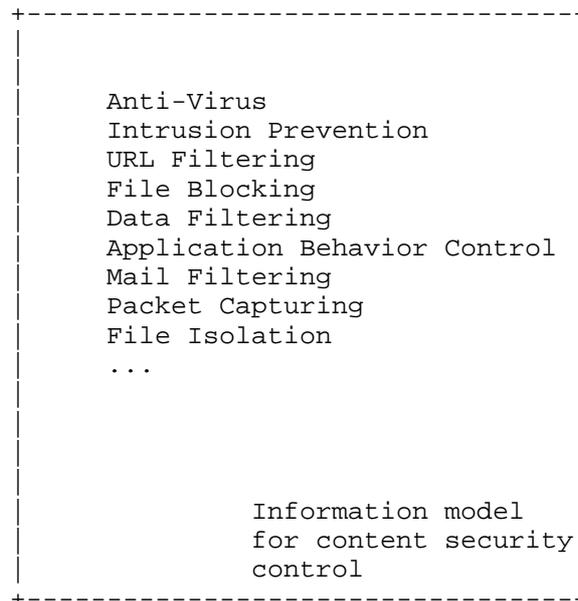


Figure 14. The basic structure of information model for content security control

The detailed description about the standard interface and the parameters for all the security capabilities of this category are TBD.

4.5. Information Model for Attack Mitigation Control

The block for attack mitigation control is composed of a number of security capabilities, while each one aims for mitigating a specific type of network attack.

Following figure shows a basic structure of the information model:

7. References

7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2234] Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, Internet Mail Consortium and Demon Internet Ltd., November 1997.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [RFC5511] Farrel, A., "Routing Backus-Naur Form (RBNF): A Syntax Used to Form Encoding Rules in Various Routing Protocol Specifications", RFC 5511, April 2009.

7.2. Informative References

- [INCITS359 RBAC] NIST/INCITS, "American National Standard for Information Technology - Role Based Access Control", INCITS 359, April, 2003
- [I-D.draft-ietf-i2nsf-problem-and-use-cases] Hares, S., et.al., "I2NSF Problem Statement and Use cases", Work in Progress, February, 2016.
- [I-D.draft-ietf-i2nsf-framework] Lopez, E., et.al., "Framework for Interface to Network Security Functions", Work in Progress, May, 2016.
- [I-D.draft-ietf-i2nsf-terminology] Hares, S., et.al., "Interface to Network Security Functions (I2NSF) Terminology", Work in Progress, April, 2016
- [I-D.draft-ietf-supra-generic-policy-info-model] Strassner, J., Halpern, J., Coleman, J., "Generic Policy Information Model for Simplified Use of Policy Abstractions (SUPA)", Work in Progress, June, 2016.

8. Acknowledgments

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

Appendix A.

This Appendix specifies the information model of security policy in Routing Backus-Naur Form [RFC5511]. This grammar is intended to help the reader better understand the english text description in order to derive a data model.

Firstly, several types of route are specified as follows:

- o IPv4: Match on destination IP address in the IPv4 header
- o IPv6: Match on destination IP address in the IPv6 header
- o MPLS: Match on a MPLS label at the top of the MPLS label stack
- o MAC: Match on MAC destination addresses in the ethernet header
- o Interface: Match on incoming/outcoming interface of the packet

Then, the I2NSF information model grammar of security policy is specified as follows:

```
<Policy> ::= <policy-name> <policy-id> (<Rule> ...)
```

```
<Rule> ::= <rule-name> <rule-id> <Match> <Action>
```

```
<Match> ::= [<subject-based-match>] [<object-based-match>]
```

```
<subject-based-match> ::= [<L234-packet-header> ...]
```

```
    [<packet-payload> ...]
```

```
<L234-packet-header> ::= [<address-scope>] [<layer-2-header>]
```

```
    [<layer-3-header>] [<layer-4-header>]
```

```
<address-scope> ::= <route-type> (<ipv4-route> | <ipv6-route> |  
    <mpls-route> | <mac-route> | <interface-route>)
```

```
<route-type> ::= <IPV4> | <IPV6> | <MPLS> | <IEEE_MAC> | <INTERFACE>
```

```
<ipv4-route> ::= <ip-route-type> (<destination-ipv4-address> |
```

```
<source-ipv4-address> | (<destination-ipv4-address>
<source-ipv4-address>))
<destination-ipv4-address> ::= <ipv4-prefix>
<source-ipv4-address> ::= <ipv4-prefix>
<ipv4-prefix> ::= <IPV4_ADDRESS> <IPV4_PREFIX_LENGTH>

<ipv6-route> ::= <ip-route-type> (<destination-ipv6-address> |
<source-ipv6-address> | (<destination-ipv6-address>
<source-ipv6-address>))
<destination-ipv6-address> ::= <ipv6-prefix>
<source-ipv6-address> ::= <ipv6-prefix>
<ipv6-prefix> ::= <IPV6_ADDRESS> <IPV6_PREFIX_LENGTH>
<ip-route-type> ::= <SRC> | <DEST> | <DEST_SRC>

<layer-3-header> ::= <ipv4-header> | <ipv6-header>
<ipv4-header> ::= <SOURCE_IPv4_ADDRESS> <DESTINATION_IPv4_ADDRESS>
<PROTOCOL> [<TTL>] [<DSCP>]
<ipv6-header> ::= <SOURCE_IPV6_ADDRESS> <DESTINATION_IPV6_ADDRESS>
<NEXT_HEADER> [<TRAFFIC_CLASS>]
[<FLOW_LABEL>] [<HOP_LIMIT>]

<object-based-match> ::= [<user> ...] [<schedule>] [<region>]
[<target>] [<state>]
<user> ::= (<login-name> <group-name> <parent-group> <password>
```

```
<expired-date> <allow-multi-account-login>
<address-binding>) | <tenant> | <VN-id>
<schedule> ::= <name> <type> <start-time> <end-time>
                <weekly-validity-time>
<type> ::= <once> | <periodic>
<target> ::= [<service>] [<application>] [<device>]
<service> ::= <name> <id> <protocol> [<protocol-num>] [<src-port>]
                [<dest-port>]
<protocol> ::= <TCP> | <UDP> | <ICMP> | <ICMPv6> | <IP>

<application> ::= <name> <id> <category> <subcategory>
                <data-transmission-model> <risk-level>
                <signature>
<category> ::= <business-system> | <Entertainment> | <internet>
                <network> | <general>
<subcategory> ::= <Finance> | <Email> | <Game> | <media-sharing> |
                <social-network> | <web-posting> | <proxy> | ...
<data-transmission-model> ::= <client-server> | <browser-based> |
                <networking> | <peer-to-peer> |
                <unassigned>
<risk-level> ::= <Exploitable> | <Productivity-loss> | <Evasive> |
                <Data-loss> | <Malware-vehicle> |
                <Bandwidth-consuming> | <Tunneling>
```

```
<signature> ::= <server-address> <protocol> <dest-port-num>
                <flow-direction> <object> <keyword>

<flow-direction> ::= <request> | <response> | <bidirection>

<object> ::= <packet> | <flow>

<device> ::= <pc> | <mobile-phone> | <tablet>

<session-state> ::= <new> | <established> | <related> | <invalid> |
                    <untracked>

<action> ::= <basic-action> [<advanced-action>]

<basic-action> ::= <pass> | <deny> | <mirror> | <call-function> |
                    <encapsulation>

<advanced-action> ::= [<profile-antivirus>] [<profile-IPS>]
                    [<profile-url-filtering>]
                    [<profile-file-blocking>]
                    [<profile-data-filtering>]
                    [<profile-application-control>]
```

Authors' Addresses

Liang Xia (Frank)
Huawei

101 Software Avenue, Yuhuatai District
Nanjing, Jiangsu 210012
China

Email: Frank.xialiang@huawei.com

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

Kepeng Li
Alibaba

Email: kepeng.lkp@alibaba-inc.com

DaCheng Zhang
Alibaba

Email: Dacheng.zdc@alibaba-inc.com

Edward Lopez
Fortinet
899 Kifer Road
Sunnyvale, CA 94086
Phone: +1 703 220 0988

EMail: elopez@fortinet.com

Nicolas BOUTHORS
Qosmos

Email: Nicolas.BOUTHORS@qosmos.com

Luyuan Fang
Microsoft
15590 NE 31st St
Redmond, WA 98052
Email: lufang@microsoft.com

I2nsf Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2017

J. You
Huawei
M. Zarny
Goldman Sachs
C. Jacquenet
M. Boucadair
France Telecom
Y. Li
J. Strassner
Huawei
S. Majee
F5 Networks
July 8, 2016

User-group-based Security Policy for Service Layer
draft-you-i2nsf-user-group-based-policy-02

Abstract

This draft defines the User-group Aware Policy Control (UAPC) framework, which facilitates consistent enforcement of security policies based on user group identity. Policies are used to control security policy enforcement using a policy server and a security controller. Northbound APIs are also discussed.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 9, 2017.

Copyright Notice

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

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

Table of Contents

- 1. Introduction 2
- 2. Terminology 4
 - 2.1. Abbreviations and acronyms 4
 - 2.2. Definitions 4
- 3. Use Cases for User-group Aware Policy Control 5
- 4. User-group Aware Policy Control 6
 - 4.1. Overview 6
 - 4.2. Functional Entities 7
 - 4.3. User Group 9
 - 4.4. Inter-group Policy Enforcement 10
 - 4.5. UAPC Implementation 12
- 5. Requirements for I2NSF 13
- 6. Security Considerations 14
- 7. IANA Considerations 14
- 8. Acknowledgements 14
- 9. References 14
 - 9.1. Normative References 14
 - 9.2. Informative References 14
- Authors' Addresses 15

1. Introduction

In traditional networks, network access is typically controlled through a combination of mechanisms such as maintaining separate static VLAN/IP subnet assignments per organization, applying Access Control Lists (ACLs) on VLANs and/or IP subnets, leveraging Network Access Control (NAC). Common side effects are:

- o Network administrators typically assume that users access the network from their own static location--from their assigned switch, VLAN, IP subnet, etc.

- o MAC or IP address of the users' device is often used as a proxy for the user's identity. As such, filtering (e.g., via ACLs) of the user is usually based on IP or MAC addresses.

- o Authentication of the user by the network, if it exists at all, typically takes place only at the access switch in conjunction with an AAA (Authentication, Authorization, Accounting) server. Different authentication mechanisms could be used - from machine-based certificates to username/password challenges, to just "authenticating" on MAC addresses, etc.

- o Network security functions such as firewalls often act only on IP addresses and ports - not on the user's identity.

These are all symptoms of a system not using actual user identification information, but rather, one or more attributes that attempt to represent a user identity.

Traditional network access control mechanisms [I-D.ietf-i2nsf-problem-and-use-cases] do not work well in newer network paradigms.

- o First, both clients and servers can move and change their IP addresses on a regular basis. For example, Wi-Fi and VPN clients, as well as back-end Virtual Machine (VM)-based servers, can move; their IP addresses could change as a result. This means relying on well-known network fields (e.g., the 5-tuple) is increasingly inadequate to ensure consistent security policy enforcement.

- o Secondly, with more people working from non-traditional office setups like "working from home", there is now a need to be able to apply different security policies to the same set of users under different circumstances. Network access needs to be granted based on such criteria as users' location, time-of-day, type of network device used (e.g., corporate issued device versus personal device), device's security posture, etc. This means the network needs to recognize the users' identity and their current context, and map the users to their correct access entitlement to the network.

- o Moreover, implementation of coherent security policy across several network and network security devices is almost impossible. NSFs in operation could be sourced from different vendors, or could be different hardware models/software versions by the same vendor. As a result, the capabilities as well as APIs of the NSFs may not be the same throughout the environment. Finally, few enterprises, if any, have a complete view of all the application flows. It is not uncommon for administrators to update a policy

on a firewall, only to later find out that related ACLs, firewall policies, and other related mechanisms were not updated.

Today, addressing the above issues takes considerable time and effort. Most network administrators have to manually plan and implement necessary changes as little automation, if any, exists across diverse sets of network security platforms. In line with the I2NSF effort to standardize APIs so as to facilitate automation, this draft defines User-group Aware Policy Control (UAPC), which facilitates consistent enforcement of policies based on user-group identity, and discusses how it operates in the I2NSF Service Layer [I-D.ietf-i2nsf-framework].

2. Terminology

2.1. Abbreviations and acronyms

AAA: Authentication, Authorization, and Accounting

ACL: Access Control List

ADSL: Asymmetric Digital Subscriber Line

AP: Access Point

LTE: Long Term Evolution

NAC: Network Admission Control

NBI: Northbound Interface

NSF: Network Security Function

UAPC: User-group Aware Policy Control

VLAN: Virtual Local Area Network

2.2. Definitions

User: An individual or a group of individuals that act as a single entity.

User-group: A group of users that share one or more characteristics and/or behaviors in common, which allows each user in the user-group to be assigned the same access control permissions. For example, sales employees are treated with equivalent service policy rules when accessing the network.

Profile: A set of capabilities, in terms of functions and behaviors, for a given entity or set of entities.

Role: A role defines a set of responsibilities of an object that it is attached to. This enables the functions and behavior of a complex object to be abstracted into just those that are required by a client in a particular context.

User-group Identifier (User-group ID): An identifier that represents the collective identity of a group of users, and is determined by a set of one or more matching criteria (e.g., roles, 4-, 5-, and 6-tuples, VLAN ID, etc.) that disambiguates this user-group entity from other entities.

3. Use Cases for User-group Aware Policy Control

With the increased popularity of enterprise wireless networks and remote access technologies such as Virtual Private Networks (VPN), enterprise networks have become borderless, and employees' locations can be anywhere. Enabling large-scale employee mobility across many access locations improves enterprise production efficiency but also introduces challenges related to enterprise network management and security. The IP address of the user can change frequently when the user is in motion. Consequently, IP address-based policies (such as forwarding, routing, QoS and security policies) may not be flexible enough to accommodate users in motion.

The User-group Aware Policy Control (UAPC) approach is intended to facilitate the consistent enforcement of policies. As shown in Figure 1, a multi-technology network (e.g., Wi-Fi, 3G/LTE, ADSL and fiber infrastructures) can connect different types of terminal devices (e.g., Smartphone, tablet, and laptop) which should be able to access networks in a secure manner. Security policies should be consistently enforced based on their user-group identities, regardless of whether these terminal devices connect to a wired or a wireless infrastructure.

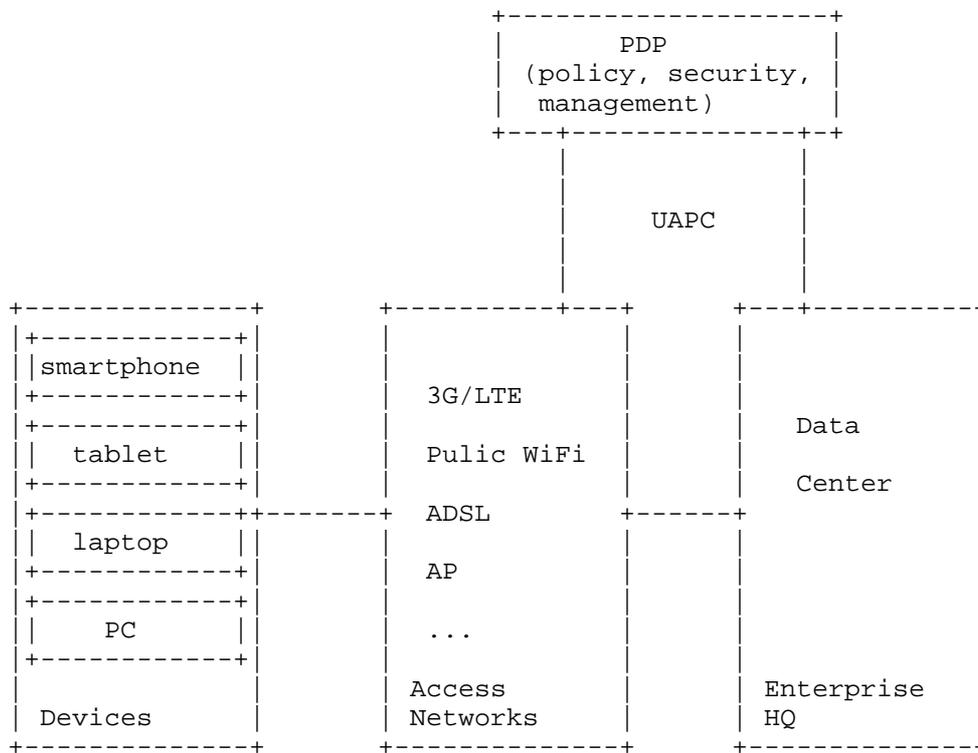


Figure 1: UAPC Framework Example

4. User-group Aware Policy Control

4.1. Overview

The UAPC framework is as follows enables users to be authenticated and classified into different user-groups at the network ingress by the Security Controller; this may require obtaining information from the Policy Server and an AAA server. The user-group is an identifier that represents the collective identity of a group of users, and is determined by a set of pre-defined policy criteria (e.g., source IP address, geo-location data, time of day, or device certificate). Users may be moved to different user-groups if their composite security context and/or environment change.

The Security Controller, if necessary, pushes the required user-group policies to all Network Security Functions (NSFs) that need them. The policies are expressed as user-group (not IP or MAC address) IDs so as to decouple the user identity from the network addresses of the user's device.

(Note that User-group IDs may be implemented in at least two ways: (1) the ingress switch inserts the user-group ID into the packets, and downstream NSF's match and act on the user-group ID, or (2) the Security Controller updates each NSF with the mapping between the user-group IDs and the packet tuples; NSF's map incoming packets to their rightful user-group IDs, and act on the user-group IDs. These and other implementation methodologies are out of scope of this document.)

The security policy provisioning information can be derived from the user's profile and credentials, as well as the group to which the user belongs; such information can also be derived from the outcomes of the dynamic security service parameter negotiation that could possibly take place between the user and the service provider or the network administrator (e.g., parameters like whether the user is entitled to access the enterprise network while in motion or not, the lease time associated to an IP address, whether the user can access the Internet or not, and whether traffic needs to be encrypted or not). This information is transferred to the Network Security Functions (NSF) from the controller. Once an incoming packet matches a certain user group on the NSF, the corresponding security policy will be enforced on that packet.

4.2. Functional Entities

The UAPC framework consists of four main components: (1) Policy Server, (2) Authentication Server, (3) Security Controller, (4) Network Security Functions:

- o Policy Server

The Policy Server houses two policy databases: (1) the user-group criteria, which assigns users to their user-group, and (2) the rule base of what each user group has access to.

- Contains (G)UI and/or APIs to enable policies to be created, modified, and deleted using command line, graphical tools, and/or programming logic
- Contains logic to create, read, update, and delete policies and policy components, and apply policies to user-groups from one or more policy repositories
- Contains logic to detect conflicts between policies
- Contains logic to resolve conflicts between policies

- Contains logic to broker and/or federate policies between domains

The above subjects are beyond the scope of this document.

o AAA Server

The AAA Server authenticates users, and then performs associated authorization and accounting functions. The AAA server classifies users into different user-groups at the network ingress. AAA server implementation details are out of scope for this document.

o Security Controller

The Security Controller coordinates various network security-related tasks on a set of NSFs under its administration. In general, there may be multiple security domains, where each domain has its own security controller. The detailed architecture is beyond the scope of this document.

- Authenticates the user at the ingress using an authentication service. While the authentication functionality is an integral part of the framework, the topics of defining and managing authentication rules are out of scope of this document.
- Asks policy server for decisions to security-related requests; takes these decisions and invokes the set of NSFs that are required to implement security for that particular packet. The security controller may cache policies.
- May perform additional actions as specified by the metadata associated with a policy rule (e.g., the "function(s)" to be executed after the actions in a policy rule are executed)
- Has an authoritative database of NSFs under its administration
- Determines on which NSFs a given policy needs to be enforced
- Presents a set of NBIs for applications, orchestration engines, etc.
- Interfaces with NSFs via (to-be-developed) I2NSF Capability Layer APIs.

o Network Security Functions

- Packet classification: Depending on the implementation model, the NSF may match on User-group IDs in the packets; or it may

match on common packet header fields such as the 5-tuple, and map the n-tuple to the appropriate User-group ID supplied out-of-band by the Security Controller.

- Policy enforcement: Enforce the corresponding policy (or set of policies) if the packet matches a specified User-group ID or set of User-group IDs
- Presents I2NSF Capability Layer APIs

4.3. User Group

The user-group is an identifier that represents the collective identity of a group of users, whose definition is controlled by one or more policy rules (e.g., source IP, geo-location, time of day, and device certificate).

A given user is authenticated, and classified at the network ingress, and assigned to a user-group. (The term "user" refers to any user of the network. As such, servers, terminals and other devices are also classified and assigned to their respective user-groups.) A user's group membership may change as aspects of the user change. For example, if the user-group membership is determined solely by the source IP address, then a given user's user-group ID will change when the user moves to a new IP address that falls outside of the range of addresses of the previous user-group.

Table 1 shows an example of how user-group definitions may be constructed. User-groups may share several common criteria. That is, user-group criteria are not mutually exclusive. For example, the policy criteria of user-groups R&D Regular and R&D-BYOD may share the same set of users that belong to the R&D organization, and differ only in the type of client (firm-issued clients versus users' personal clients); likewise, the same user may be assigned to different user-groups depending on the time of day or the type of day (e.g., weekdays versus weekends); and so on.

Table 1: User-Group Example

Group Name	Group ID	Group Definition
R&D	10	R&D employees
R&D BYOD	11	Personal devices of R&D employees
Sales	20	Sales employees
VIP	30	VIP employees
Workflow	40	IP addresses of Workflow resource servers
R&D Resource	50	IP addresses of R&D resource servers
Sales Resource	54	IP addresses of Sales resource servers

4.4. Inter-group Policy Enforcement

Within the UAPC framework, inter-group policy enforcement requires two key components: (1) user-group-to-user-group access policies, and (2) sets of NSFs that are managed by sets of policies.

First, the framework calls for an authoritative rule-base that lists all the destination user-groups to which all the source user-groups are entitled to access. The rule-base, hosted on the Policy Server, enables administrators to construct authorized inter-group access relationships. The simple example in Table 2 shows a policy matrix in which the row represents source user-groups and the column represents destination ones. The inter-group rule-base is similar to firewall rule-bases, which are mostly made up of 5-tuples. (Firewall rule-bases could and do include criteria other than the standard 5-tuple. Also, the user-group rule-base could consist of other criteria. Actual implementation details are out of scope of this document.)

The responsibility of implementing and managing the inter-group policies falls to the Security Controller. The controller first needs to determine, (or is told) the specific NSFs on which a given policy is to be implemented. The controller then communicates with each NSF via the I2NSF APIs to execute the required tasks.

Table 2: Inter-group Policy Example

Source Group	Destination Group		
	Workflow Group	R&D Resource Group	Sales Resource Group
R&D group	Permit	Permit	Deny
R&D BYOD group	Traffic-rate	Deny	Deny
Sales group	Permit	Deny	Permit
VIP user group	Traffic-mark	Traffic-mark	Traffic-mark

Inter-user-group rules are configurable. Figure 2 illustrates how various user-groups and their entitlements may be structured. The example shows a "north-south" model that shows how users may access internal network resources. Similar models can be developed for "east-west" intra-data center traffic flows.

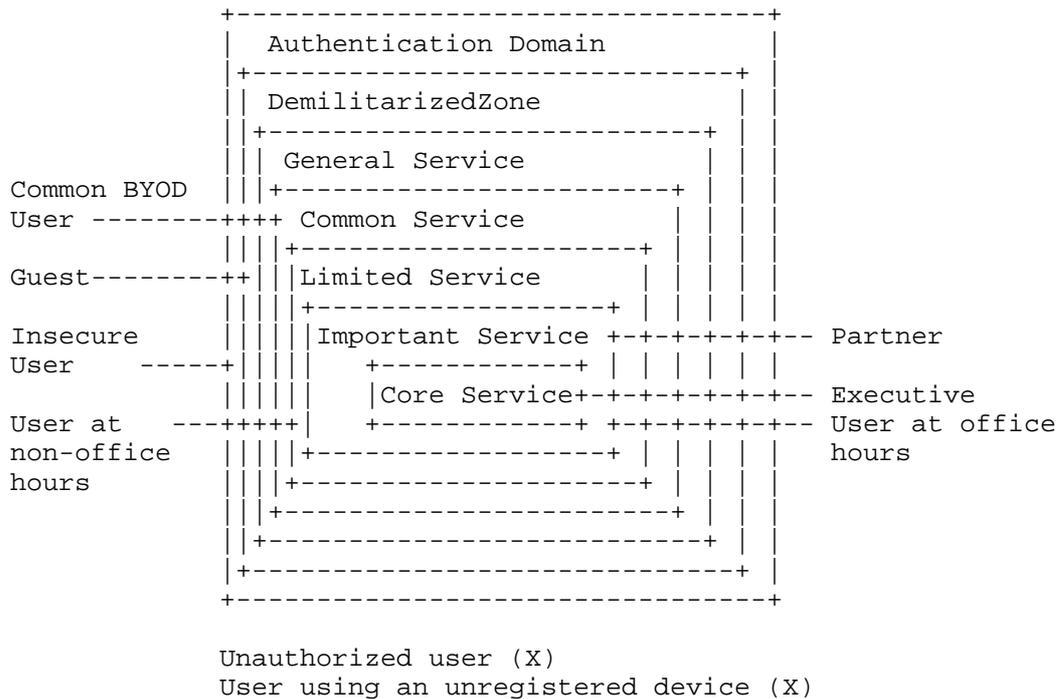


Figure 2: Sample Authorization Rules for User-group Aware Policy Control

4.5. UAPC Implementation

The security policies are instantiated and maintained by the policy server. The associated computation logic (to instantiate such policies) may be dynamically fed with instructions coming from the application. The policy decisions could also be from the outcomes of dynamic security service parameter negotiations that typically take place at the management plane between the user and the service provider [RFC7297].

The NSF's receive group-based policy provisioning information from the security controller. The security policies will be enforced so that participating NSF's can process traffic accordingly. There are five steps for implementing the UAPC framework, which are shown in Figure 3.

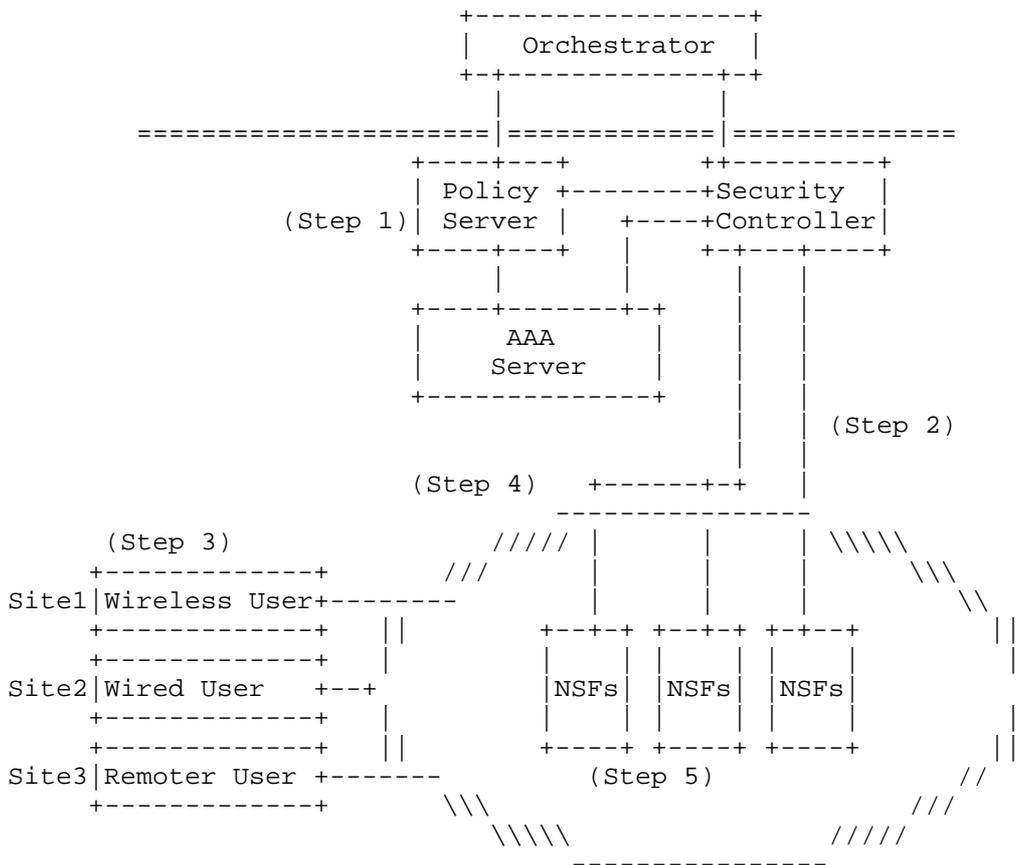


Figure 3: Unified Policy Procedures

1. User-group identification policies and inter-user-group access policies on the Policy Server are managed by authorized user(s) and/or team(s).
 2. The user-group-based policies are implemented on the NSFs under the Security Controller's management.
 3. When a given user first comes logs onto the network, the user is authenticated at the ingress switch.
 4. If the authentication is successful, the user is placed in a user-group, as determined by the Policy Server. If the authentication is not successful, then the user is not assigned a user-group, which means that the user has no access permissions for the network.
 5. The user's subsequent traffic is allowed or permitted based on the user-group ID by the NSFs per the inter-user-group access policies. (It is beyond the scope of this document as to how user-group IDs may be delivered to non-ingress NSFs. See Section 4.1 for a brief overview of possible implementation methods.)
5. Requirements for I2NSF

Key aspects of the UAPC framework fall within the Service Layer of the I2NSF charter. If the community adopts the approach as one possible framework for the Service Layer, the I2NSF Service Layer MUST support at least the following northbound APIs (NBIs):

- o The user-group classification policy database on the Policy Server
- o The inter-user-group access policy rule-base on the Policy Server
- o The inventory of NSFs under management by the Security Controller
- o The list of NSFs on which a given inter-user-group policy is to be implemented by the Security Controller.

The framework also assumes that the I2NSF Capability Layer APIs will be there for the NSFs.

6. Security Considerations

This document provides the UAPC framework, and discusses how it operates in the I2NSF Service Layer. It is not intended to represent any particular system design or implementation, nor does it define a protocol, and as such it does not have any specific security requirements.

7. IANA Considerations

This document has no actions for IANA.

8. Acknowledgements

The editors would like to thank Linda Dunbar for a thorough review and useful comments.

9. References

9.1. 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>>.

[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>>.

9.2. Informative References

[I-D.ietf-i2nsf-framework] elopez@fortinet.com, e., Lopez, D., Dunbar, L., Strassner, J., Zhuang, X., Parrott, J., Krishnan, R., and S. Durbha, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-02 (work in progress), July 2016.

[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-00 (work in progress), February 2016.

Authors' Addresses

Jianjie You
Huawei
101 Software Avenue, Yuhuatai District
Nanjing, 210012
China

Email: youjianjie@huawei.com

Myo Zarny
Goldman Sachs
30 Hudson Street
Jersey City, NJ 07302
USA

Email: myo.zarny@gs.com

Christian Jacquenet
France Telecom
Rennes 35000
France

Email: christian.jacquenet@orange.com

Mohamed Boucadair
France Telecom
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Yizhou Li
Huawei
101 Software Avenue, Yuhuatai District
Nanjing, 210012
China

Email: liyizhou@huawei.com

John Strassner
Huawei
2330 Central Expressway
San Jose, CA
USA

Email: john.sc.strassner@huawei.com

Sumandra Majee
F5 Networks
3545 N 1st St
San Jose, CA 95134

Email: S.Majee@f5.com

I2nsf Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 19, 2017

J. You
J. Strassner
Huawei
M. Zarny
Independent
C. Jacquenet
France Telecom
S. Majee
F5 Networks
July 18, 2016

User-Group-based Security Policy for Capability Layer
draft-you-i2nsf-user-group-policy-capability-00

Abstract

This draft defines the I2NSF Capability APIs for implementing User-Group-based Security Policies using the Event-Condition-Action Policy Rule paradigm.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 19, 2017.

Copyright Notice

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

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

Table of Contents

1.	Introduction	2
2.	Terminology	3
2.1.	Abbreviations and Acronyms	3
2.2.	Definitions	3
3.	Overall Architecture	4
3.1.	Alternative 1: Using Packet User-Group Labels	4
3.2.	Alternative 2: Using User-group IDs Directly	5
4.	ECA for User-group-based Security Policy	5
4.1.	Information Model Design	6
4.2.	FlowSpecECAPolicyRule Class Definition	6
4.3.	Event	8
4.4.	Condition	8
4.5.	Action	8
4.5.1.	Traffic-rate Action	8
4.5.2.	Traffic-detail Action	9
4.5.3.	Redirect Action	9
4.5.4.	Traffic-marking Action.	9
4.6.	Capability Layer Rules	9
5.	Security Considerations	9
6.	IANA Considerations	9
7.	Acknowledgements	9
8.	References	9
8.1.	Normative References	9
8.2.	Informative References	10
	Authors' Addresses	11

1. Introduction

In traditional networks, network access is typically controlled through a combination of mechanisms. These include maintaining separate static VLAN/IP subnet assignments per organization, applying Control Lists (ACLs) on VLANs and/or IP subnets, and leveraging Network Access Control(NAC). However, traditional access control mechanisms ([I-D.ietf-i2nsf-problem-and-use-cases]) do not work well with newer network paradigms and architectures. Most network administrators have to manually plan and implement necessary changes because little, if any, automation exists across diverse sets of network security platforms.

[I-D.you-i2nsf-user-group-based-policy] discusses User-Group aware Policy Control (UAPC). This facilitates consistent enforcement of policies based on User-Group identity, since the User-Group IDs are generated by a set of ECA Policy Rules. It also discusses how this can be used in the I2NSF Service Layer [I-D.ietf-i2nsf-framework]. The UAPC mechanism calls for: (1) a User-Group identifier (e.g., source and destination IP address, time-of-day, device certificate, or a combination of these and other similar attributes); (2) a policy server service that maintains policies for defining and managing user-groups as well as permissions associated with user-groups; and (3) a logical security controller that is responsible for managing Network Security Functions (NSFs), and installing necessary policies on them.

This document proposes I2NSF capability layer APIs for implementing User-Group-based security policies by using the Event-Condition-Action (ECA) Policy Rule paradigm.

2. Terminology

This section contains terminology and definitions that are important for understanding the technical content of this document.

2.1. Abbreviations and Acronyms

AAA: Authentication, Authorization, and Accounting
ECA: Event-Condition-Action
NSF: Network Security Function
UAPC: User-Group Aware Policy Control
VRF: Virtual Routing and Forwarding instance

2.2. Definitions

Event: An event is defined as any important occurrence in time of a change in the system being managed, and/or in the environment of the system being managed. An Event, when used in the context of a Policy Rule, is used to determine whether the Condition clause of an imperative Policy Rule can be evaluated or not. For an ECA Policy Rule, if the Event clause is TRUE, then the Condition Clause MUST be evaluated.

Condition: A set of attributes, features, and/or values that are to be compared with a set of known attributes, features, and/or values in order to make a decision. A Condition, when used in the context of a Policy Rule, is used to determine whether or not the set of Actions in that Policy Rule can be executed or not. For an ECA Policy Rule, if the Condition clause is TRUE, then at least one Action contained in this ECA Policy Rule MUST be evaluated.

Action: An Action is a set of purposeful activity that has associated behavior. An Action, when used in the context of a Policy Rule, may be executed when both the Event and the Condition clauses of its owning Policy Rule evaluate to true. The execution of this Action MAY be influenced by applicable metadata.

3. Overall Architecture

The Security Controller coordinates various network security-related tasks on a set of NSFs under its administration, and invokes the set of NSFs that are required to implement security for particular packets.

The NSF may match on User-Group IDs in the packets, or it may match on common packet header fields such as an n-tuple, and then map the n-tuple to the appropriate User-Group ID supplied out-of-band by the Security Controller. If the packet matches a specified User-Group ID, the NSF enforces the corresponding policies.

The interface between the Security Controller and the NSF is called the capability interface in the I2NSF context. This document describes the I2NSF capability layer API for implementing User-Group-based security policies by using policy rules that are defined in an Event-Condition-Action (ECA) structure.

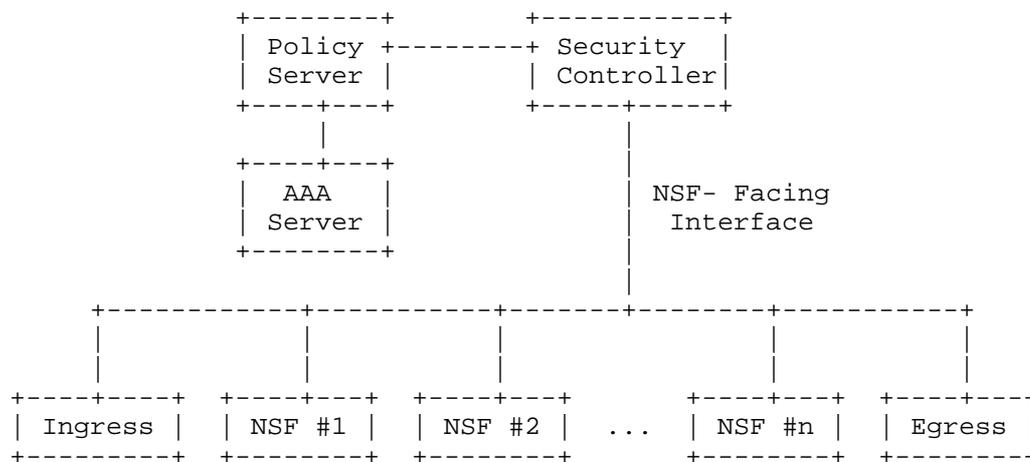


Figure 1. Functional Architecture

3.1. Option 1: Using Packet User-Group Labels

This option empoloyes the User-Group Label (UGL), a packet header field, to carry the User-Group ID. UGLs are disseminated using the Registration Interface (see [I-D.ietf-i2nsf-terminology] and [I-D.ietf-i2nsf-framework]).

UGs work in a 'UGL-capable' domain. This is a domain in which all NSFs residing in that domain support both the UGL field as well as mapping the UGL field (and hence, the User-Group ID that the UGL field represents) to a set of ECA Policy Rules.

In this alternative, the ingress NSF matches on common packet header fields, such as an n-tuple, and then maps the n-tuple to the appropriate User-Group ID supplied out-of-band by the Security Controller. Then, the ingress NSF inserts the corresponding UGL into the packet. The UGL is used by NSFs to make forwarding decisions in the UGL-supported domain.

The UGL field can be inserted into the packet using, for example, SFC encapsulation [I-D.ietf-sfc-nsh]. One benefit of option 1 is that only edge NSFs need to do UGL insertion or removal; other NSFs can easily enforce User-Group based policies based on the UGL carried in the packet. If the UGL field does not match any policy, a UGL-compliant NSF may apply a default policy, such as dropping or forwarding, to the packet. For example, if no policy is matched, firewalls typically will drop the packet, since Firewalls define a deny action as the default action to use.

3.2. Option 2: Using User-0Group IDs Directly

This alternative option relies on all NSFs being able to identify appropriate fields in a packet, map them to an appropriate User-Group, and then use that User-Group to control actions on the packet. In this option, User-Groups are disseminated using the Registration Interface.

The Ingress NSF matches on common packet header fields, such as an n-tuple, and then maps the n-tuple to the appropriate User-Group ID (which is supplied out-of-band by the Security Controller). Then, the ingress NSF enforces User-Group Policy Rules based on the identified User-Group ID, without inserting any field into the packet. Hence, option 2 requires all NSFs on the forwarding path to support User-Group mapping. If the NSF fails to associate the packet with a User-Group, it may use a default Policy Rule that is associated with, for example, a default unknown User-Group. The Policy associated with the default unknown User-Group should then be enforced. For example, an unknown User-Group could be mapped to a null VLAN, and a message sent to the appropriate Controller to perform basic security checking on it and then assign a real User-Group to the packet or flow (and a corresponding VLAN).

4. ECA for User-group-based Security Policy

This document uses Policy Rules, in the form of an Event-Condition-Action (ECA) structure, to describe User-Group-based security policies.

The fundamental constructs of ECA languages are reactive rules of the form:

```

IF the event clause evaluates to TRUE
  IF the condition clause evaluates to TRUE
    THEN execute actions in the action clause
  ENDIF
ENDIF
    
```

The Event clause, Condition clause, and Action clause collectively form a three-tuple. The Event and Condition clauses are made up of one or more Boolean statements. If the Event clause evaluates to FALSE, execution stops. If the Event clause evaluates to TRUE, then the Condition clause is evaluated. Once again, if the Condition clause evaluates to FALSE, execution stops; otherwise, Actions in the Policy Rule may be executed.

4.1. Information Model Design

The information model design for User-Group-based Security Policies is based on the information model design of the capability interface [I-D.draft-xia-i2nsf-capability-interface-im].

It is assumed that an external model, or set of models, is used to define the concept of an ECA Policy Rule and its components (e.g., Event, Condition, and Action objects). A functional block diagram of the ECA Model is shown in Figure 2 below:

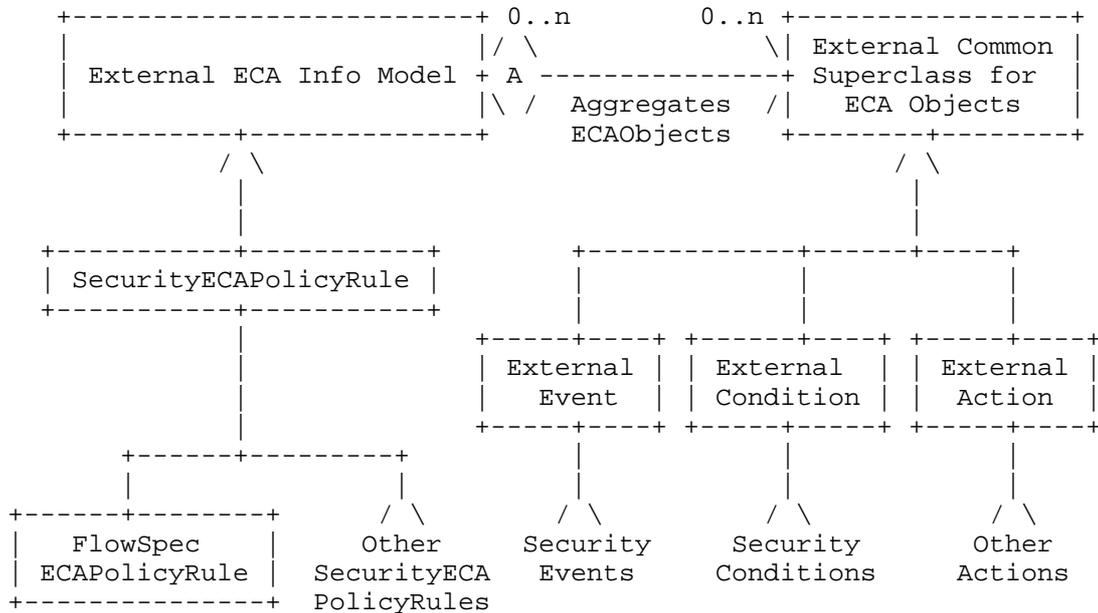


Figure 2. High-Level Information Model Design

The SecurityECAPolicyRule is the top of the User-Group ECA Policy Rule hierarchy. It inherits from the (external) generic ECA Policy Rule to define Security ECA Policy Rules that are specific to managing User-Groups. The SecurityECAPolicyRule contains all of the attributes, methods, and relationships defined in its (generic) superclass, and adds additional concepts that are required for Network Security (these will be defined in the next version of this draft).

The AggregatesECAObjects relationship defines the set of Events, Conditions, and Actions that are aggregated by a specific ECAPolicyRule. This aggregation is inherited by all subclasses of SecurityECAPolicyRule.

This draft defines a single subclass of SecurityECAPolicyRule, called FlowSpecECAPolicyRule. Additional ECA Policy Rules will be defined in future versions of this draft.

This draft defines four subclasses of the (external) generic Action class. Additional Event, Condition, and Action subclasses will be defined in future versions of this draft. Note that any of the Events, Conditions, and Actions defined in [I-D.draft-xia-i2nsf-capability-interface-im] can also be used; this is because the FlowSpecECAPolicyRule is really a container, which is meant to aggregate Events, Conditions, and Actions. The Flow Spec Rule is thus generic; application-specific needs are provided by choosing a suitable set of Events, Conditions, and Actions.

It is assumed that the (external) generic ECAPolicyRule class defines basic information in the form of attributes, such as a unique object ID, as well as a description and other basic, but necessary, information. It is also assumed that the (external) generic ECA Policy Rule is abstract; the SecurityECAPolicyRule is also abstract. This enables data model optimizations to be made while making this information model detailed but flexible and extensible.

The SecurityECAPolicyRule defines network security policy as a container that aggregates Event, Condition, and Action objects, which are described in Sections 4.3, 4.4, and 4.5, respectively. Events, Conditions, and Actions can be generic or security-specific.

Brief class descriptions of these classes are provided in the following sub-sections. In addition, none of the ECAPolicyRule subclasses will define attributes. This enables them to be viewed as simple object containers, and hence, applicable to a wide variety of content. It also means that the content of the function is defined solely by the set of Events, Conditions, and Actions that are contained by the particular subclass. This enables the Policy Rule, with its aggregated set of Events, Conditions, and Actions, to be treated as a reusable object.

4.2. FlowSpecECAPolicyRule Class Definition

The purpose of a FlowSpecECAPolicyRule is to define an ECA Policy Rule that can invoke FlowSpecs as Actions. The set of invoked Actions are triggered by a set of Events and Conditions. This ECA Policy Rule serves as a reusable container, and hence, the set of Events, Conditions, and Actions that it aggregates are also reused.

4.3. Event

Events are external stimuli, such as alarms, user actions (e.g., logon and logoff, or access requests), and packet arrival or departure occurrences. A set of exemplary Events are defined in [I-D.draft-xia-i2nsf-capability-interface-im].

4.4. Condition

Conditions are typically attributes or values that affect the state of a managed entity. These include:

- n-tuple of the incoming packet
- cross checking with other data, such as correlation with packets received from different ports or past time, or
- the current state of a flow

A set of exemplary Conditions are defined in [I-D.draft-xia-i2nsf-capability-interface-im].

4.5. Action

This document defines a minimum set of Actions. The first set of Actions are, based on [RFC5575] and [RFC7674]. This is not meant to be an inclusive list of all possible Actions, but only a subset that pertain specifically to flows, which can be interpreted consistently across the network. Other Actions will be defined in future versions of this document. An exemplary set of additional Actions is defined in [I-D.draft-xia-i2nsf-capability-interface-im].

Note that [RFC5575] and [RFC7674] define a general procedure to encode flow specification rules for aggregated traffic flows, so that they can be distributed as BGP [RFC4271] network layer reachability information.

4.5.1. Traffic-rate Action

The traffic-rate instructs a system to shape a certain stream to a set of predefined bandwidth characteristics. This is implemented using BGP extended community values attributes [RFC4360]. A traffic-rate of 0 should result in all traffic for this particular flow to be discarded.

4.5.2. Traffic-detail Action

The traffic-detail enables traffic sampling and logging for a particular flow. This is implemented using BGP extended community values attributes [RFC4360].

4.5.3. Redirect Action

The Redirect allows the traffic to be redirected to a specified VRF routing instance that lists the specified route-target in its import policy. This is implemented using BGP extended community values attributes [RFC4360].

4.5.4. Traffic-marking Action

The Traffic-marking instructs a system to modify the DSCP bits of a transiting IP packet to the corresponding value. This is implemented using BGP extended community values attributes [RFC4360].

4.6. Capability Layer Rules

This will be completed in the next version of this document.

5. Security Considerations

This document provides an Event-Condition-Action model for describing user-group-based security policies. It is not intended to represent any particular system design or implementation, nor does it define a protocol, and as such it does not have any specific security requirements.

6. IANA Considerations

This document has no actions for IANA.

7. Acknowledgements

TBD.

8. References

8.1. 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>>.

- [RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <<http://www.rfc-editor.org/info/rfc4271>>.
- [RFC4360] Sangli, S., Tappan, D., and Y. Rekhter, "BGP Extended Communities Attribute", RFC 4360, DOI 10.17487/RFC4360, February 2006, <<http://www.rfc-editor.org/info/rfc4360>>.
- [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>>.
- [RFC7674] Haas, J., Ed., "Clarification of the Flowspec Redirect Extended Community", RFC 7674, DOI 10.17487/RFC7674, October 2015, <<http://www.rfc-editor.org/info/rfc7674>>.

8.2. 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-01 (work in progress), July 2016.
- [I-D.ietf-i2nsf-framework]
Lopez, E., Lopez, D., Dunbar, L., Strassner, J., Zhuang, J., Parrott, J., Krishnan, R., Durbha, S., "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-02 (work in progress), July 2016
- [I-D.ietf-sfc-nsh]
Quinn, P. and Elzur, U., "Network Service Header", draft-ietf-sfc-nsh-05 (work in progress), May 2016.
- [I-D.you-i2nsf-user-group-based-policy]
You, J., Zarny, M., Jacquenet, C., Boucadair, M., Yizhou, L., Strassner, J., and S. Majee, "User-group-based Security Policy for Service Layer", draft-you-i2nsf-user-group-based-policy-02 (work in progress), July 2016.
- [I-D.draft-xia-i2nsf-capability-interface-im]
Xia, L. Strassner, J., Li, K., Zhang, D., Lopez, E., Bouthors, N., Fang, L., "Information Model of Interface to Network Security Functions Capability interface", draft-xia-i2nsf-capability-interface-im-06 (work in progress), July 2016

[I-D.ietf-i2nsf-terminology]

Hares, S., Strassner, J., Lopez, D., Xia, L., "Interface
to Network Security Functions (I2NSF) Terminology",
draft-ietf-i2nsf-terminology-01, July 2016

Authors' Addresses

Jianjie You
Huawei
101 Software Avenue, Yuhuatai District
Nanjing, 210012
China
Email: youjianjie@huawei.com

John Strassner
Huawei
2330 Central Expressway
San Jose, CA
USA
Email: john.sc.strassner@huawei.com

Myo Zarny
Independent
Email: myo.zarny@gmail.com

Christian Jacquenet
France Telecom
Rennes 35000
France
Email: christian.jacquenet@orange.com

Sumandra Majee
F5 Networks
3545 N 1st St
San Jose, CA 95134
Email: S.Majee@f5.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: January 2, 2017

D. Zhang
Y. Wu
Aliababa Group
L. Xia
Huawei
July 1, 2016

An Information Model for the Monitoring of Network Security Functions
(NSF)
draft-zhang-i2nsf-info-model-monitoring-01

Abstract

The Network Security Functions (NSF) Capability interface exists between the Service Provider's management system (or Security Controller) and the NSFs to enforce the rule provisioning and monitoring on the NSFs in the functional implementation level. This document focuses on the monitoring part of it and proposes the information model for it.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 2, 2017.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	3
2.	Terminology	3
2.1.	Key Words	3
2.2.	Definition of Terms	3
3.	Common Information	4
4.	Alarm	4
4.1.	System Alarm	4
4.1.1.	Memory Alarm	4
4.1.2.	CPU Alarm	5
4.1.3.	DISK Alarm	5
4.1.4.	Session Table Alarm	5
4.1.5.	Interface Alarm	5
4.2.	Security Event Alarm	6
4.2.1.	DDoS Alarm	6
4.2.2.	Virus Alarm	6
4.2.3.	Intrusion Alarm	7
4.2.4.	Botnet Alarm	8
4.2.5.	Web Attack Alarm	9
5.	Reports	10
5.1.	Attack Report	10
5.1.1.	DDoS Report	10
5.1.2.	Virus Report	10
5.1.3.	Intrusion Report	11
5.1.4.	Botnet Report	11
5.1.5.	Web Attack Report	12
5.2.	Service Report	12
5.2.1.	Traffic Report	12
5.2.2.	Policy HIT Report	13
5.2.3.	DPI Report	14
5.2.4.	Vulnerability Scanning Report	15
5.2.5.	User Activity Report	15
5.3.	System Report	16
5.3.1.	Operation Report	16
5.3.2.	Running Report	16
6.	IANA Considerations	17
7.	Security Considerations	17
8.	Acknowledgements	17
9.	References	17
9.1.	Normative References	17
9.2.	Informative References	17
	Authors' Addresses	18

1. Introduction

According to [I-D.ietf-i2nsf-framework], the interface provided by a NSF (e.g., FW, AAA, IPS, Anti-DDOS, or Anti-Virus) for other network entities (e.g., NMS, security controller) to enforce the rule provisioning and monitoring on the NSF is referred to as a 'capability interface'. The monitoring part of the capability interface is meant to monitor the network events and the execution status of the NSFs, then aggregate and analyze them to learn what is happening and send a report to the administrator. Regarding to monitoring, the NSF can communicate with the security controller though the capability interface in either a 'push' or a 'pull' way. The NSF can send out a report about its status or about certain network event proactively or send out message as a reply to security controller who control or monitor it. This document will not go into the design details of capability interface. Instead, this draft is focused on specifying the information that a NSF needs to provide in the monitoring part of the capability interface, as well as its information model. Besides, [I-D.draft-xia-i2nsf-capability-interface-im] specifies the information model for the rule provisioning part of the capability interface.

In this document, the following types of security information that a NSF needs to provide are considered:

- o The alarm triggered by a certain status in NSF
- o The alarm triggered by a certain security event
- o The report about the security events occurred in a certain period
- o The report about the status of NSF in a certain period

2. Terminology

2.1. Key Words

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].

2.2. Definition of Terms

This document uses the terms defined in [I-D.draft-ietf-i2nsf-terminology].

3. Common Information

Some general information should be provided in each message sent from a NSF to the security controller who monitor it.

- o time_stamp: Indicate the time when the message is generated
- o vendor_name: The name of the NSF vendor
- o NSF_name: The name (or IP) of the device generatign the message
- o NSF_type: Inidcate what type the NSF is (e.g., firewall, WAF, IPS)
- o NSF_version: The software version of the NSF
- o module_name: Indicate the module for outputting alarms or reports
- o version: Indicate the version of the log format and is a two-digit decimal numeral starting from 01
- o log_type: Alarm, periodical report, etc
- o severity: Indicates the level of the logs. There are total eight levels, from 0 to 7. The smaller the numeral is, the higher the severity is. The details is: 0 - Emergency; 1 - Alert; 2 - Critical; 3 - Error; 4 - Warning; 5 - Notification; 6 - Informational; 7 - Debugging.

4. Alarm

An alarm is the message generated by a NSF. An alarm could be triggered by certain abnormal conditions occurred in a NSF (referred to as a System Alarm) or a detected network abnormal conditions (referred to as a Security Event Alarm).

4.1. System Alarm

4.1.1. Memory Alarm

The following information should be included in a Memory Alarm:

- o event_name: 'MEM_USAGE_HIGH'
- o usage: The usage rate of memory
- o threshold: The threshold triggering the event
- o message: 'The memory usage exceeded the threshold'

4.1.2. CPU Alarm

The following information should be included in a CPU Alarm:

- o event_name: 'CPU_USAGE_HIGH'
- o usage: The usage rate of CPU
- o threshold: The threshold triggering the event
- o message: 'The CPU usage exceeded the threshold'

4.1.3. DISK Alarm

The following information should be included in a Disk Alarm:

- o event_name: 'DISK_USAGE_HIGH'
- o usage: The usage rate of disk
- o threshold: The threshold triggering the event
- o message: 'The disk usage exceeded the threshold'

4.1.4. Session Table Alarm

The following information should be included in a Session Table Alarm:

- o event_name: 'SESSION_USAGE_HIGH'
- o current: The number of concurrent sessions
- o max: The maximum number of sessions that the session table can support
- o threshold: The threshold triggering the event
- o message: 'The number of session table exceeded the threshold'

4.1.5. Interface Alarm

The following information should be included in a Interface Alarm:

- o event_name: 'IFNET_STATE'
- o interface_Name: The name of interface

- o state: 'UP' or 'DOWN'
- o message: 'Current interface state'

4.2. Security Event Alarm

4.2.1. DDoS Alarm

The following information should be included in a DDoS Alarm:

- o event_name: 'SEC_EVENT_DDoS'
- o sub_attack_type: Any one of Syn flood, ACK flood, SYN-ACK flood, FIN/RST flood, TCP Connection flood, UDP flood, Icmp flood, HTTPS flood, HTTP flood, DNS query flood, DNS reply flood, SIP flood, and etc.
- o dst_ip: The IP address of a victum under attack
- o dst_port: The port numbers that the attack traffic aims at.
- o start_time: The time stamp indicating when the attack started
- o end_time: The time stamp indicating when the attack ended. If the attack is still undergoing when sending out the alarm, this field can be empty.
- o attack_rate: The PPS of attack traffic
- o attack_speed: the bps of attack traffic
- o rule_id: The ID of the rule being triggered
- o rule_name: The name of the rule being triggered
- o profile: Security profile that traffic matches.

4.2.2. Virus Alarm

The following information should be included in a Virus Alarm:

- o event_Name: 'SEC_EVENT_Virus'
- o virus_type: Type of the virus, e.g., trojan, worm, macro Virus type
- o virus_name

- o `dst_ip`: The destination IP address of the packet where the virus is found
- o `src_ip`: The source IP address of the packet where the virus is found
- o `src_port`: The source port of the packet where the virus is found
- o `dst_port`: The destination port of the packet where the virus is found
- o `src_zone`: The source security zone of the packet where the virus is found
- o `dst_zone`: The destination security zone of the packet where the virus is found
- o `file_type`: The type of the file where the virus is hided within
- o `file_name`: The name of the file where the virus is hided within
- o `virus_info`: The brief introduction of virus
- o `raw_info`: The information describing the packet triggering the event.
- o `rule_id`: The ID of the rule being triggered
- o `rule_name`: The name of the rule being triggered
- o `profile`: Security profile that traffic matches.

4.2.3. Intrusion Alarm

The following information should be included in a Intrustion Alarm:

- o `event_name`: The name of event: 'SEC_EVENT_Intrusion'
- o `sub_attack_type`: Attack type, e.g., brutal force, buffer overflow
- o `src_ip`: The source IP address of the packet
- o `dst_ip`: The destination IP address of the packet
- o `src_port`:The source port number of the packet
- o `dst_port`: The destination port number of the packet

- o src_zone: The source security zone of the packet
- o dst_zone: The destination security zone of the packet
- o protocol: The employed transport layer protocol, e.g.,TCP, UDP
- o app: The employed application layer protocol, e.g.,HTTP, FTP
- o rule_id: The ID of the rule being triggered
- o rule_name: The name of the rule being triggered
- o profile: Security profile that traffic matches
- o intrusion_info: Simple description of intrusion
- o raw_info: The information describing the packet triggering the event.

4.2.4. Botnet Alarm

The following information should be included in a Botnet Alarm:

- o event_name: the name of event: 'SEC_EVENT_Botnet'
- o botnet_name: The name of the detected botnet
- o src_ip: The source IP address of the packet
- o dst_ip: The destination IP address of the packet
- o src_port: The source port number of the packet
- o dst_port: The destination port number of the packet
- o src_zone: The source security zone of the packet
- o dst_zone: The destination security zone of the packet
- o protocol: The employed transport layer protocol, e.g.,TCP, UDP
- o app: The employed application layer protocol, e.g.,HTTP, FTP
- o role: The role of the communicating parties within the botnet:
 1. the packet from zombie host to the attacker
 2. The packet from the attacker to the zombie host

3. The packet from the IRC/WEB server to the zombie host
 4. The packet from the zombie host to the IRC/WEB server
 5. The packet from the attacker to the IRC/WEB server
 6. The packet from the IRC/WEB server to the attacker
 7. The packet from the zombie host to the victim
- o botnet_info: Simple description of Botnet
 - o rule_id: The ID of the rule being triggered
 - o rule_name: The name of the rule being triggered
 - o profile: Security profile that traffic matches
 - o raw_info: The information describing the packet triggering the event.

4.2.5. Web Attack Alarm

The following information should be included in a Web Attack Alarm:

- o event_name: the name of event: 'SEC_EVENT_WebAttack'
- o sub_attack_type: Concret web attack type, e.g., sql injection, command injection, XSS, CSRF
- o src_ip: The source IP address of the packet
- o dst_ip: The destination IP address of the packet
- o src_port: The source port number of the packet
- o dst_port: The destination port number of the packet
- o src_zone: The source security zone of the packet
- o dst_zone: The destination security zone of the packet
- o req_method: The method of requirement. For instance, 'PUT' or 'GET' in HTTP
- o req_url: Requested URL
- o url_category: Matched URL category

- o `filtering_type`: URL filtering type, e.g., Blacklist, Whitelist, User-Defined, Predefined, Malicious Category, Unknown
- o `rule_id`: The ID of the rule being triggered
- o `rule_name`: The name of the rule being triggered
- o `profile`: Security profile that traffic matches.

5. Reports

Different from Alarms, a report normally triggered by a timer or a request from the NE monitoring the device. So compared to alarms, a report contains more statical information.

5.1. Attack Report

5.1.1. DDoS Report

Besides the fields in an DDoS Alarm, the following information should be included in a DDoS Report:

- o `attack_type`: DDoS
- o `attack_ave_rate`: The average pps of the attack traffic within the recorded time
- o `attack_ave_speed`: The average bps of the attack traffic within the recorded time
- o `attack_pkt_num`: The number attack packets within the recorded time
- o `attack_src_ip`: The source IP addresses of attack traffics. If there are a large amount of IP addresses, then pick a certain number of resources according to different rules.
- o `action`: Actions against DDoS attacks, e.g., Allow, Alert, Block, Discard, Declare, Block-ip, Block-service.

5.1.2. Virus Report

Besides the fields in an Virus Alarm, the following information should be included in a Virus Report:

- o `attack_type`: Virus
- o `protocol`: The transport layer protocol

- o app: The name of the application layer protocol
- o times: The time of detecting the virus
- o action: The actions dealing with the virus, e.g., alert, block
- o os: The OS that the virus will affect, e.g., all, android, ios, unix, windows

5.1.3. Intrusion Report

Besides the fields in an Intrusion Alarm, the following information should be included in a Intrusion Report:

- o attack_type: Intrusion
- o times: The times of intrusions happened in the recorded time
- o os: The OS that the intrusion will affect, e.g., all, android, ios, unix, windows
- o action: The actions dealing with the intrusions, e.g., e.g., Allow, Alert, Block, Discard, Declare, Block-ip, Block-service
- o attack_rate: NUM the pps of attack traffic
- o attack_speed: NUM the bps of attack traffic

5.1.4. Botnet Report

Besides the fields in an Botnet Alarm, the following information should be included in a Botnet Report:

- o attack_type: Botnet
- o botnet_pkt_num: The number of the packets sent to or from the detected botnet
- o action: The actions dealing with the detected packets, e.g., Allow, Alert, Block, Discard, Declare, Block-ip, Block-service, etc
- o os: The OS that the attack aiming at, e.g., all, android, ios, unix, windows, etc.

5.1.5. Web Attack Report

Besides the fields in an Web Attack Alarm, the following information should be included in a Web Attack Report:

- o attack_type: Web Attack
- o rsp_code: Response code
- o req_clientapp: The client application
- o req_cookies: Cookies
- o req_host: The domain name of the requested host
- o raw_info: The information describing the packet triggering the event.

5.2. Service Report

5.2.1. Traffic Report

Traffic reports provide visibility into traffic signatures, bandwidth usage, and how the configured security and bandwidth policies have been applied.

- o src_zone: Source security zone of traffic
- o dst_zone: Destination security zone of traffic
- o src_region: Source region of the traffic
- o dst_region: Destination region of the traffic
- o src_ip: Source IP address of traffic
- o src_user: User who generates traffic
- o dst_ip: Destination IP address of traffic
- o src_port: Source port of traffic
- o dst_port: Destination port of traffic
- o protocol: Protocol type of traffic
- o app: Application type of traffic

- o policy_id: Security policy id that traffic matches
- o policy_name: Security policy name that traffic matches
- o in_interface: Inbound interface of traffic
- o out_interface: Outbound interface of traffic
- o total_traffic: Total traffic volume
- o in_traffic_ave_rate: Inbound traffic average rate in pps
- o in_traffic_peak_rate: Inbound traffic peak rate in pps
- o in_traffic_ave_speed: Inbound traffic average speed in bps
- o in_traffic_peak_speed: Inbound traffic peak speed in bps
- o out_traffic_ave_rate: Outbound traffic average rate in pps
- o out_traffic_peak_rate: Outbound traffic peak rate in pps
- o out_traffic_ave_speed: Outbound traffic average speed in bps
- o out_traffic_peak_speed: Outbound traffic peak speed in bps.

5.2.2. Policy HIT Report

Policy HIT reports record the security policy that traffic matches and its hit count. It can check if policy configurations are correct.

- o src_zone: Source security zone of traffic
- o dst_zone: Destination security zone of traffic
- o src_region: Source region of the traffic
- o dst_region: Destination region of the traffic
- o src_ip: Source IP address of traffic
- o src_user: User who generates traffic
- o dst_ip: Destination IP address of traffic
- o src_port: Source port of traffic

- o dst_port: Destination port of traffic
- o protocol: Protocol type of traffic
- o app: Application type of traffic
- o policy_id: Security policy id that traffic matches
- o policy_name: Security policy name that traffic matches
- o hit_times: The hit times that the security policy matches the specified traffic.

5.2.3. DPI Report

DPI reports provide statistics on uploaded and downloaded files and data, sent and received emails, and alert and block records on websites. It's helpful to learn risky user behaviors and why access to some URLs is blocked or allowed with an alert record.

- o type: DPI action types. e.g., File Blocking, Data Filtering, Application Behavior Control
- o file_name: The file name
- o file_type: The file type
- o src_zone: Source security zone of traffic
- o dst_zone: Destination security zone of traffic
- o src_region: Source region of the traffic
- o dst_region: Destination region of the traffic
- o src_ip: Source IP address of traffic
- o src_user: User who generates traffic
- o dst_ip: Destination IP address of traffic
- o src_port: Source port of traffic
- o dst_port: Destination port of traffic
- o protocol: Protocol type of traffic
- o app: Application type of traffic

- o policy_id: Security policy id that traffic matches
- o policy_name: Security policy name that traffic matches
- o action: Action defined in the file blocking rule, data filtering rule, or application behavior control rule that traffic matches.

5.2.4. Vulnerability Scanning Report

Vulnerability scanning reports record the victim host and its related vulnerability information that should to be fixed. the following information should be included in the report:

- o victim_ip: IP address of the victim host which has vulnerabilities
- o vulnerability_id: The vulnerability id
- o vulnerability_level: The vulnerability level. e.g., high, middle, low
- o OS: The operating system of the victim host
- o service: The service which has vulnerability in the victim host
- o protocol: The protocol type. e.g., TCP, UDP
- o port: The port number
- o vulnerability_info: The information about the vulnerability
- o fix_suggestion: The fix suggestion to the vulnerability.

5.2.5. User Activity Report

User activity reports provide visibility into users' online records (such as login time, online/lockout duration, and login IP addresses) and the actions users perform. User activity reports are helpful to identify exceptions during user login and network access activities.

- o user: Name of a user
- o group: Group to which a user belongs
- o login_ip_address: Login IP address of a user
- o authentication_mode: User authentication mode. e.g., Local Authentication, Third-Party Server Authentication, Authentication Exemption, SSO Authentication

- o access_mode: User access mode. e.g., PPP, SVN, LOCAL
- o online_duration: Online duration
- o lockout_duration: Lockout duration
- o type: User activities. e.g., Succeeded User Login, Failed User Login, User Logout, Succeeded User Password Change, Failed User Password Change, User Lockout, User Unlocking, Unknown
- o cause: Cause of a failed user activity

5.3. System Report

5.3.1. Operation Report

Operation reports record administrators' login, logout, and operations on the device. By analyzing them, security vulnerabilities can be identified. The following information should be included in operation report:

- o Administrator: Administrator that operates on the device
- o login_ip_address: IP address used by an administrator to log in
- o login_mode: Mode in which an administrator logs in
- o operation_type: The operation type that the administrator execute, e.g., login, logout, configuration, etc
- o result: Command execution result
- o content: Operation performed by an administrator after login.

5.3.2. Running Report

Running reports record the device system's running status, which is useful for device monitoring. The following information should be included in running report:

- o system_status: The current system's running status
- o CPU_usage: The usage rate of CPU
- o memory_usage: The usage rate of memory
- o disk_usage: The usage rate of disk

- o disk_left: The left space of disk
- o session_number: The concurrent sessions' number
- o process_number: The number of system process
- o in_traffic_rate: The total inbound traffic rate in pps
- o out_traffic_rate: The total outbound traffic rate in pps
- o in_traffic_speed: The total inbound traffic speed in bps
- o out_traffic_speed: The total outbound traffic speed in bps

6. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

7. Security Considerations

TBD

8. Acknowledgements

9. References

9.1. 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>>.

9.2. Informative References

- [I-D.ietf-i2nsf-framework] elopez@fortinet.com, e., Lopez, D., Dunbar, L., Strassner, J., Zhuang, X., Parrott, J., Krishnan, R., and S. Durbha, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-01 (work in progress), June 2016.

[I-D.xia-i2nsf-capability-interface-im]

Xia, L., Zhang, D., elopez@fortinet.com, e., Bouthors, N.,
and L. Fang, "Information Model of Interface to Network
Security Functions Capability Interface", draft-xia-i2nsf-
capability-interface-im-05 (work in progress), March 2016.

Authors' Addresses

Dacheng Zhang
Aliababa Group

Email: dacheng.zdc@alibaba-inc.com

Yi Wu
Aliababa Group

Email: anren.wy@alibaba-inc.com

Liang Xia
Huawei

Email: frank.xialiang@huawei.com