

Network Working Group
Internet Draft
Intended status: Informational
Expires: November 2015

L. Dunbar
Huawei
M. Zarny
Goldman Sachs
C. Jacquenet
M. Boucadair
France Telecom
S. Chakrabarty
US Ignite

May 28, 2015

Interface to Network Security Functions (I2NSF) Problem Statement
draft-dunbar-i2nsf-problem-statement-05.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 November 28, 2015.

Copyright Notice

Copyright (c) 2015 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 motivation and the problem statement for Interface to Network Security Functions (I2NSF).

Table of Contents

1. Introduction.....	3
2. Requirements Language.....	4
3. Problem Space.....	5
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....	7
3.1.4. More Distributed NSFs and vNSFs.....	7
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
3.2. Challenges Facing Customers.....	9
3.2.1. NSFs from heterogeneous administrative domains.....	9
3.2.2. Today's Control Requests are Vendors Specific.....	9

3.2.3. Difficulty to Monitor the Execution of Desired Policies	11
3.3. Difficulty to Validate Policies across Multiple Domains..	11
3.4. Lack of Standard Interface to Inject Feedback to NSF.....	12
3.5. Lack of Standard Interface for Capability Negotiation....	12
4. Scope of the proposed work.....	12
5. Other Potential Uses of I2NSF.....	14
6. Related Industry Initiatives.....	14
6.1. Related IETF WGs.....	14
6.2. Relationship with ETSI NFV ISG.....	16
6.3. OpenStack Firewall/Security as a Service.....	16
6.4. Security as a Service by Cloud Security Alliance.....	17
7. Manageability Considerations.....	17
8. Security Considerations.....	17
9. IANA Considerations.....	17
10. References.....	17
10.1. Normative References.....	17
10.2. Informative References.....	17
11. Acknowledgments.....	19
11.1. Appendix: Relationship with Open Source Communities.....	20

1. Introduction

This document describes the motivation and the problem space for the Interface to Network Security Functions (I2NSF) effort.

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, 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 the result, the Network security functions (NSFs) are provided and consumed in increasingly diverse environments. Users of NSFs could 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 does not elaborate on specific use case. The reader should refer to [I2NSF-ACCESS], [I2NSF-DC] and [I2NSF-Mobile] for a more in-depth discussion on the I2NSF use cases.

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].

This document makes use of the following terms and acronyms:

DC: Data Center

Network Security Function (NSF): functions to ensure integrity, confidentiality and availability of network communications, to detect unwanted activity, and to block it or at least mitigate its effects on the network.

Hosted security function: Refers to a security function that it is hosted by another network.

Flow-based Network Security Function: A function that inspects network flows according to a policy 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 (MAC rewrites, TTL decrement action; even NAT would be outside the inspection process).

3. Problem Space

The following sub-sections describe the problems and challenges facing customers and security service providers (called service provider, for short) when security functions are no longer physically hosted by customer's administrative domain.

The "Customer-Provider" relationship may be between any two parties: 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 procedure, alarms reporting, etc. There is currently no standard mechanism to capture those requirements.

Note 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 & Attack Protection:

e.g., Firewall/ACL; Authentication; IPS; IDS; Endpoint Protection; etc;

Security Functions in a DMZ:

e.g., Firewall/ACL; IDS/IPS, authentication and authorization services, NAT, forward proxies, application FWs, AAA; etc.

Internal Security Analysis & report:

e.g., Security Log; Event Correlation; Forensic Analysis; etc;

Internal Data and Content Protection:

e.g., Encryption; Authorization; Public/Private key management for internal database, etc.

Given the diversity of security functions, contexts in which they can be deployed, and constant evolution of these functions, standardizing all aspects of security functions is challenging, most probably not feasible, and not necessary. For example, from an I2NSF perspective, there is no need to standardize on how a firewall filters are created or applied. What is needed is having an interface to control and monitor the behavior of NSFs.

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.

Yet because no widely accepted industry standard security interfaces exist today, management of NSFs (device and policy provisioning, monitoring, etc.) tends to be bespoke, essentially as offered by product vendors. As a result, automation of such services, if it exists at all, is also bespoke. It is worth noting that even with the traditional way of deploying security features, there is still a gap to coordinate among implementations from distinct vendors. This is mainly the reason why mono-vendor security functions are enabled in a given network segment.

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. As such, 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 that 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, network security functions (vNSF).

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

3.1.5. More Demand to Control NSFs Dynamically

In the advent of SDN [SDN-Security], more clients, applications or application controllers need to dynamically update their communication policies that are enforced by NSFs. The Security Service Providers have to dynamically update control requests to NSFs upon receiving the requests from their clients.

3.1.6. Demand for multi-tenancy to control and monitor NSFs.

Service providers may require having 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 manufactured by multiple

vendors. Even within one product category (e.g., firewall), security functions provided by different vendors can have different features and capabilities: filters that can be designed and activated by a firewall may or may not support IPv6, depending on the firewall technology, for example.

Service Provider management system (or controller) needs ways to retrieve the capabilities of service functions by different vendors so that it could build an effective security solution.

These capabilities can be documented in a static manner or via an interface for security functions vendors to register to service provider security management system. This 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 can't automate the process of selecting NSFs by different vendors to accommodate customer's requirements.

3.1.8. Lack of mechanism for NSFs to utilize external profiles

Many security functions depend on signature files or profiles to perform, e.g. IPS/IDS Signatures. Different policies might need different signatures or profiles. Today, most vendors have their vendor specific signatures or profiles. As the industry moves towards more open environment, sharing profile or black database can be 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 NSFs to use external profiles.

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 security skills. As such, they may not be able to express sufficiently precise requirements or security policies. Usually these customers express expectations (that can be viewed as loose security requirements). Customers may also express guidelines such as which critical communications are to be preserved during critical events, which hosts are to service even during severe security attacks, etc.

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 use. They are looking for combining 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 can consist of NSFs on customer networks and NSFs on service provider networks.

3.2.2. Today's Control Requests are Vendors Specific

Customers may consume NSFs by multiple service providers. Customers need to express their security requirements, guidelines, and expectations to the service providers, which in turn will be translated into security policies and associated configuration sets to the set of security functions. But no standard technical characterization and/or APIs exist, even for most common security

services. Most security services are accessible only through disparate, proprietary interfaces (e.g., portals, APIs), in whatever format vendors choose to offer.

Without standard interfaces it is complex for customers to update security policies and integrate with services provided by the security service providers. This complexity is induced by the diversity of the configuration models, policy models, supported management interfaces, etc.

The current practices that rely on the use of scripts that generates automatically scripts have to be adjusted each time an implementation from a different vendor is enabled in a provider side.

Customers may also require means to easily update/modify their security requirements with immediate effect in the underlying involved NSFs.

While security agreements are in place, security functions may be solicited without requiring an explicit invocation means. Nevertheless, some explicit invocation means may be required to interact with a service function.

Here is an example of how standard interfaces could help achieve faster implementation time cycles. Let us consider a customer who would like to dynamically allow an encrypted flow with specific port, src/dst addresses or protocol type through the firewall/IPS to enable an encrypted video conferencing call only during the time of the call. With no commonly accepted interface in place, the customer would have to learn about the particular provider's firewall/IPS interface, and send the request in the provider's required format. If a firewall/IPS interface standard exists, the customer would be able to send the request, without having to do much preliminary legwork. Such a standard helps providers too since they could now offer the same firewall/IPS interface to represent firewall/IPS services, which may be offered by different vendors' products. They have now abstracted the firewall/IPS services. Lastly, it helps the firewall/IPS vendors since they could now work on common specifications.

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 is the result of the execution of their desired security requirements, guidelines and expectations.

Today, there is no standard way for customers to get security service assurance (including running "what-if" scenarios to assess the efficiency of the delivered security service) of their specified security policies properly enforced by the security functions in the provider domain.

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 to have a standard interface to express, monitor and verify security policies that combine several distributed security functions. This becomes more crucial when NSFs are instantiated in Virtual Machines because NSFs can be more distributed and sometimes multiple NSFs are combined together to perform one task.

Without standard interfaces and security policy data models, the enforcement of a customer-driven security policy remains challenging because of the inherent complexity brought by the combined invocation of several, yet vendor-specific security functions, but also because of the accompanying complexity of configuration procedures and operational tasks in a multi-vendor, heterogeneous environment.

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

3.4. Lack of Standard Interface to Inject Feedback to NSF

Today, many security functions, such as IPS 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. [CA] 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 NSFs selected can't perform the policies from the Security Controller, due to resource constraints. To support the automatic control in the SDN-era, it is necessary to have a set of messages for proper negotiation between the Security Controller and the NSFs.

4. Scope of the proposed work

The primary goal of I2NSF is to define an information model, a set of software interfaces and data models for controlling and monitoring aspects of physical and virtual NSFs. Other aspects of NSFs, such as device or network provisioning and configuration, are out of scope. Controlling and monitoring of NSFs should include the ability to specify, query, monitor, and control the NSFs by one or more management entities. Since different security vendors support different features and functions on their devices, I2NSF will focus on flow-based NSFs that provide treatment to packets/flows, such as IPS/IDS, Web filtering, flow filtering, deep packet inspection, or pattern matching and remediation.

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

- The I2NSF Capability Layer specifies how to control and monitor NSFs at a functional implementation level. That is, I2NSF will standardize a set of interfaces by which control and management of NSFs may be invoked, operated, and monitored. (I2NSF will not work on any other aspects of NSFs. Nor will I2NSF at this stage specify how to derive control and monitoring capabilities from higher level security policies for the Capability Layer.)
- The I2NSF Service Layer defines how clients' security policies may be expressed and monitored. The Service Layer is out of scope for this phase of I2NSF's work. However, I2NSF will provide a forum for Informational drafts on data models, APIs, etc. that demonstrate how service layer policies may be translated to Capability Layer functions.

The concrete work at the I2NSF Capability Layer includes development of

- An information model that defines concepts required for standardizing the control and monitoring of NSFs.
- A set of YANG data models, derived from the above information model.
- The capability registry (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. The registry enables various mechanisms, including policy rules, to be used to match monitor and control functions to the needs of an application and/or environment.
- The proper secure communication channels to carry the controlling and monitoring information between the NSFs and their management entity (or entities).

Standard interfaces for monitoring and controlling the behavior of NSFs are essential building blocks for Security Service Providers to automate the use of different NSFs from multiple vendors by their Security management entities. This work will leverage the existing protocols and data models defined by I2RS, Netconf, and NETMOD.

I2NSF may be invoked by any (authorized) client-e.g., upstream applications (controllers), orchestration systems, security portals, etc.

5. Other Potential Uses of I2NSF

The I2NSF framework allows the clients to view, request, and/or verify the security functions/policies offered by providers at different premises. This framework can make it possible for a cluster of devices requiring the similar security policies to have consistent policies across multiple sites.

Network service providers can provide "Hosted Security Functions" services. Network providers can also act as security function brokers to facilitate if not optimize the enforcement of customer-driven security policies. They can expose a service catalog and standard mechanisms by which enterprises (or applications) can query, request, or/and verify the needed security functions or policies.

With the standard interfaces for clients to request the required security functions and policies, network operators can leverage their current service to enterprises (e.g. VPN, private IP services) and access to a vast population of end users to offer a set of consolidated Security solutions and policies. Network operators can be instrumental in defining a common interface and framework as part of an IETF-conducted specification effort.

6. Related Industry Initiatives

6.1. Related IETF WGs

IETF NETCONF: I2NSF should consider using the NETCONF protocol exchange security policy provisioning information between participating devices/security functions and the computation logic (a.k.a., a security Policy Decision Point (PDP)) that resides in the control plane and which makes the decisions to dynamically allocate resources and enforce customer-driven security policies.

NETMOD ACL Model: [I-D.ietf-netmod-acl-model] describes the very basic attributes for access control. I2NSF will extend the ACL data model to be more comprehensive, for example, extend to multiple actions and policies, and describes various services associated with the security functions under consideration.

In addition, I2NSF has to specify ways to monitor/report of Packet Based Security Functions.

I2RS: the WG currently discusses the specification of an interface between the forwarding and the control planes, to facilitate the dynamic enforcement of traffic forwarding policies based upon IGP/BGP route computation results. I2NSF is looking specifically into expressing security policies in two layers. I2NSF should leverage the protocols and data models developed by I2RS.

I2NSF aims to develop the additional information models and data models for distributed security functions, like the firewall and IPS/IDS. The policy structure specified by [I-D.hares-i2rs-bnp-info-model] can be used by I2NSF to be extended to include recursive actions to other security functions.

The IETF SFC WG specifies service function chaining techniques while treating service functions as a black box; VNFpool is about the reliability and availability of the virtualized network functions. But neither addresses how service functions are invoked, or configured.

Both SFC and VNFpool do not cover in-depth specification (e.g. rules for the requested FW) to invoke security functions. In SFC and VNFpool, a firewall function is a black box that is treated in the same way as a video optimization function. SFC and VNFpool do not cover the negotiation part, e.g. Client needs Rules x/y/z for FW, but the Provider can only offer x/z.

The IETF SACM (Security Assessment and Continuous Monitoring) WG specifies mechanisms to assess endpoint security. The endpoints can be routers, switches, clustered DB, or an installed piece of software. SACM is about "How to encode that policy in a manner where assessment can be automated". For example:

- a Solaris 10 SPARC or Windows 7 system used in an environment that requires adherence to a policy of Mission Critical Classified,
- rules like "The maximum password age must be 30 days" and "The minimum password age must be 1 day"

[I2NSF-GAP] has a more extensive study comparing I2NSF with various existing efforts in similar/adjacent areas.

6.2. Relationship with ETSI NFV ISG

ETSI's NFV ISG defines the architecture to pool together many virtual network functions to be managed and consumed collectively.

I2NSF is one of the enabling tools for NFV, specifically the VNF as a Service (VNFaaS) specified by ETSI NFV Group Specification Use Cases [gs_NFV].

ETSI's NFV ISG effort is actively contributed by service providers. It defines a detailed service model for VNFaaS as well as requirements that should be taken into account by the I2NSF initiative.

6.3. OpenStack Firewall/Security as a Service

Open source projects like OpenStack and CloudStack have begun to tackle the issues of interfaces to security functions but much work remains.

OpenStack completed the Firewall as a Service project and specified the set of APIs for Firewall services [API]

OpenStack has defined the APIs for managing Security Groups [SG]

The attributes defined by OpenStack Firewall/Security as a Service are at this point are basic. However, they can serve as the basis of the information model that the I2NSF IETF initiative aims to specify.

6.4. Security as a Service by Cloud Security Alliance

https://cloudsecurityalliance.org/research/secaas/#_get-involved

SaaS by CSA is at the initial stage of defining the scope of work.

7. Manageability Considerations

Management of NSFs usually include configuration of devices, signaling and policy provisioning. I2NSF will only focus on the policy provisioning part.

8. Security Considerations

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

9. IANA Considerations

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

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.

10.2. Informative References

[SG] http://docs.openstack.org/admin-guide-cloud/content/securitygroup_api_abstractions.html

[API] http://docs.openstack.org/admin-guide-cloud/content/fwaas_api_abstractions.html

[CA] <http://cyberthreatalliance.org/>

- [I-D.hares-i2rs-bnp-info-model] Hares, S., Wu, Q., Tantsura, J., and R. White, "An Information Model for Basic Network Policy and Filter Rules", draft-hares-i2rs-bnp-info-model-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-opsawg-firewalls] Baker, F. and P. Hoffman, "On Firewalls in Internet Security", draft-ietf-opsawg-firewalls-01 (work in progress), October 2012.
- [RFC7297] Boucadair, M., "IP Connectivity Provisioning Profile", RFC7297, April 2014.
- [I2NSF-PACKET] E. Lopez, "Packet-based Paradigm for Interfaces to NSFs", <draft-lopez-i2nsf-packet-00>, March 2015.
- [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.
- [SDN-Security] J. Jeong, et al, "Requirement for Security Services based on Software-Defined Networking", <draft-jeong-i2nsf-sdn-security-services-01>, March 2015.
- [I2NSF-GAP] D. Zhang, et al, "Analysis of Existing Work for I2NSF", <draft-zhang-gap-analysis-00>, Feb 2015.

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

[Gartner-2013] E. Messmer, "Gartner: Cloud-based security as a service set to take off", Network World, 31 October 2013

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

[Application-SDN] J. Giacomoni, "Application Layer SDN", Layer 123 ONF Presentation, Singapore, June 2013

11. Acknowledgments

Acknowledgments to Diego Lopez, Ed Lopez, Andy Malis, John Strassner, and many others for review and contribution to the content.

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

Authors' Addresses

Linda Dunbar
Huawei Technologies
5340 Legacy Drive, Suite 175
Plano, TX 75024, USA
Phone: (469) 277 5840
Email: ldunbar@huawei.com

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

Shaibal Chakrabarty
US Ignite
1776 Massachusetts Ave NW, Suite 601
Washington, DC 20036
Phone: (214) 708 6163
Email: shaibalc@us-ignite.org

11.1. Appendix: Relationship with Open Source Communities

One of the goals of the I2NSF initiative is to form a collaborative loop from IETF to Industry Open Source Communities.

Open-source initiatives are not to be considered as an alternative to formal standardization processes. On the contrary, they are complementary, with the former acting as an enabler and accelerator of the latter. Open-source provides an ideal mechanism to quick prototyping and validating contending proposals, and demonstrating the feasibility of disruptive ideas that could otherwise not be considered. In this respect, open-source facilitates the engagement in the standardization process of small (and typically more dynamic) players such as start-ups and research groups, which would see better opportunities of being heard and a clearer rewards to their efforts. An open-source approach is extremely useful as well for the production of open reference implementations of the standards at the same (or even faster) pace they are defined. The availability of such reference implementations translate into much simpler interoperability and conformance assessments for both providers and users, and can become the basis for incremental differentiation of a common solution, thus allowing a cooperative competition ("coopetition") model.

INTERNET-DRAFT
Intended Status: Standards Track
Expires: September 21, 2016

Luyuan Fang
Deepak Bansal
Microsoft

March 21, 2016

Inter-Cloud DDoS Mitigation API
draft-fang-i2nsf-inter-cloud-ddos-mitigation-api-01

Abstract

This document defines an Inter-Cloud DDoS Mitigation Abstract Layer and corresponding standardized APIs to enable the exchange of real time automated information to enable DDoS mitigation across Cloud Service Providers and Network Service Providers.

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/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

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. Terminology	3
2. Problem Statement	4
3. Inter-Cloud DDoS Mitigation Layer	5
4. Inter-Cloud DDoS Mitigation API	7
4.1. Categories of Inter-cloud API	7
4.1.1. Capability information exchange:	8
4.1.2. Mitigation Request and response:	8
4.1.3. Monitoring and Reporting:	8
4.1.4. Knowledge sharing:	8
4.2. REST API format	8
4.2.1. Capability	8
4.2.1.1. GET	8
4.2.2. Mitigation	9
4.2.2.1. POST	9
4.2.2.2. GET	9
4.2.2.3. PUT	9
4.2.2.4. DELETE	9
4.2.3. Monitor & Reporting	10
4.2.3.1. POST	10
4.2.3.2. GET	10
4.2.3.3. PUT	10
4.2.3.4. DELETE	10
4.2.3.5. GET	11
4.2.4. Knowledge Sharing	11
4.2.4.1. GET	11
5. Security Considerations	11
6. IANA Considerations	11
7. References	11
7.1. Normative References	11
7.2. Informative References	12
Authors' Addresses	12
Contributing Authors' Addresses	12

1. Introduction

We recently observe the following characteristics of the DDoS attacks in the Cloud era: 1) Growing in volume: for example, 450 Gbps peak speed DDoS attack in an ISP network was observed in December 2014, while over 300 Gbps DDoS attack was reported in 2013; 2) Growing in frequency; 3) Using Cloud services to launch major attacks, especially when some cloud services do not impose bandwidth and compute resource limitation; 4) Growing in sophistication: leverage vulnerable services like NTP, DNS, and BitTorrent to amplify the available bandwidth; 5) Growing attack to Inter-cloud/Inter-provider connection links, large volume attack can disrupt all cloud services traversing through the inter-connection links.

This draft is focus on Inter-Cloud/Inter-provider DDoS attack mitigation. The fast growth in volume and scale of Distributed Denial of Service (DDoS) attacks, particularly its impact on the large pipes of Inter-Cloud, Inter-Provider connections, calls for mechanisms to enable DDoS mitigation across Cloud Service Providers (CSPs) and Network Service Providers (NSPs). These mechanisms require to define an Inter-Cloud DDoS Mitigation Abstract Layer with corresponding standardized APIs to allow real time, automated information exchange among CSPs and NSPs, and achieve rapid protective response and effective Inter Cloud/Inter Provider DDoS attack mitigation. The need for such standard Inter-Cloud DDoS Mitigation APIs is strong and urgent.

This document defines the Inter-Cloud DDoS Mitigation Abstract Layer and APIs.

This document focuses on Inter-Cloud, Inter-Provider automated exchange of DDoS Mitigation information, although similar APIs could be used within each cloud for handling malicious traffic.

1.1. 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 [RFC2119].

This document uses the terminology defined in [I-D.draft-ietf-i2nsf-gap-analysis].

In addition, this document uses the following terms.

Term	Definition
-----	-----
BGP	Border Gateway Protocol
CSP	Cloud Service Provider
DC	Data Center
DCI	Data Center Interconnect
DDoS	Distributed Denial of Service
DLC	Disruption Life Cycle
Inter-Cloud	The interconnection between the cloud of different providers
NSP	Network Service Provider
SDN	Software Defined Network
SVR	Server

2. Problem Statement

Along with the rapid growth of cloud services, the large pipes of Inter-Cloud, Inter-Provider connections are increasingly the subject of DDoS attacks. Since these connections are between clouds of different providers, implementing mechanism to achieve rapid protective response in case of attack is challenging. While within its own cloud each provider may be able to protect effectively its network using various DDoS protection techniques, for the Inter-Cloud/Inter-Provider links, each provider does not have full visibility of the attack, and therefore response times may be longer, counter-measures may be less effective, and therefore the severity and impact of the attacks may be very significant.

Large DDoS attacks targeting the Inter-Cloud, Inter-Provider links may consume the available bandwidth or the router/switch/server resources within tens of seconds. While the attack is on, legitimate traffic is prevented from being forwarded over the saturated links. With saturated Inter-Cloud, Inter-Provider links, even if within each cloud the DDoS mitigation may be working effectively, it can quickly be rendered irrelevant.

How does Distributed DoS attack relate to Inter-Cloud connections? The DDoS attack can be targeting the hosts, servers, end-points, gateways, or any devices in between. Regardless of the target, the attack traffic flows through the "Pipes"/inter-connection links, and can saturate these large pipes. Attack volume is the key issue here. DDoS attack BW is increasing very fast in the recent years. Attack BW greater than 100G is not uncommon any more, and 450G peak speed DDoS attack has been seen in some SP networks end of 2014. The DDoS attack can consume BW, impact multi-region Data Centers and Inter-Cloud connectivity, and interrupt multi-services. Because of its massive scale, it can also make fast mitigation more challenging.

Today, exchange of DDoS attack information and mitigation strategy among providers is largely manual and typically relies on customized operation processes established ad hoc between each provider. Manual means someone has to send emails, or make phone calls to reach the people in another Cloud, another ISP, etc. No signaling, no common API, no automation across the provider boundaries available. Because of largely manual escalation procedures, providers' reaction times to DDoS attacks to Inter-Cloud, Inter-Provider links tends to be slow (it can easily take tens of minutes if not hours to put effective mitigation measures in place) compared to Intra-Cloud DDoS mitigation, and thus the damage caused by such attacks can be substantial. The reaction time may exceed the Disruption Life Cycle (DLC) of the attack.

Sophisticated and determined malicious attackers are able to quickly learn the intended Inter-Cloud Inter-Provider link capabilities and limitations through probing. This includes bandwidth capacity, saturation resistance - the attack cannot saturate the connection links and make them unusable, and DDoS absorption resilience of the link - the attack can be absorbed without taking down the network connections and impact the services. The attacker is also able to learn the DDoS countermeasures and their response times, from which the attacker can infer the DLC that can be exacted toward the intended target. The DLC is measured by the assailant from the time the attack is initiated to the time the mitigation response becomes evident. An attacker can then use this information to design the attacks in such a way that the current and subsequent attacks inflict the most harm.

In order to achieve rapid protective response, the exchange of DDoS mitigation information between providers must be enabled in real time and in an automated, standardized fashion.

3. Inter-Cloud DDoS Mitigation Layer

The Inter-Cloud DDoS Mitigation Layer and its corresponding standardized, secure Inter-Cloud DDoS Mitigation APIs is illustrated in Figure 1.

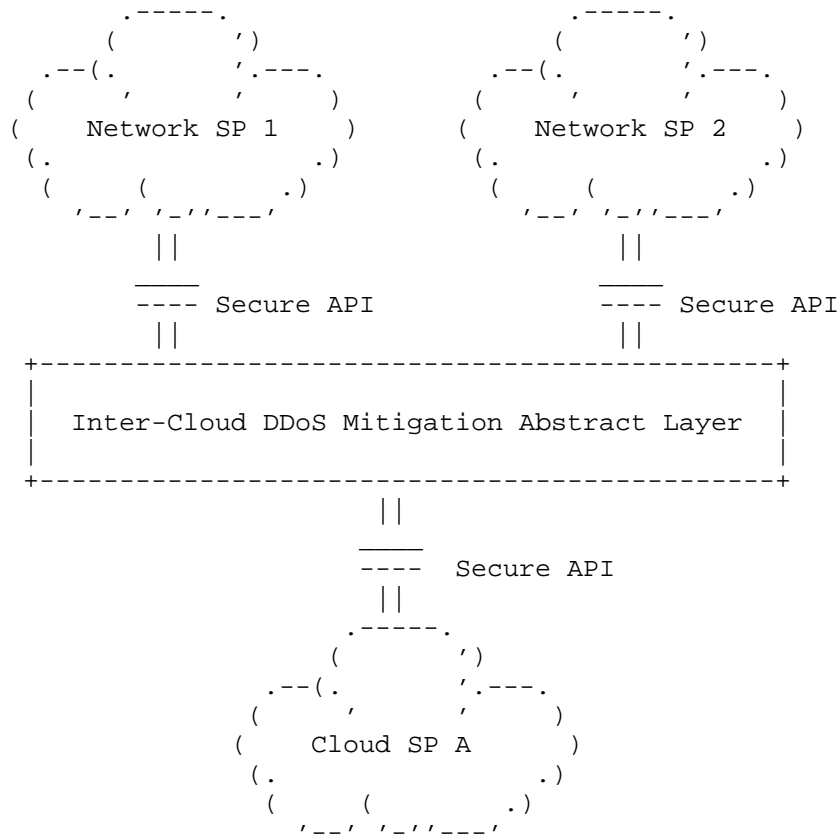


Figure 1. Inter-Cloud DDoS Mitigation Abstract Layer and APIs

Today there is no accepted industry common DDoS Mitigation Layer that can be used to reduce the reaction time and increase the effectiveness of mitigation in case of attack.

The Inter-Cloud DDoS Mitigation Abstract Layer provides standardized secure APIs that can be used by each provider to programmatically initiate real time information exchanges to other providers to provide visibility of the attack and coordinate DDoS mitigation mechanisms. Exchanged information may include signatures and forensic of the attack, timestamps, and black-holing countermeasures.

The Inter-Cloud DDoS Mitigation Abstract Layer provides corresponding API calls to exchange mitigation information on the following areas.

DDoS Protection Types:

- o TCP flood rate limiting
- o UDP flood rate limiting
- o TCP SYN.ACK/RST flood protection and authentication
- o Maximum concurrent connections per interval rate limiting
- o Maximum number of new connections allowed per interval rate limiting
- o Maximum fragment packets allowed per interval rate limiting
- o Maximum number of packets allowed per interval rate limiting
- o Black-holing
- o Use BGP Flowspec [RFC5575] to auto-coordinate traffic filtering, DDoS mitigation
- o Other BGP Signaling and Mitigation examples
 - o BGP /24 route advertisement with community string option
 - o Mitigation support for /32 with type and rate limit thresholds
 - o /32 removal from mitigation
 - o BGP support for /24 removal

Attack Lifecycle Monitoring and Reporting

- o Volume and scale of the attack, signatures, forensic
- o Timestamps

4. Inter-Cloud DDoS Mitigation API

4.1. Categories of Inter-cloud API

The following describe the basic functions the Inter-Cloud DDoS mitigation MUST support.

4.1.1.1. Capability information exchange:

Support "Query" the DDoS capabilities from one provider to another provider.

4.1.1.2. Mitigation Request and response:

Mitigation Request: One provider can "Request" for mitigation by partner provider based on pre-agreement.

Mitigation Response: The provider received DDoS mitigation request first acknowledge the request, then execute a particular DDoS capability on behalf of the requesting provider, and respond back with the logged actions performed and mitigation status.

4.1.1.3. Monitoring and Reporting:

Monitoring: Allow another provider to monitor DDoS status and mitigation processes.

Reporting: Provider DDoS status reports to partner providers.

4.1.1.4. Knowledge sharing:

Allow partner providers to query for a specific DDoS related data to enhance their DDoS resiliency and perform coordinate mitigation when possible.

4.2. REST API format

4.2.1. Capability

Definition: A participating provider should allow another provider to query for its DDoS capabilities.

The following REST API are the basic ones that every provider participating MUST provide.

4.2.1.1. GET

Example 1: GET (DDoS mitigation Capabilities)

a. Description: The receiving provide returns a list of DDoS mitigation it can perform

b. Parameters: None

c. Responses: 200, an array of mitigation objects format.

Example 2: GET (DDoS mitigation Capabilities - protocol)

a. Description: Return a list of DDoS mitigation that this provider can perform for the protocol specified.

b. Parameters: protocol is one of the following strings {tcp, udp, dns}

c. Responses: 200, OK, an array of mitigation objects format.

(more details to be added especially around format of the object to be returned).

4.2.2. Mitigation

Definition: Mitigation Request and Response must be supported between participating providers for executing a particular DDoS capability.

The following REST API are the baselines that each participating providers MUST support.

4.2.2.1. POST

a. Description: Create a new policy what will cause a mitigation to be performed based on a specific trigger.

b. Parameters: PolicyObject {To be specified}

c. Responses: 200, OK, return an identifier.

4.2.2.2. GET

a. Description: Get an existing policy.

b. Parameters: id identifier of the policy that was created.

c. Responses: 200, OK, return the policy of the specified id

4.2.2.3. PUT

a. Description: Update a particular policy.

b. Parameters: PolicyObject {To be specified} & id which is the identifier which was returned after a successful create of a policy.

4.2.2.4. DELETE

a. Description: Delete a policy and therefore end any mitigation that is currently active.

b. Parameters: id the identifier of the policy that was created.

c. Response: 200, OK, policy deleted.

4.2.3. Monitor & Reporting

Definition: A participating provider MUST allow another provider to monitor a particular DDoS mitigation.

The following REST API are the basic ones that every provider must provide.

4.2.3.1. POST

a. Description: Created a new monitored object for a policy/mitigation.

b. Parameter: MonitoredObject {To be specified} & id which is the mitigation identifier. The MonitoredObject will have parameter to enable retrieving sFlow to a particular endpoint for collection of the metrics. By default, you can use REST API calls as defined below to retrieve monitored objects stats.

c. Responses: 200, OK

4.2.3.2. GET

a. Description: Get the current monitoring settings for this mitigation/policy.

b. Parameter: id identifier for the mitigation/policy.

c. Responses: 200, OK, monitoring settings

4.2.3.3. PUT

a. Description: Update the current monitoring settings for this mitigation/policy

b. Parameter: id identifier for the mitigation/policy.

c. Responses: 200, OK

4.2.3.4. DELETE

a. Description: Remove all monitoring configuration for this mitigation/policy

b. Parameter: id identifier for the mitigation/policy.

c. Responses: 200, OK

4.2.3.5. GET

- a. Description: Return the stats available for this mitigation and monitored object.
- b. Parameters: None
- c. Responses: 200, OK, stats

4.2.4. Knowledge Sharing

Definition: A participating provider MUST allow another participating provider to query for a specific DDoS related data to enhance their DDoS resiliency.

The following REST API are the basic ones that every provider must provide.

4.2.4.1. GET

- a. Description: Return the current blacklist.
- b. Parameter: Size to limit the returned list.
- c. Responses: 200, OK, return a string array of blacklisted IPs.

5. Security Considerations

Given the subject of the draft is Inter-Cloud/Inter-Provider DDoS mitigation, security policies among the participating providers must be agreed upon and strictly followed. Authentication MUST be enforced on all interconnections and APIs in discussion.

6. IANA Considerations

None.

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.

[RFC5575] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J., and D. McPherson, "Dissemination of Flow Specification Rules," RFC

5575, August 2009.

7.2. Informative References

[I-D.draft-ietf-i2nsf-gap-analysis] S. Hares et al., "Analysis of Use Cases and Gaps in Technology for I2NSF ",draft-ietf-i2nsf-gap-analysis-00.txt, Feb. 2016.

Authors' Addresses

Luyuan Fang
Microsoft
15590 NE 31st St
Redmond, WA 98052
Email: lufang@microsoft.com

Deepak Bansal
Microsoft
15590 NE 31st St
Redmond, WA 98052
Email: dbansal@microsoft.com

Contributing Authors' Addresses

Jim Nyland
Microsoft
15590 NE 31st St
Redmond, WA 98052
Email: jnyland@microsoft.com

Geoff Outhred
Microsoft
15590 NE 31st St
Redmond, WA 98052
Email: geoffo@microsoft.com

Anh Cao
Microsoft
15590 NE 31st St
Redmond, WA 98052
Email: anhcao@microsoft.com

I2NSF BOF
Internet-Draft
Intended status: Standards Track
Expires: April 20, 2016

S. Hares
Huawei
A. Pastor
Telefonica I+D
K. Wang
China Mobile
D. Zhang

M. Zarny
Goldman Sachs
October 18, 2015

Analysis of Use Cases and Gaps in Technology for I2NSF
draft-hares-i2nsf-use-case-gap-analysis-00.txt

Abstract

This document provides a summary of the I2NSF use cases plus a summary of the state of the art in industries and IETF work which is 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. The use cases are organized in two basic scenarios. In the access network scenario, mobile and residential users access NSF capabilities using their network service provider infrastructure. In the data center scenario customers manage NSFs hosted in the data center infrastructure.

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 April 20, 2016.

Copyright Notice

Copyright (c) 2015 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	4
1.2. I2NSF Standarization	4
1.3. Structure of the document	5
2. Requirements Language	6
3. Terminology	6
4. Use Cases	7
4.1. General Use Cases	7
4.1.1. Instantiation and Configuration of NSFs	8
4.1.2. Updating of NSFs	8
4.1.3. Collecting the Status of NSFs	9
4.1.4. Validation of NSFs	9
4.2. Access Networks	9
4.2.1. vNSF Deployment	10
4.2.2. vNSF Customer Provisioning	10
4.3. Cloud Datacenter Scenario	10
4.3.1. On-Demand Virtual Firewall Deployment	11
4.3.2. Firewall Policy Deployment Automation	11
4.4. Considerations on Policy and Configuration	12
4.4.1. Translating Policies into NSF Capabilities	13
5. Gap Analysis	14
5.1. Structure of the gap analysis	14
5.2. IETF Gap analysis	15
5.2.1. Traffic Filters	15
5.3. ETSI NFV	22
5.3.1. ETSI Overview	22
5.3.2. I2NSF Gap Analysis	23
5.4. OPNFV	24
5.4.1. OPNFV Moon Project	24
5.4.2. Gap Analysis for OPNFV Moon Project	26
5.5. OpenStack Security Firewall	26

5.5.1.	Overview of API for Security Group	27
5.5.2.	Overview of Firewalls as a Service	27
5.5.3.	I2NSF Gap analysis	28
5.6.	CSA Secure Cloud	28
5.6.1.	CSA Overview	28
5.6.2.	I2NSF Gap Analysis	40
5.7.	In-depth Review of IETF protocols	40
5.7.1.	NETCONF and RESTCONF	40
5.7.2.	I2RS Protocol	41
5.7.3.	NETMOD Yang modules	42
5.7.4.	COPS	42
5.7.5.	PCP	43
5.7.6.	NSIS - Next steps in Signalling	44
6.	Summarized Requirements	45
7.	IANA Considerations	46
8.	Security Considerations	46
9.	Contributors	47
10.	References	47
10.1.	Normative References	47
10.2.	Informative References	47
	Authors' Addresses	54

1. Introduction

Enterprise, residential, and mobile customers are becoming more and more aware of the need for network security, just to find that security services are hard to operate and become expensive in the case of reasonably sophisticated ones. This general trend has caused numerous operators and security vendors to start to leverage on cloud-based models to deliver security solutions. In particular, the methods around Network Function Virtualization (NFV) are meant to facilitate the elastic deployment of software images providing the network services, and require the management of various resources by customers, who may not own or physically host those network functions.

There are numerous benefits by defining such interfaces. Operators could provide more flexible and customized security services for specific users and this would provide more efficient and secure protection to each user.

This document provides an analysis of the use cases, gaps analysis of existing technology, recommendations for requirements for I2NSF, and security considerations.

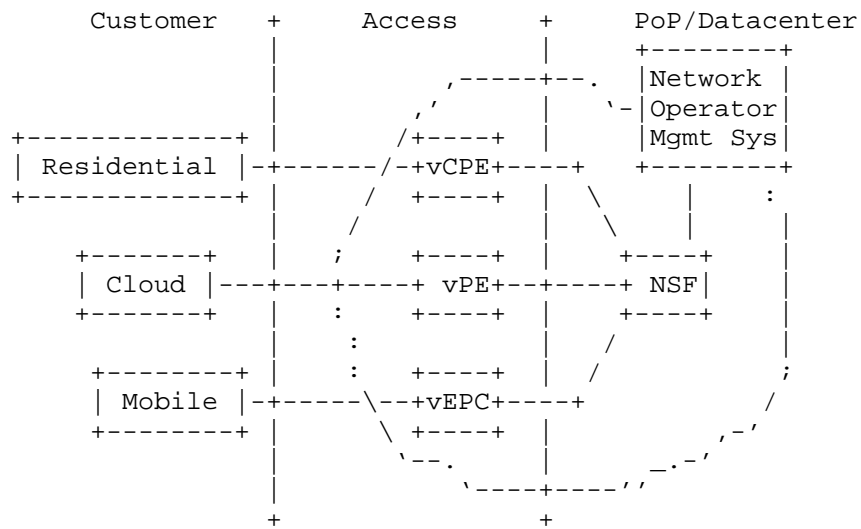


Figure 1: NSF and actors

1.1. What is I2NSF

A Network Security Function (NSF) is a function used to ensure integrity, confidentiality, or availability of network communications, to detect unwanted network activity, or to block or at least mitigate the effects of unwanted activity. NSFs are provided and consumed in increasingly diverse environments. Users could consume network security services enforced by NSFs hosted by one or more providers - which may be their own enterprise, service providers, or a combination of both. Similarly, service providers may offer their customers network security services that are enforced by multiple security products, functions from different vendors, or open source technologies. NSFs may be provided by physical and/or virtualized infrastructure. Without standard interfaces to control and monitor the behavior of NSFs, it has become virtually impossible for providers of security services to automate service offerings that utilize different security functions from multiple vendors.

1.2. I2NSF Standardization

The Interface to NSF devices (I2NSF) work proposes to standardize a set of software interfaces and data modules 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 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.

For the I2NSF capability layer, the I2NSF work proposes an interoperable protocol that passes NSF provisioning rules and orchestration information between I2NSF client on a network manager and I2NSF agent on an NSF device. 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).

1.3. Structure of the document

This document reviews the terminology (section 3), analyzes the use cases (section 4) and gaps in current technology (section 5), recommends certain requirements for I2NSF protocol(section 6), and discusses security consideration (section 8).

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. Terminology

- o Network Security Function (NSF): A functional block within a network infrastructure to ensure integrity, confidentiality and availability of network communications, to detect unwanted activity, and to deter and block this unwanted activity or at least mitigate its effects on the network
- o vNSF: Virtual Network Security Function: A network security function that runs as a software image on a virtualized infrastructure, and can be requested by one domain but may be owned or managed by another domain.
- o type of NSFs: NSFs considered in this draft include virtualized and non-virtualized NSFs.
- o Cloud DC: A data center that is not on 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>) focus on adding security to this environment. A specific research topic is security as a service within the cloud data center.
- o Cloud-based security functions: Network Security Function (NSF) hosted and managed by service providers or different administrative entity.
- o DC: Data Center
- o 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.

- o I2NSF agent - a piece of software in a device that implements a network security function which receives provisioning information and requests for operational data (monitoring data) across the I2NSF protocol from an I2NSF client.
- o 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)
- o 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. This management system is denoted as I2NS management system in this document.
- o Virtual Security Function: a security function that can be requested by one domain but may be owned or managed by another domain.

4. Use Cases

This section discusses general use cases, access use cases, and cloud use cases.

4.1. General Use Cases

User request security services through specific clients (a customer app, the NSP BSS/OSS or management platform...) and the appropriate NSP network entity will invoke the (v)NSFs according to the user service request. We will call this network entity the security controller. The interaction between the entities discussed above (client, security controller, NSF) is shown in the following diagram:

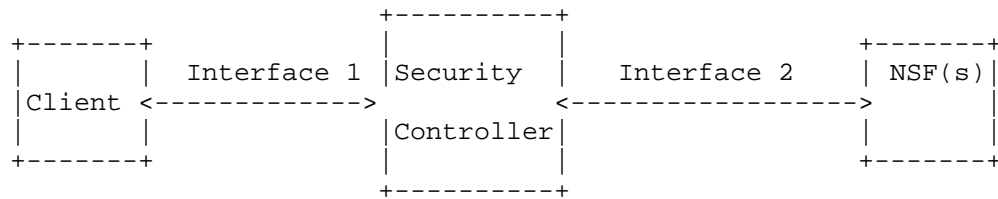


Figure 2: Interaction between Entities

Interface 1 is used for receiving security requirements from client and translating them into commands that NSFs can understand and execute. Moreover, it is also responsible for giving feedback of the NSF security statistics to client. Interface 2 is used for interacting with NSFs according to commands, and collect status information about NSFs.

4.1.1.1. Instantiation and Configuration of NSFs

Client sends collected security requirements through Interface 1 to the security controller in the NSP network, which then translates them into a set of security functions. Then the corresponding NSFs are instantiated and configured through Interface 2.

As an example, consider an enterprise user A who wants to prevent a certain kind of traffic from flowing to their network. Such a requirement is sent from client to security controller through Interface 1. The security controller translates the requirement into a firewall function plus a rules for filtering out TCP and/or UDP data packets. Then it instantiates a firewall NSF through Interface 2. The corresponding filter rules are also configured onto this firewall NSF through Interface 2.

4.1.1.2. Updating of NSFs

A user can direct the client to require the update of security service functions, including adding/deleting a security service function and updating configurations of former security service function.

As an example, consider a user who has instantiated a security service before and decides to enable an additional IDS service. This requirement will be sent to the security controller through Interface 1 and be translated, so the security controller instantiates and configures an IDS NSF through Interface 2.

4.1.3. Collecting the Status of NSF's

When users want to get the executing status of security service, they can request the status statistics information of NSF's from the client. The security controller will collect NSF status statistics 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.

4.1.4. Validation of NSF's

Customers may require to validate NSF availability, provenance, and its correct execution. This validation process, especially relevant for vNSF's, includes at least

Integrity of the NSF. Ensure that the NSF is not manipulated.

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

In order to achieve this the security controller has to collect security measurements and share them with an independent and trusted third party, allowing the user to attest the NSF by using Interface 1 and the information of the trusted third party.

4.2. Access Networks

This scenario describes use cases for users (enterprise user, network administrator, 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 vNSF's.

The Virtual CPE described in [NFVUC] use cases #5 and #7 cover the model of virtualization for mobile and residential access, where the operator may offload security services from the customer local environment (or even the terminal) to the operator infrastructure supporting the access network.

These use cases defines the operator interaction with vNSF's through automated interfaces, typically by B2B communications performed by the operator management systems (OSS/BSS).

4.2.1. vNSF Deployment

The deployment process consists of 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.

4.2.2. vNSF Customer Provisioning

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

- 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 of 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 Datacenter Scenario

In a datacenter, network security mechanisms such as firewalls may need to be added or removed dynamically for a number of reasons. It 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. Likewise, a service provider may need to provision a new firewall instance in a completely new environment due to a new requirement.

The on-demand, dynamic nature of deployment essentially requires that the network security "devices" be in software or virtual form factors, rather than in a physical appliance form. (This 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 right is of paramount importance but also to 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 separate; or that IPS/IDS services for investment banking and non-banking traffic be separate 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 myriad security policy requirements. 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.

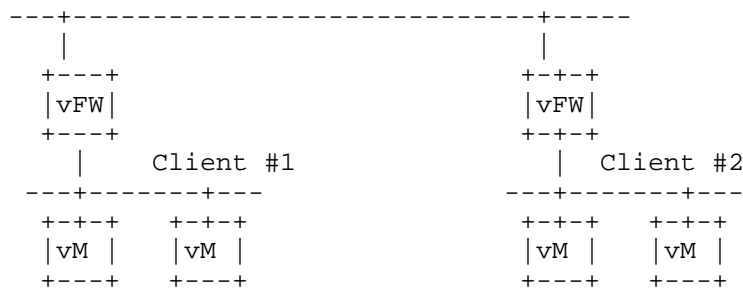


Figure 3: NSF in DataCenter

4.3.2. Firewall Policy Deployment Automation

Firewall configuration today is a highly complex process that involves consulting established security policies, translating those policies into firewall rules, further translating those rules into vendor-specific configuration sets, identifying all the firewalls, and pushing configurations to those firewalls.

This is often a time consuming, complex and error-prone process even within a single organization/enterprise framework. It becomes far more complex in provider-owned cloud networks that serve myriad customers.

Automation can help address many of these issues. Automation works best when it can leverage a common set of standards that will work across multiple entities.

4.3.2.1. Client-Specific Security Policy in Cloud VPNs

Clients of service provider operated cloud data centers need not only secure virtual private networks (VPNs) but also virtual security functions that enforce the clients' security policies. The security policies may govern communications within the clients' own virtual networks and those with external networks. For example, VPN service providers may need to provide firewall and other security services to their VPN clients. Today, it is generally not possible for clients to dynamically view, much less change, what, where and how security policies are implemented on their provider-operated clouds. Indeed, no standards-based framework that allows clients to retrieve/manage security policies in a consistent manner across different providers exists.

4.4. Considerations on Policy and Configuration

NSF configurations can vary from simple rules (i.e. block a DDoS attack) to very complex configuration (i.e. define a user firewall rules per application, protocol, source and destination port and address). The possibility of using configuration templates per control and management type is a common option as well.

A NSP can push security policies using complex configurations in their managed vNSF through its management system. The open Control and management interface has to accommodate this application-driven behavior.

Computer-savvy customers may pursue a similar application-driven configuration through the open Control and management interface, but standard residential and mobile customers may prefer to use the definition of security policies in the form of close-to-natural-language sentences with high-level directives or a guide configuration process. The representation for these policies will be of the form:

```
+-----+ +-----+ +-----+ +-----+
|Subject| + |Action| + |Object| + |Field_type = Value|
+-----+ +-----+ +-----+ +-----+
```

Figure 4: High-Level Security Policy Format

Subject indicates the customer or device in the access.

Action can include a variety of intent-based actions: check, redirect, allow, block, record, inspect..

Object can be optional and specifies the nature of the action. The default is all the customer traffic, but others possible values are connections and connections attempts.

Field_type allows to create fine-grained policies, including destinations list (i.e. IPs, domains), content types (i.e. files, emails), windows of time (i.e. weekend), protocol or network service (i.e. HTTP).

An example of a customer policy is:

"My son is allowed to access Facebook from 18:30 to 20:00"

4.4.1. Translating Policies into NSF Capabilities

Policies expressed in the above model are suitable for what we depicted as Interface 1 in Figure 2. In order to allow the security controller to deal with the different NSFs an intermediate representation used for expressing specific configurations in a device-independent format is required. For this purpose, the definition of a set of security capabilities provides a means for categorizing the actions performed by network security functions. An initial, high-level set of such capabilities consists of:

- o Identity Management: Includes all services related with identity, authentication and key management. Some examples are:
 - * AAA (Authentication, Authorization, Accounting) services
 - * Remote identity management
 - * Remote identity management
- o Traffic Inspection: A common use case for customers accessing the Internet or additional services through it is security supervision. Control and Management interfaces will allow the configuration of the vNSF inspection features: signatures updates, behavioral parameters or type of traffic to supervise. Some examples are:
 - * IDS/IPS (Intrusion Detection System/Intrusion Prevention System,
 - * Deep packet inspection,

- * Data leakage protection,
- o Traffic Manipulation: A more intrusive use case of NSF includes the capacity of manipulate the client traffic. Control and Management interfaces will allow the configuration of the NSF manipulation features, such as redirect and block rules. Some examples are:
 - * Redirect traffic, as in the case of captive portals,
 - * Block traffic: Firewalls, intrusion prevention system, DDOS/Anti-DOS (Distributed Denial-of-Service/Anti-Denial-of-Service),
 - * Encrypt traffic: VPN services that encapsulate and encrypt the user traffic. A SSL VPN is a representative example.
- o Impersonation: Some NSFs can impersonate a customer service or Internet service to provide security functions. Control and Management interfaces will allow the configuration of the service to impersonate and his behavioral. Some examples are:
 - * Honeypots, impersonating customer services, such as HTTP, NetBios or SSH,
 - * Anonymization services, hiding the source identity, as in the case of TOR.

Service Chain will allow for more than one of the aforementioned functions to engage in a specific order to a particular flow

5. Gap Analysis

5.1. Structure of the gap analysis

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

IETF working groups (section 5.2)

ETSI Network Functions Virtualization Industry Specification Group (ETSI NFV ISG), (section 5.3)

OPNFV Open Source Group (section 5.4)

Open Stack - Firewall as a service (OpenStack Firewall FaaS) (section 5.5) (http://docs.openstack.org/admin-guide-cloud/content/install_neutron-fwaas-agent.html)

Cloud Security Alliance Security (CSA) as a Service (section 5.6)
(https://cloudsecurityalliance.org/research/secaas/#_overview)

In-Depth Review of Some IETF Protocols (section 5.7)

5.2. 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.

5.2.1. Traffic Filters

5.2.1.1. 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]), and
- o the Port Control Protocol (PCP) to enables IPv4 to IPv6 flexible address and port mapping for 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.

The routing filtering is outside the scope of the flow filtering, but flow filtering may be impacted by route filtering. An initial model for the routing policy is in [I-D.shaikh-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 the Detailed I2NSF analysis.

5.2.1.1.1. 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 lists filters which can permit or deny traffic flow based on headers at the MAC, 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.

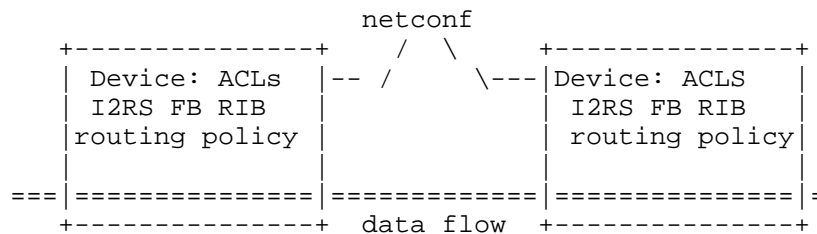


Figure 5 -I2RS Filter-Based RIB

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]). Please see the background section on I2RS for additional details on the requirements for this extension to the NETCONF/RESTCONF protocol suite.

The vocabulary set in [I-D.ietf-netmod-acl-model] is limited, so additional protocol independent filters were written for the I2RS Filter-Based RIBs in [I-D.hares-i2rs-bnp-eca-data-model], and protocol specific filters for SFC [I-D.dunbar-i2rs-discover-traffic-rules].

One thing important to note is that NETCONF and RESTCONF manage device layer yang models. However, as figure 6 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).

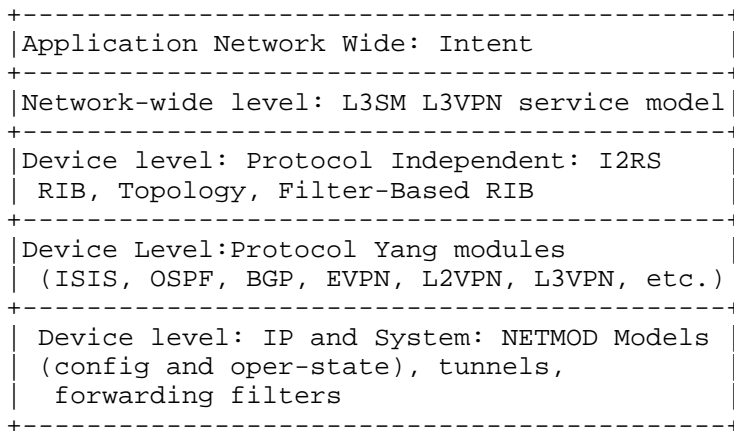


Figure 6 levels of Yang modules

5.2.1.1.2. 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 security expertise for these IDP/IPS, Web-filter, security-based deep-packet inspection 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.

5.2.1.2. Middle-box Filters

5.2.1.2.1. Midcom

Midcom Summary: MIDCOM developed the protocols for applications to communicate with middle boxes. However, MIDCOM have not used by the industry for a long time. This is because there was a lot of IPR

encumbered technology and IPR was likely a bigger problem for IETF 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 word, there's a trust model there 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 than Midcom

MIDCOM is different than I2NSF because its SNMP scheme doesn't work with the virtual network security functions (vNSF) management.

MidCom RFCs:

[RFC3303] - Midcom architecture

[RFC5189] - Midcom Protocol Semantics

[RFC3304] - Midcom protocol requirements

5.2.1.3. Security Work

5.2.1.3.1. 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 re-mediation. 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.

define the requirements for an I2NSF protocol to pass this traffic. (Hopefully 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 control), DOTS (DDoS Open Threat Signalling), and MILE (Management Incident Lightweight Exchange).

5.2.1.3.2. 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 5.7.4.

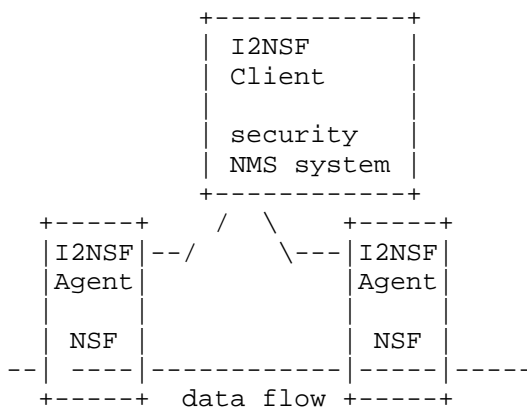


Figure 7

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

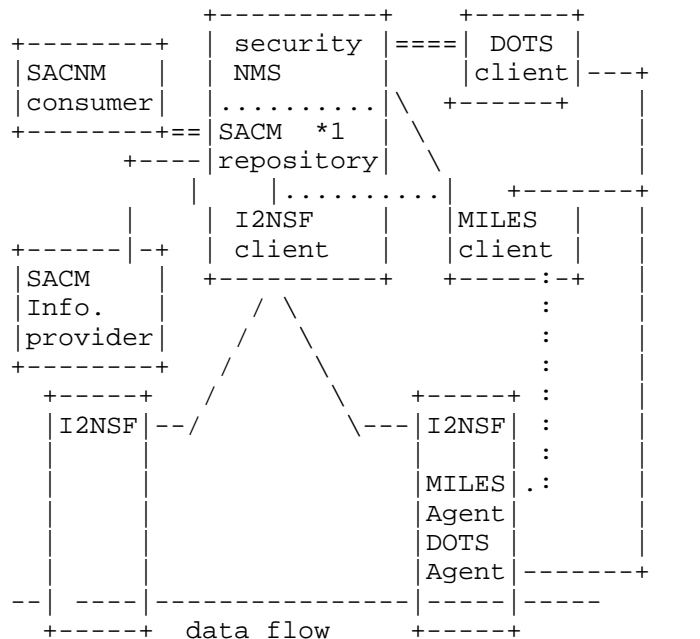
- o SACM [I-D.ietf-sacm-architecture] describes an architecture which tries to determine if the end-point security policies and the reality (denoted as security posture) align.

[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.

5.2.1.3.3. 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 8

Figure 8 provides a diagram of a system the I2NSF, SACM, DOTS and MILES 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 provider and a SACM consumer. 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 MILES client on a security network management system talking to the MILES 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.

5.2.1.3.4. 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 NETMOD's, ACLS are not sufficient for all filters or control desired by the NSF boxes.

The incident data captured by MILES 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.

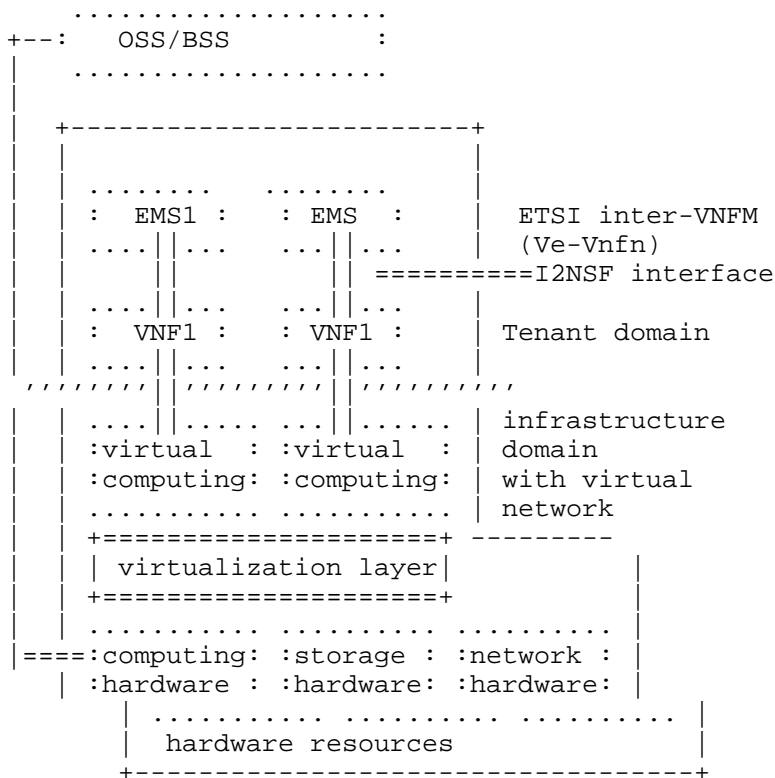
5.3. ETSI NFV

5.3.1. ETSI Overview

Network Function Virtualization (NFV) provides the 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.

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 single 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 device 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 will be deployed along networks using other security and NFV technology. As section 3 described, the NFV NSF security is deployed along side other security functions (AAA, SACM, DOTS, and MILE devices) or deep-packet-inspection. 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



In the NFV-related productions, the current architecture does not have a protocol to maintain an interoperability provisioning from I2NSF client to I2NSF agent. The result is that service providers have to manage the interoperability using private 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 for providers to start their NFV work from. However, this code base faces the same problem. There is no defacto standard protocol.

5.4. OPNFV

The OPNFV (www.opnfv.org) is a carrier-grade integrated, open source platform focused on accelerating the introduction of new Network Function 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.

5.4.1. 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 cart" to enforcement related security managers. A security management system integrates mechanisms of different security aspects. This project will first 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 10 shows. Figure 10 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 computing and Swift for software. Each system has a configuration file and its own security policy. This lacks the synchronization mechanism to build a complete secure configuration for OPNF.
- o No dynamic control so that if a user obtains the token, there is no way to obtain control over the user.
- o No customization or flexibility to allow integration into different vendors,
- o No fine grain authorization at user level. Authorization is only at the API

Moon addresses these issues adding authorization, logging, IDS, enforcement of network policy, and storage protection. Moon is based on OpenStack Keystone.

Deliverable time frame: 2S 2015

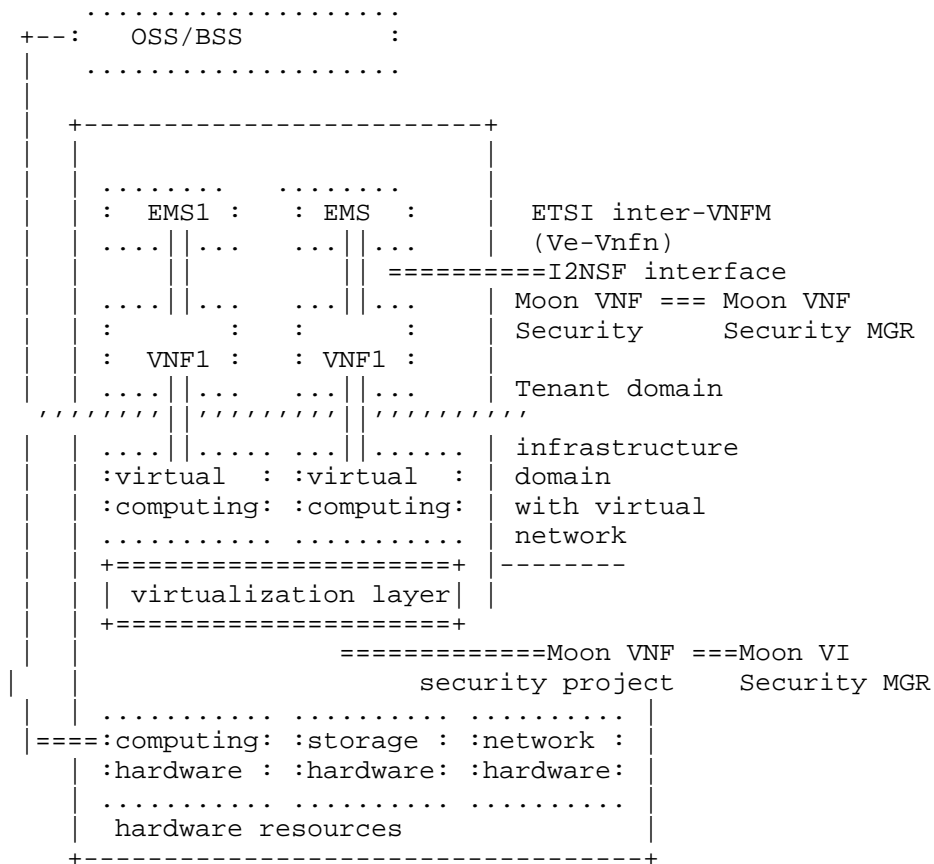


Figure 10

5.4.2. Gap Analysis for OPNFV Moon Project

OpenStack congress does not provide vendor independent systems.

5.5. 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

5.5.1. Overview of API for Security Group

The security group with the security group rules provides ingress and egress traffic filters based on port. The default group drops all ingress traffic and allows all egress traffic. The groups with additional filters are added to change this behaviour. To utilize the security groups, the networking plug-in for Open Stack must implement the security group API. The following plug-ins in OpenSTsack 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.

5.5.2. Overview of Firewalls 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, 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

5.5.3. 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. Security devices take time to be tested for functionality and their detection of security issues. 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 single management interfaces for a variety of security devices.

I2NSF is proposing rules that will include Event-Condition-matches (ECA) with the following matches

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

context based matches on schedule state and schedule, [Editor: Need more details here.]

The I2NSF is proposing action for these ECA policies of:

basic actions of deny, permit, and mirror,

advanced actions of: IPS signature filtering and URL filtering.

5.6. CSA Secure Cloud

5.6.1. 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

5.6.1.1. 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 guidances

5.6.1.2. 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 I2NSF

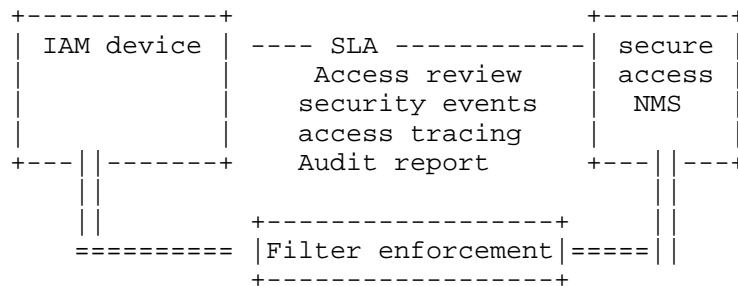


Figure 11

5.6.1.4. 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.

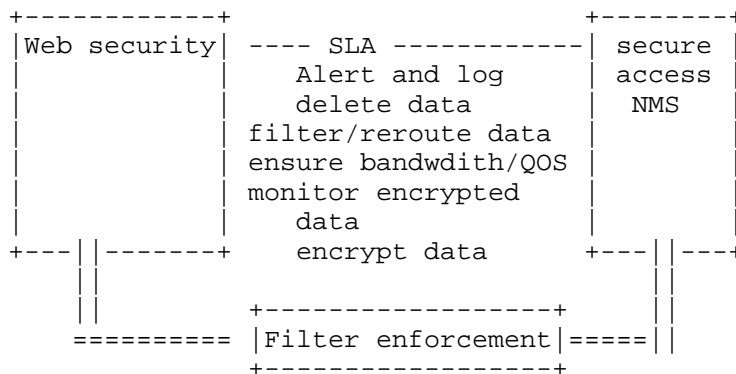


Figure 13

5.6.1.5. 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,

alert and provide rapid response to email threats,
 identify email users and secure remote access to email,
 do on-demand provisioning of email services,
 filter out (block/quarantine) email data,
 know where the email traffic or data is residing (to to regulatory
 issues), and
 be able to monitor encrypted email,
 be able to encrypt email,
 be able to retain email records (while abiding with privacy
 concerns), and
 interact with policy and network management systems.

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

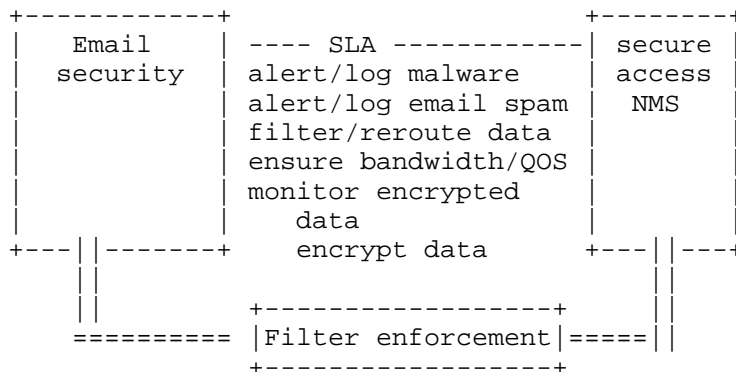


Figure 14

5.6.1.6. Security Assessment

Document :

https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_5_Security_Assessments_Implementation_Guidance.pdf

The CSA SaaS Security assessment indicates that assessments need to be done on the following devices:

- o hypervisor infrastructure,

- 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.

5.6.1.7. 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 (IDS) or intrusion protection devices. The management of these systems requires also requires:

Central reporting of events and alerts,
 administrator notification of intrusions,
 Mapping of alerts to Cloud-Layer Tenancy,
 Cloud sourcing information to prevent false positives in
 detection, and
 allowing for redirection of traffic to allow remote storage or
 transmission to prevent local evasion.

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

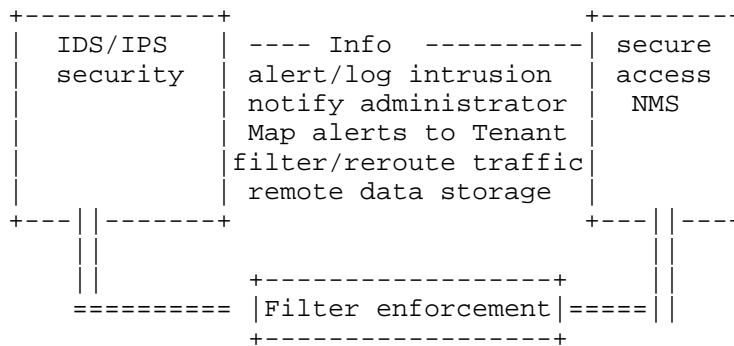


Figure 15

5.6.1.8. Security Information and Event Management (SIEM)

Document:

https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_7_SIEM_Implementation_Guidance.pdf

The Security Information and Event Management (SIEM) receives data from a wide range of security systems such as Identity management systems (IAM), data loss prevention (DLP), web security (Web), email security (email), intrusion detection/prevision (IDS/IPS), encryption, disaster recovery, and network security. The SIEM combines this data into a single streams. All the requirements for data to/from these systems are replicated in these systems needs to give a report to the SIEM system.

A SIEM system would be prime candidate to have a 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.

5.6.1.9. 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, Identitybased Encryption, and others;

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

- Cryptographic Security Standards and Guidelines

The wide variety of encryption services require the 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.

5.6.1.10. 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;

old 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.

The wide variety of BC/DR requires the 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.

5.6.1.11. 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 addition, the need for the management protocol like I2NSF is critical in the sprawl of Cloud environment.

5.6.2. I2NSF Gap Analysis

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

5.7. In-depth Review of IETF protocols

5.7.1. 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 datastores defined in NETCONF.

RESTCONF supports "two edit condition detections" - time stamp and entity tag. RESTCONF uses a 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]

5.7.2. 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 Ephemeral state [I-D.ietf-i2rs-ephemeral-state],

- o Publication-Subscription notifications [I-D.ietf-i2rs-pub-sub-requirements],
- o Traceability [I-D.ietf-i2rs-traceability],
- o Security requirements [I-D.hares-i2rs-auth-trans]

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.

5.7.3. 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 (merge of [I-D.shaikh-idr-bgp-model] and [I-D.zhdankin-idr-bgp-cfg] with the bgp policy proposed multiple Working groups (idr and rtgwg)). BGP Services yang models have been proposed for PPB EVPN ([I-D.tsingh-bess-pbb-evpn-yang-cfg]), EVPN ([I-D.zhuang-bess-evpn-yang]), L3VPN ([I-D.zhuang-bess-l3vpn-yang]), and multicast MPLS/BGP IP VPNs ([I-D.liu-bess-mvpn-yang]).

5.7.4. 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 16. The Policy decision point kept network-wide policy (E.g. ACLs) and sent it to Policy enforcements who then would control what data flows between the two. These decision points controlled data flow from PEP to PEP. [RFC3084] describes COPS use for policy provisioning.

Figure 16

[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 the management of port and address information rather than any other security functions
- o Cover the proxy, firewall and NAT box proposals in I2NSF

5.7.6. NSIS - Next steps in Signalling

NSIS is for standardizing an IP signalling protocol (RSVP) along data path for end points to request its unique QoS characteristics, unique FW policies or NAT needs (RFC5973) that are different from the FW/NAT original setting. 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 you've got one or more NATs present in the network, or when trying to locate appropriate tunnel endpoints).

A diagram should be added here showing I2NSF and NSIS

Why I2NSF is different than NSIS:

- o The I2NSF requests from clients do not go directly to network security devices, but instead to controller or orchestrator that can translate the application/user oriented policies to the involved devices in the interface that they support.
- 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 in the controller and orchestrator
- 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 we believe I2NSF has a higher chance to be deployed than NSIS:

- o Open Stack 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 Open Stack 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].

6. Summarized Requirements

The I2NSF framework should provide a set of standard interfaces that facilitate:

- o Dynamic creation, enablement, disablement, and removal of network security functions;
- o Policy-driven placement of new function instances in the right administrative domain;
- o Attachment of appropriate security and traffic policies to the function instances
- o Management of deployed instances in terms of fault monitoring, utilization monitoring, event logging, inventory, etc.

Moreover, an I2NSF must support different deployment scenarios:

- o Single and multi-tenant environments: The term multi-tenant does not mean just different companies subscribing to a provider's

offering. It can for instance cover administrative domains/ departments within a single firm that require different security and traffic policies.

- o Premise-agnostic: Said network security functions may be deployed on premises or off premises of an organization.

The I2NSF framework should provide a standard set of interfaces that enable:

- o Translation of security policies into functional tasks. Security policies may be carried out by one or more security functions. For example, a security policy may be translated into an IDS/IPS policy and a firewall policy for a given application type.
- o Translation of functional tasks into vendor-specific configuration sets. For example, a firewall policy needs to be converted to vendor-specific configurations.
- o Retrieval of information such as configuration, utilization, status, etc. Such information may be used for monitoring, auditing, troubleshooting purposes. The above functionality should be available in single- or multi-tenant environments as well as on-premise or off-premise clouds.

7. IANA Considerations

No IANA considerations exist for this document.

8. Security Considerations

The relationship between different actors define the security level for the different use cases and must be associated with administrative domains:

- o Closed environments where there is only one administrative network domain. More permissive access controls and lighter validation shall be allowed inside the domain because of the protected environment. Integration with existing identity management systems is also possible.
- o Open environments where some NSFs can be hosted in different administrative domains, and more restrictive security controls are required. The interfaces to the NSFs must use trusted channels. Identity frameworks and federations are common models for authentication and Authorization. Security controllers will be in charge of this functionalities.

Virtualization applied to NSF environment (vNSF) generate several concerns in security, being one of the most relevant the attestation of the vNSF by the clients. A holistic analysis has been done in [NFVSEC].

9. Contributors

I2NSF is a group effort. The following people contributed actively to the initial use case text: Diego R. Lopez (Telefonica I+D), Xiaojun Zhuang (China Mobile), Minpeng Qi (China Mobile), Sumandra Majee (F5), Nic Leymann (Deutsche Telekom), Linda Dunbar (Huawei).

10. References

10.1. 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>>.

10.2. Informative References

- [I-D.dunbar-i2rs-discover-traffic-rules]
Dunbar, L. and S. Hares, "An Information Model for Filter Rules for Discovery and Traffic for I2RS Filter-Based RIB", draft-dunbar-i2rs-discover-traffic-rules-00 (work in progress), March 2015.
- [I-D.hares-i2rs-auth-trans]
Hares, S., Migault, D., and J. Halpern, "I2RS Security Related Requirements", draft-hares-i2rs-auth-trans-05 (work in progress), August 2015.
- [I-D.hares-i2rs-bnp-eca-data-model]
Hares, S., Wu, Q., Tantsura, J., and R. White, "An Information Model for Basic Network Policy and Filter Rules", draft-hares-i2rs-bnp-eca-data-model-00 (work in progress), July 2015.
- [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.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-09 (work in progress), March 2015.
- [I-D.ietf-i2rs-ephemeral-state]
Haas, J. and S. Hares, "I2RS Ephemeral State Requirements", draft-ietf-i2rs-ephemeral-state-02 (work in progress), September 2015.
- [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-06 (work in progress), January 2015.
- [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-03 (work in progress), October 2015.
- [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-01 (work in progress), September 2015.
- [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-07 (work in progress), September 2015.
- [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-03 (work in progress), May 2015.
- [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-01 (work in progress), July 2015.

- [I-D.ietf-i2rs-yang-network-topo]
Clemm, A., Medved, J., Varga, R., Tkacik, T., Bahadur, N.,
and H. Ananthakrishnan, "A Data Model for Network
Topologies", draft-ietf-i2rs-yang-network-topo-01 (work in
progress), June 2015.
- [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-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-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-03 (work in progress), March 2015.

[I-D.ietf-sacm-terminology]

Waltermire, D., Montville, A., Harrington, D., Cam-Winget, N., Lu, J., Ford, B., and M. Kaeo, "Terminology for Security Assessment", draft-ietf-sacm-terminology-06 (work in progress), February 2015.

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

Kini, S., Hares, S., Dunbar, L., Ghanwani, A., Krishnan, R., Bogdanovic, D., Tantsura, J., and R. White, "Filter-Based RIB Information Model", draft-kini-i2rs-fb-rib-info-model-01 (work in progress), July 2015.

[I-D.l3vpn-service-yang]

Litkowski, S., Shakir, R., Tomotaki, L., and K. D'Souza, "YANG Data Model for L3VPN service delivery", draft-l3vpn-service-yang-00 (work in progress), February 2015.

[I-D.liu-bess-mvpn-yang]

Liu, Y. and F. Guo, "Yang Data Model for Multicast in MPLS/BGP IP VPNs", draft-liu-bess-mvpn-yang-00 (work in progress), April 2015.

- [I-D.shaikh-idr-bgp-model]
Shaikh, A., D'Souza, K., Bansal, D., and R. Shakir, "BGP Model for Service Provider Networks", draft-shaikh-idr-bgp-model-01 (work in progress), March 2015.
- [I-D.shaikh-rtgwg-policy-model]
Shaikh, A., Shakir, R., D'Souza, K., and C. Chase, "Routing Policy Configuration Model for Service Provider Networks", draft-shaikh-rtgwg-policy-model-01 (work in progress), July 2015.
- [I-D.tsingh-bess-pbb-evpn-yang-cfg]
Tiruveedhula, K., Singh, T., Sajassi, A., Kumar, D., and L. Jalil, "YANG Data Model for PBB EVPN protocol", draft-tsingh-bess-pbb-evpn-yang-cfg-00 (work in progress), March 2015.
- [I-D.zhang-i2rs-l1-topo-yang-model]
Zhang, X., Rao, B., and X. Liu, "A YANG Data Model for Layer 1 Network Topology", draft-zhang-i2rs-l1-topo-yang-model-01 (work in progress), March 2015.
- [I-D.zhdankin-idr-bgp-cfg]
Alex, A., Patel, K., Clemm, A., Hares, S., Jethanandani, M., and X. Liu, "Yang Data Model for BGP Protocol", draft-zhdankin-idr-bgp-cfg-00 (work in progress), January 2015.
- [I-D.zhuang-bess-evpn-yang]
Zhuang, S. and Z. Li, "Yang Model for Ethernet VPN", draft-zhuang-bess-evpn-yang-00 (work in progress), December 2014.
- [I-D.zhuang-bess-l3vpn-yang]
Zhuang, S. and Z. Li, "Yang Data Model for BGP/MPLS IP VPNs", draft-zhuang-bess-l3vpn-yang-00 (work in progress), December 2014.
- [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>>.
- [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

Antonio Pastor
Telefonica I+D
Don Ramon de la Cruz, 82
Madrid 28006
Spain

Email: antonio.pastorperales@telefonica.com

Ke Wang
China Mobile
32 Xuanwumenxi Ave,Xicheng District
Beijing 100053
China

Email: wangkeyj@chinamobile.com

Dacheng Zhang
Beijing
China

Email: dacheng.zdc@aliabab-inc.com

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

Email: myo.zarny@gs.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.

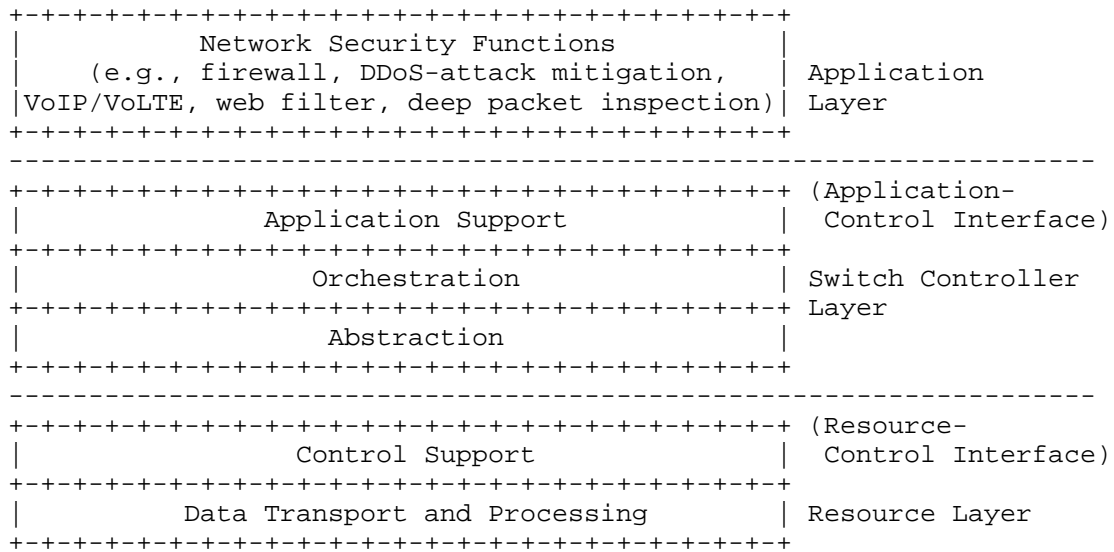


Figure 1: High-level Architecture for SDN-based Security Services

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.

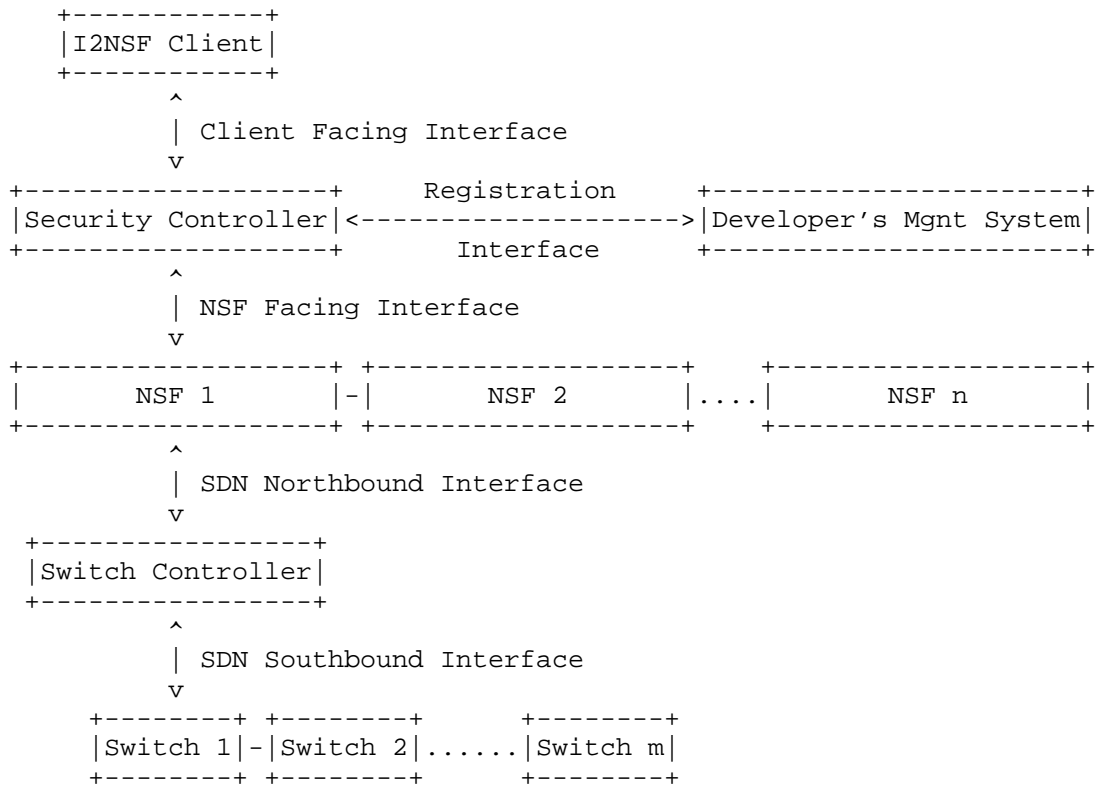


Figure 2: A Framework for SDN-based Security Services using I2NSF

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: Informational
Expires: September 2016

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

March 16, 2016

Framework for Interface to Network Security Functions
draft-merged-i2nsf-framework-05.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 September 16, 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 document defines the framework for guiding the functionality provided by I2NSF. Network security functions (NSFs) are packet-processing engines that inspect and optionally modify packets traversing networks, either directly or in the context of sessions in which the packet is associated. This document provides an overview of how NSFs are used, and describes how NSF software interfaces are controlled and monitored using rulesets. The design of these software interfaces must prevent the creation of implied constraints on NSF capability and functionality.

Table of Contents

1. Introduction.....	3
2. Conventions used in this document.....	4
3. Interfaces to Flow-based NSFs.....	4

4. Reference Models in Managing Flow-based NSFs.....	7
4.1. NSF Facing (Capability Layer) Interface.....	8
4.2. Client Facing (Service Layer) Interface.....	9
4.3. Vendor Facing Interface.....	9
4.4. The Network Connecting the Security Controller and NSFs...	9
4.5. Interface to vNSFs.....	10
5. Flow-based NSF Capability Characterization.....	11
6. Structure of Rules for governing NSFs.....	15
6.1. Capability Layer Rules and Monitoring.....	15
6.2. Service Layer Policy.....	16
7. Capability Negotiation.....	19
8. Types of I2NSF clients.....	19
9. Manageability Considerations.....	20
10. Security Considerations.....	20
11. IANA Considerations.....	20
12. References.....	21
12.1. Normative References.....	21
12.2. Informative References.....	21
13. Acknowledgments.....	22

1. Introduction

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

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

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

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [RFC2119].

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

BSS: Business Support System

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

FW: Firewall

IDS: Intrusion Detection System

IPS: Intrusion Protection System

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

OSS: Operation Support System

vNSF: refers to NSF being instantiated on Virtual Machines.

3. Interfaces to Flow-based NSFs

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

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

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

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

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

- Narrowly define NSF categories, or their roles when implemented within a network
- Attempt to impose functional requirements or constraints, either directly or indirectly, upon NSF developers

- Be a limited lowest common denominator approach, where interfaces can only support a limited set of standardized functions, without allowing for vendor-specific functions
- Be seen as endorsing a best common practice for the implementation of NSF

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

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

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

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

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

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

Actions for flow-based NSF's include:

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

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

4. Reference Models in Managing Flow-based NSF's

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

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

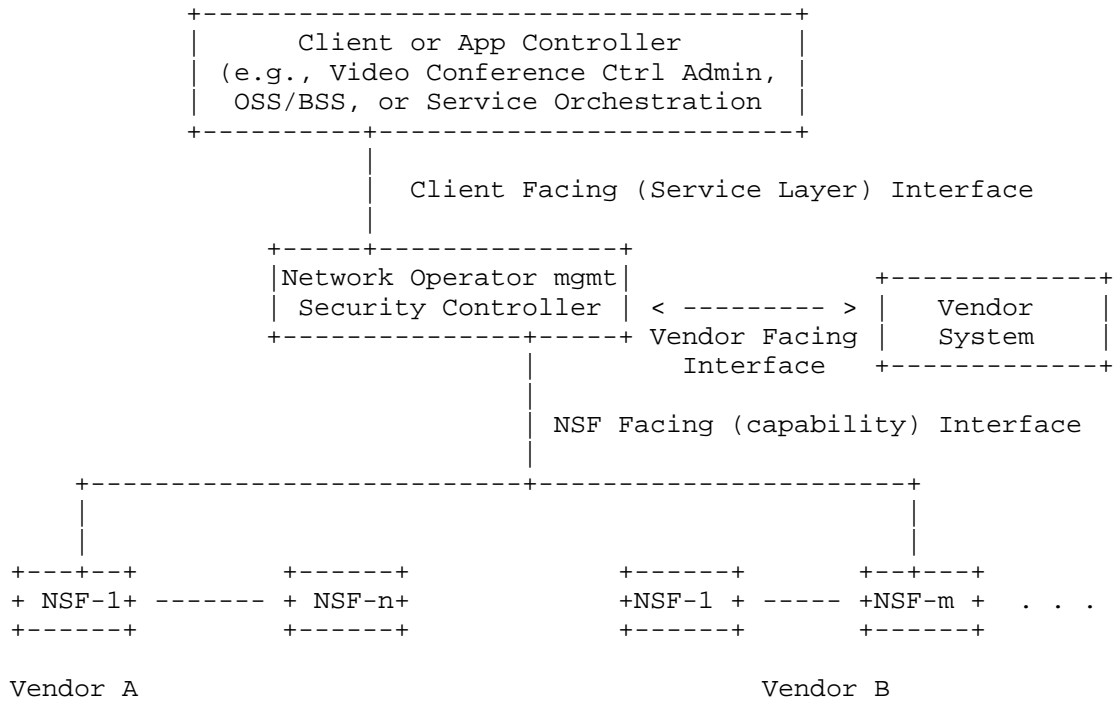


Figure 1: Multiple Interfaces

4.1. NSF Facing (Capability Layer) Interface

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

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

4.2. Client Facing (Service Layer) Interface

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

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

4.3. Vendor Facing Interface

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

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

4.4. The Network Connecting the Security Controller and NSFs

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

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

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

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

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

4.5. Interface to vNSFs

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

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

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

- When multiple instantiations of one single NSF appear as one single entity to the Security Controller, the policy provisioning has to be sent to the NSF's sub-controller, which in turn disseminates the 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.

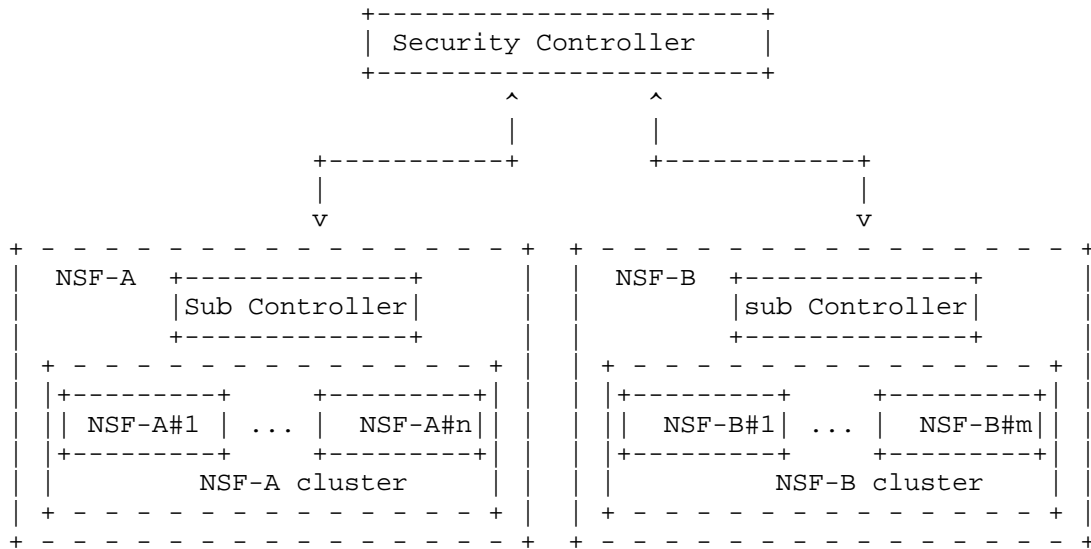


Figure 2: Cluster of NSF Instantiations Management

5. Flow-based NSF Capability Characterization

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

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

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

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

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

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

Vendors can register their NSF's 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 IPv4 Header	Layer header fields: protocol dest port src port src address dest address dscp length flags ttl
IPv6 Header	addr protocol/nh src port dest port src address dest address length traffic class hop limit flow label dscp
TCP SCTP DCCP	Port syn ack fin rst ? psh

	? urg ? window sockstress Note: bitmap could be used to represent all the fields
UDP	
HTTP layer	flood abuse fragment abuse Port 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

6. Structure of Rules for governing NSFs

6.1. Capability Layer Rules and Monitoring

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

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

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

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

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

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

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

6.2. Service Layer Policy

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

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

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

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

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

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

The I2NSF simple service layer can have the following entities:

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

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

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

7. Capability Negotiation

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

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

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

8. Types of I2NSF clients

It is envisioned that I2NSF clients include:

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

- Operator Management System dynamically updates, monitors and verifies the security policies to NSF's (by different vendors) in a network.
 - Third party system
- Security functions send requests for more sophisticated functions upon detecting something suspicious, usually via a security controller.

9. Manageability Considerations

Management of NSF's 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.

10. 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).

11. IANA Considerations

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

12. References

12.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3060] Moore, B, et al, "Policy Core Information Model (PCIM)", RFC 3060, Feb 2001.
- [RFC3460] Moore, B. "Policy Core Information Model (PCIM) Extensions", RFC3460, Jan 2003.
- [RFC5575] Marques, P, et al, "Dissemination of Flow Specification Rules", RFC 5575, Aug 2009.
- [RFC7297] Boucadair, M., "IP Connectivity Provisioning Profile", RFC7297, April 2014.

12.2. Informative References

- [I2NSF-ACCESS] A. Pastor, et al, "Access Use Cases for an Open OAM Interface to Virtualized Security Services", <draft-pastor-i2nsf-access-usecases-00>, Oct 2014.
- [I2NSF-DC] M. Zarny, et al, "I2NSF Data Center Use Cases", <draft-zarny-i2nsf-data-center-use-cases-00>, Oct 2014.
- [I2NSF-MOBILE] M. Qi, et al, "Integrated Security with Access Network Use Case", <draft-qi-i2nsf-access-network-usecase-00>, Oct 2014
- [I2NSF-Problem] L. Dunbar, et al "Interface to Network Security Functions Problem Statement", <draft-dunbar-i2nsf-problem-statement-01>, Jan 2015
- [ACL-MODEL] D. Bogdanovic, et al, "Network Access Control List (ACL) YANG Data Model", <draft-ietf-net-acl-model-00>, Nov 2014.

- [gs_NFV] ETSI NFV Group Specification, Network Functions 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.

13. Acknowledgments

Acknowledgements to xxx for his review and contributions.

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

Authors' Addresses

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

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

XiaoJun Zhuang
China Mobile
Email: zhuangxiaojun@chinamobile.com

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

John Strassner
Huawei
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

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.

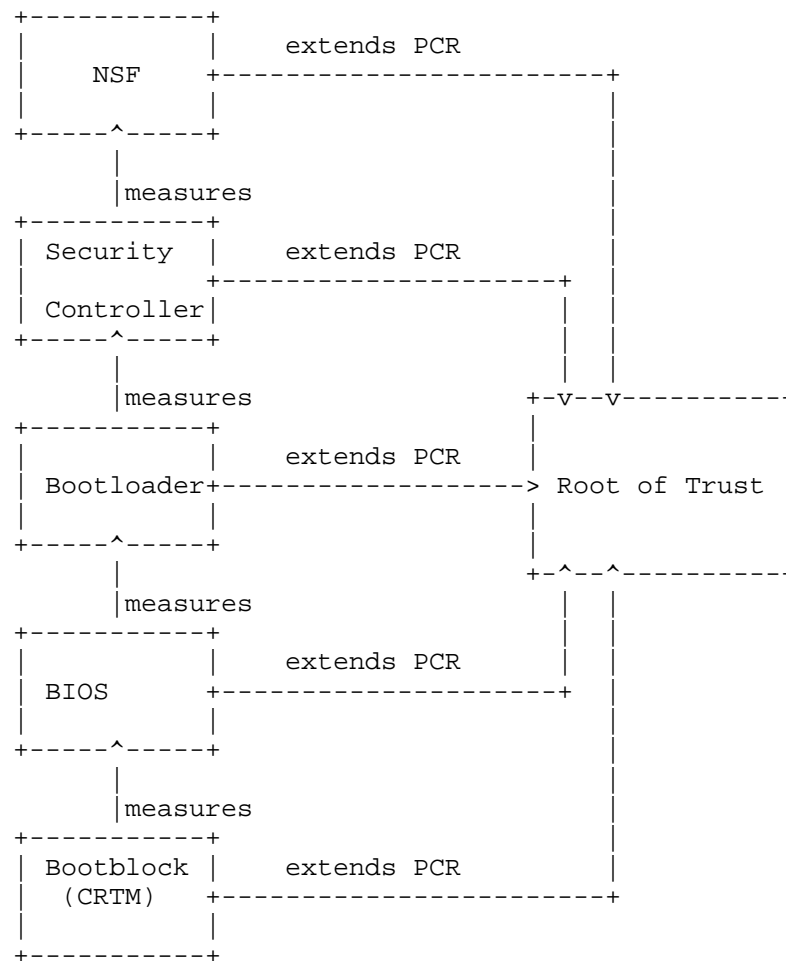


Figure 1: Applying Trusted Computing

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.

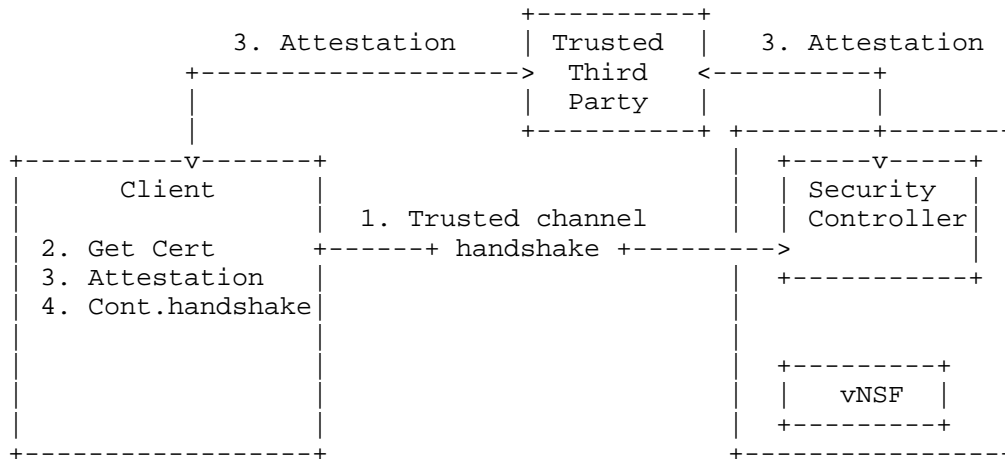


Figure 2: Steps for remote attestation

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:



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-supa-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:

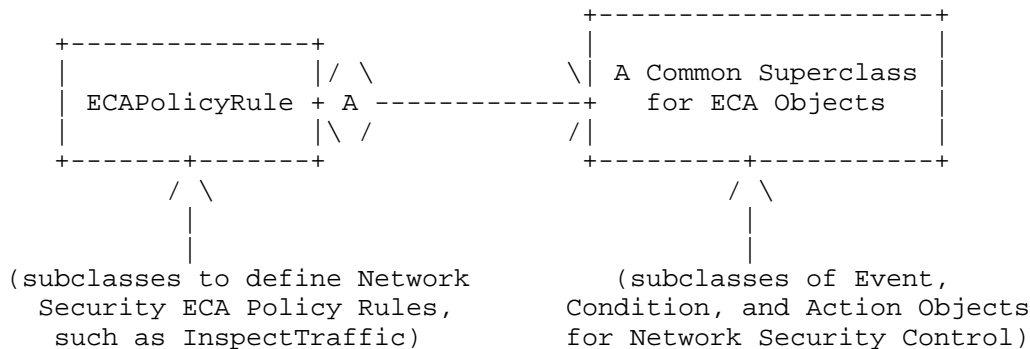


Figure 2. Network Security Information Sub-Model Overview

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.

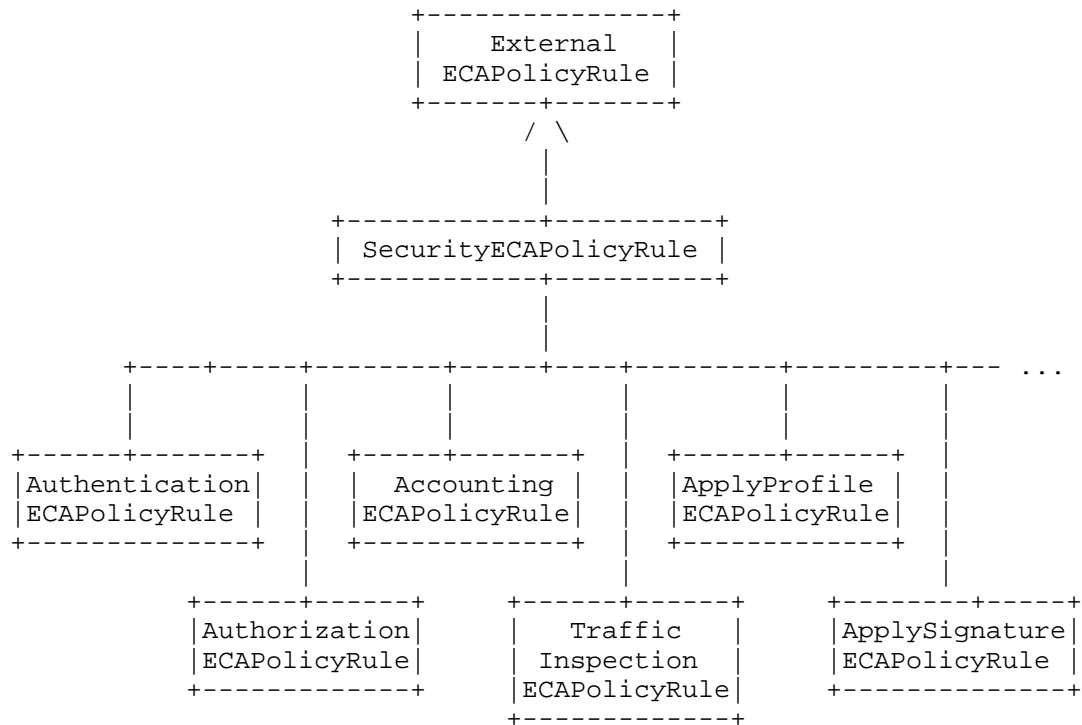


Figure 3. Network Security Info Sub-Model ECAPolicyRule Extensions

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.

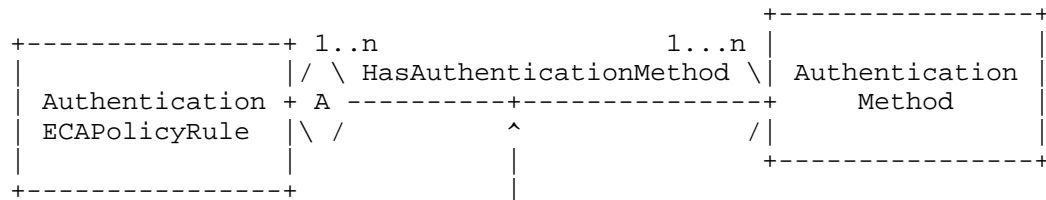
Brief class descriptions of these six ECA Policy Rules are provided in the following sub-sections. Note that there is a common pattern that defines how these ECAPolicyRules operate; this simplifies their implementation. All of these six ECA Policy Rules are concrete classes.

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:



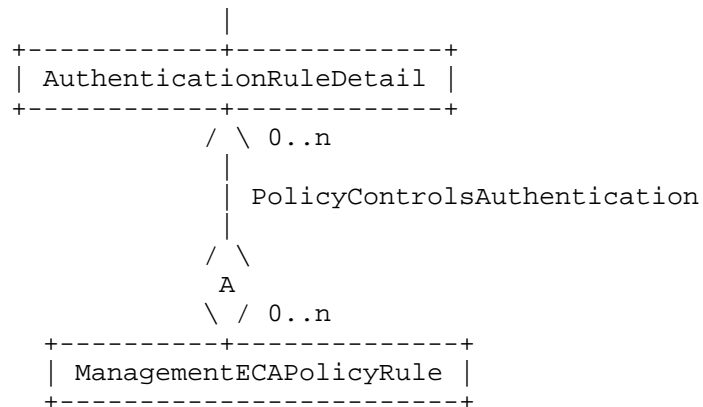


Figure 4. Modeling Authentication Mechanisms

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:

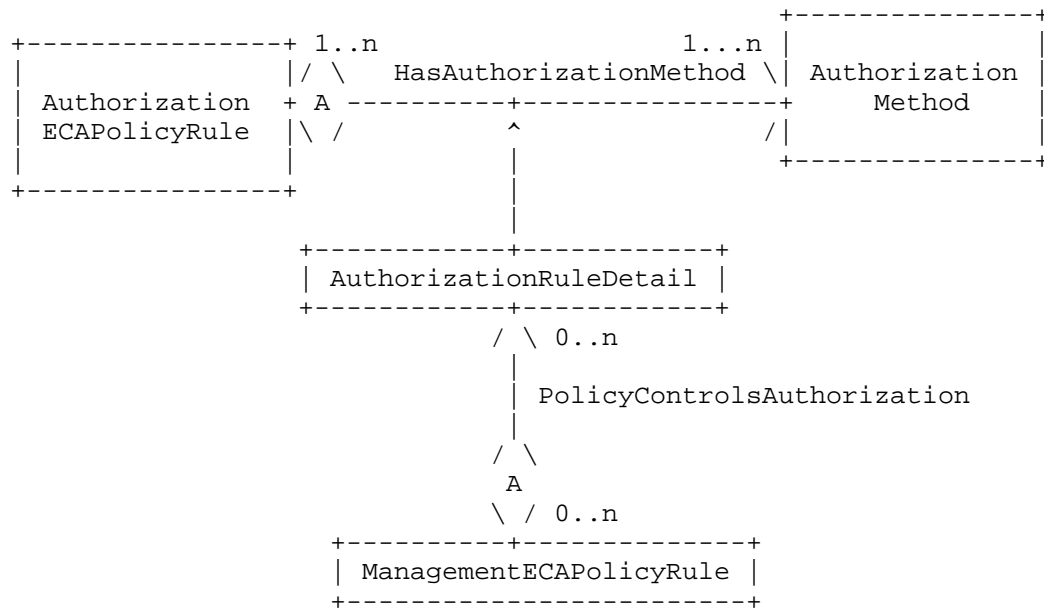


Figure 5. Modeling Authorization Mechanisms

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:

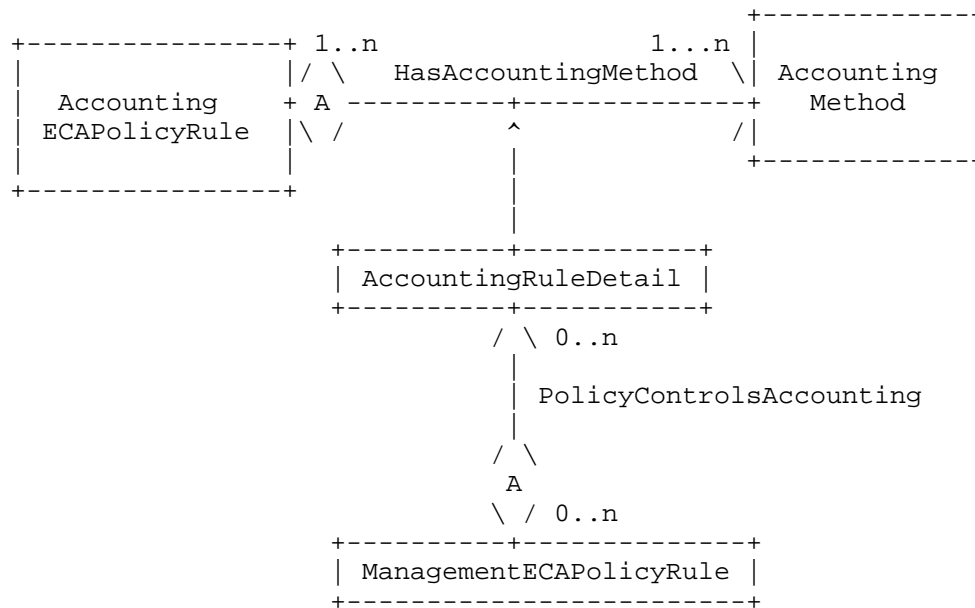


Figure 6. Modeling Accounting Mechanisms

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:

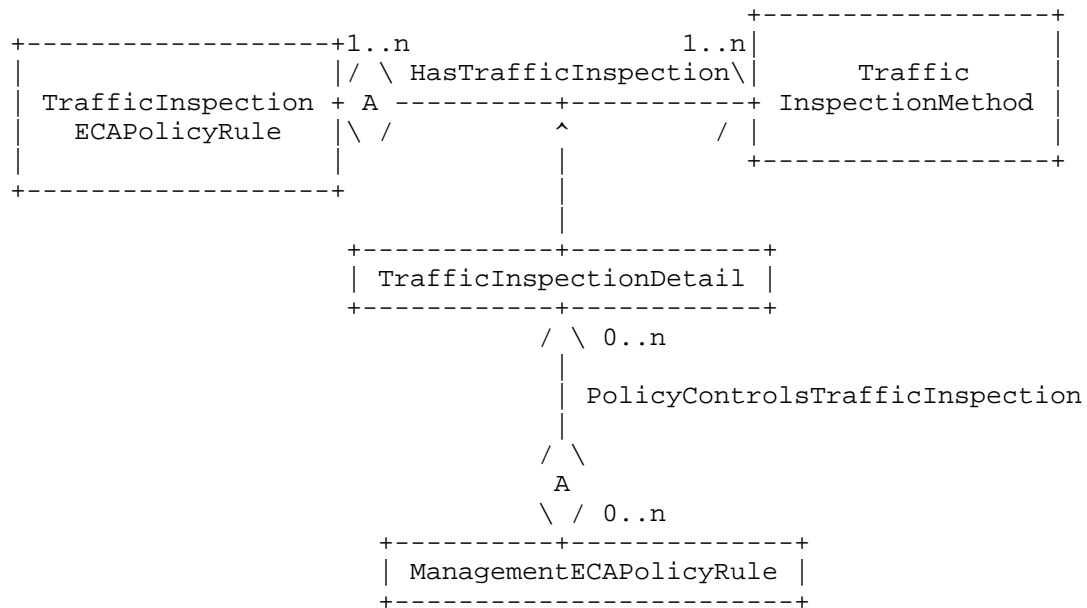


Figure 7. Modeling Traffic Inspection Mechanisms

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:

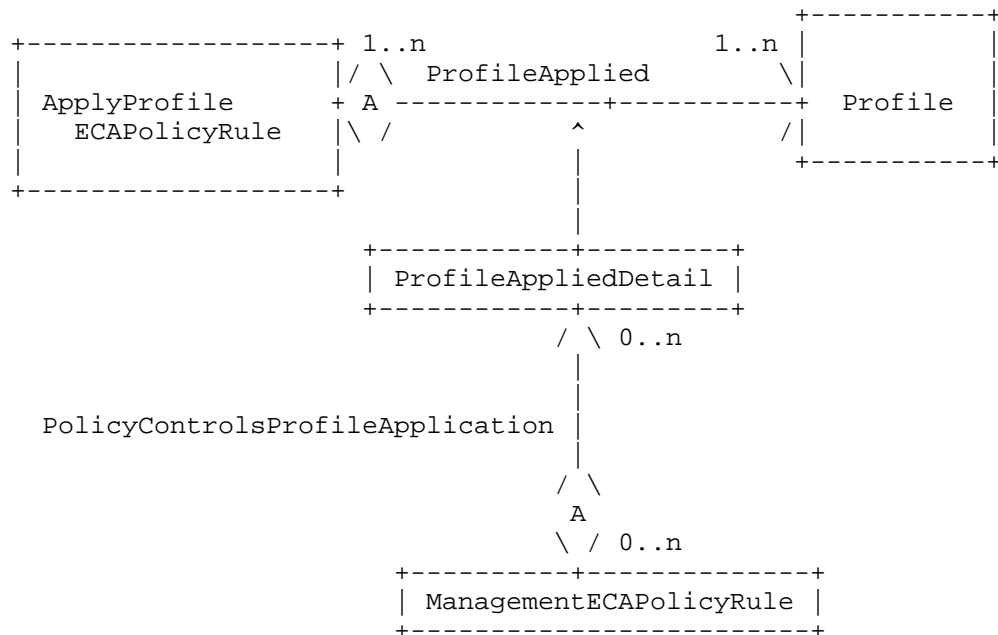


Figure 8. Modeling Profile ApplicationMechanisms

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 ApplyProfileECAPolicyRuleclass, 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. ApplySignatureECAPolicyRuleClass 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 aURL 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:

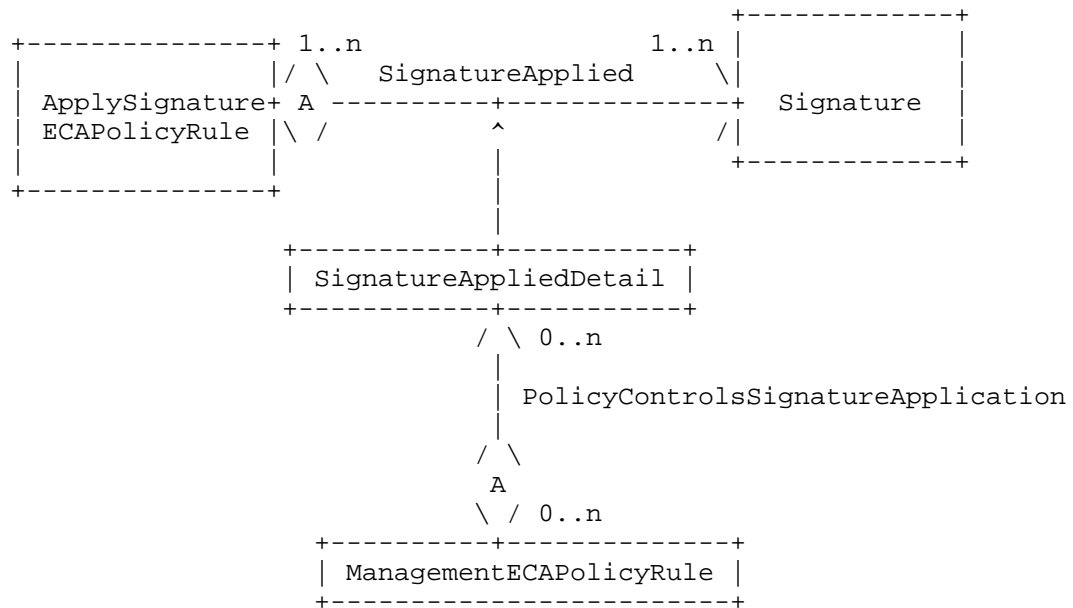


Figure 9. Modeling Sginature Application Mechanisms

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.

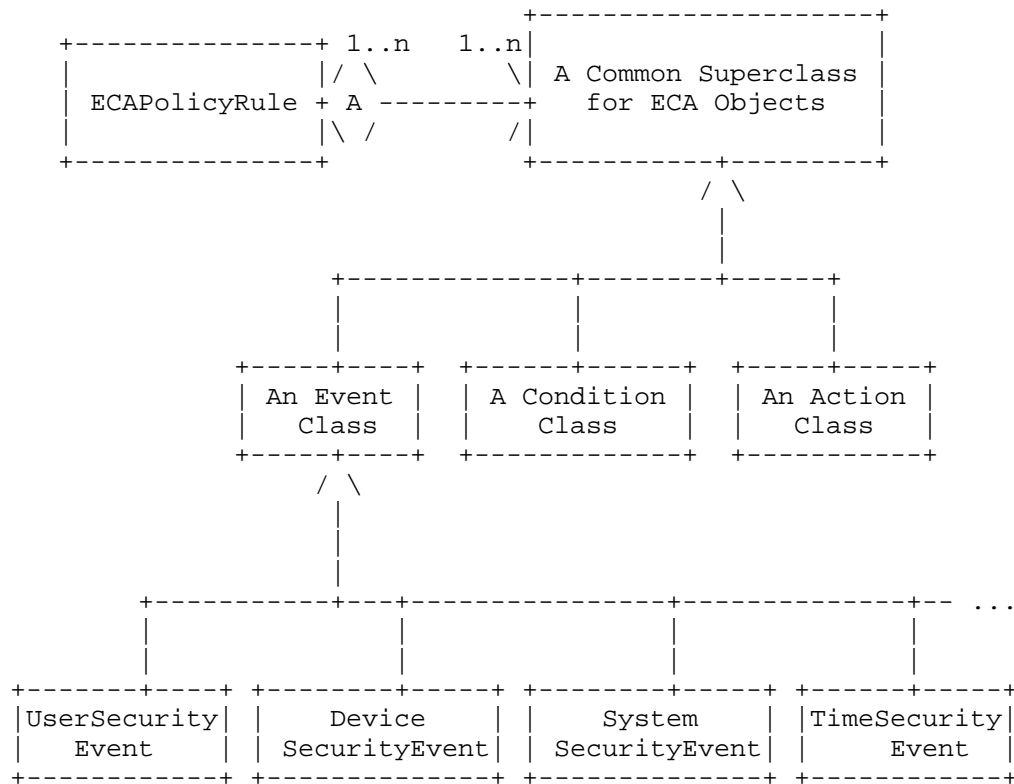


Figure 10. Network Security Info Sub-Model Event Class Extensions

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.

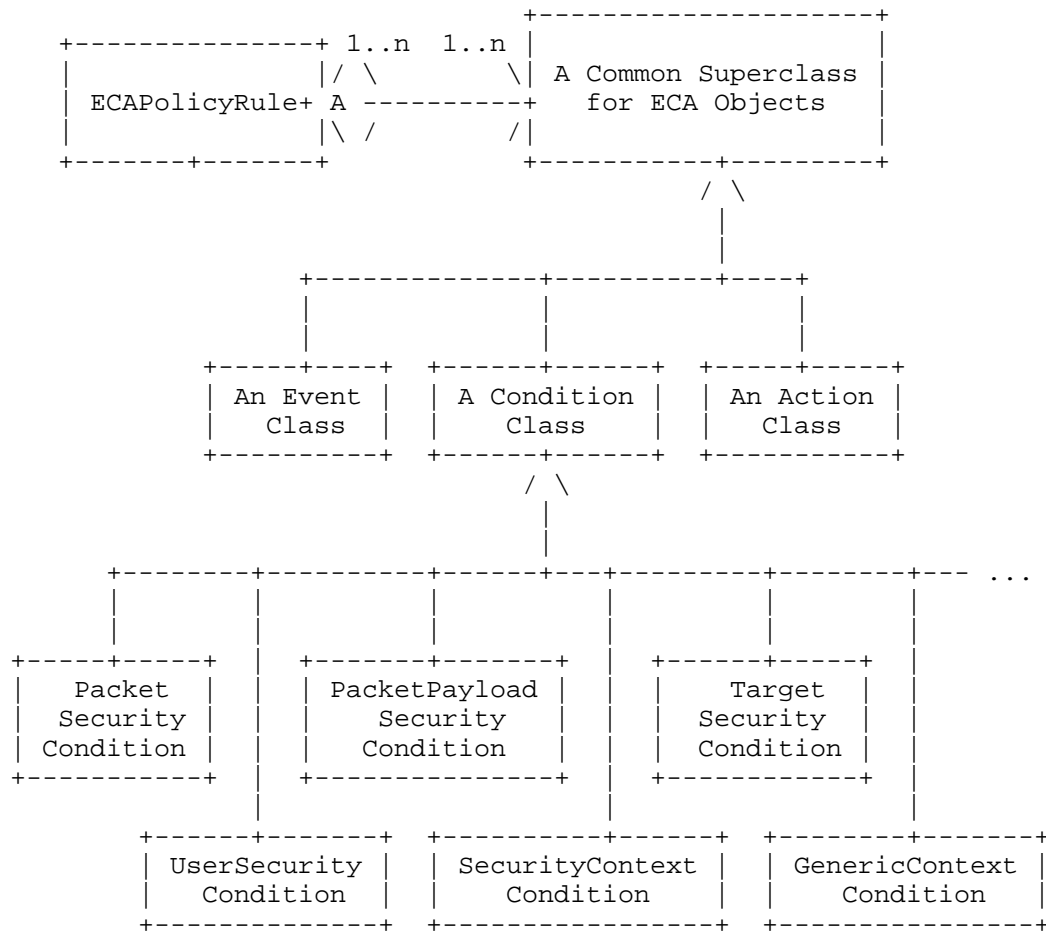


Figure 11. Network Security Info Sub-Model Condition Class Extensions

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.

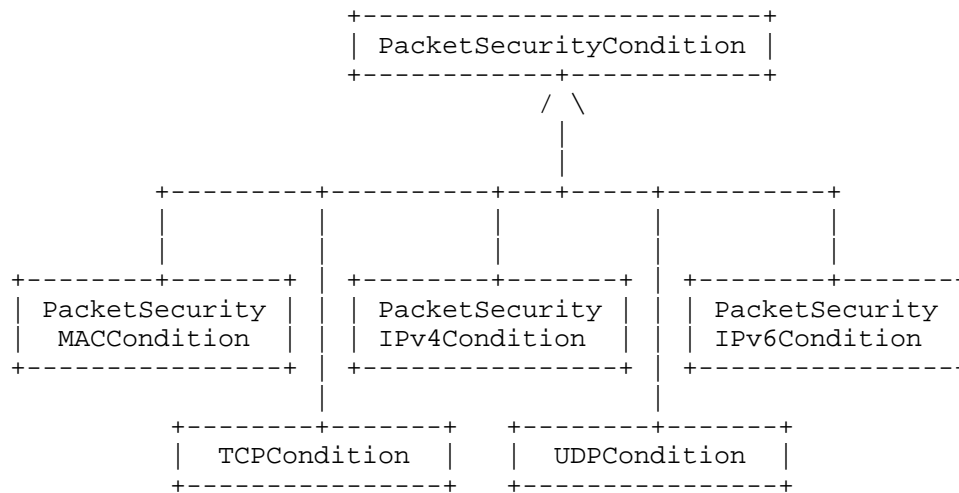


Figure 12. Network Security Info Sub-Model PacketSecurityCondition Class Extensions

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.

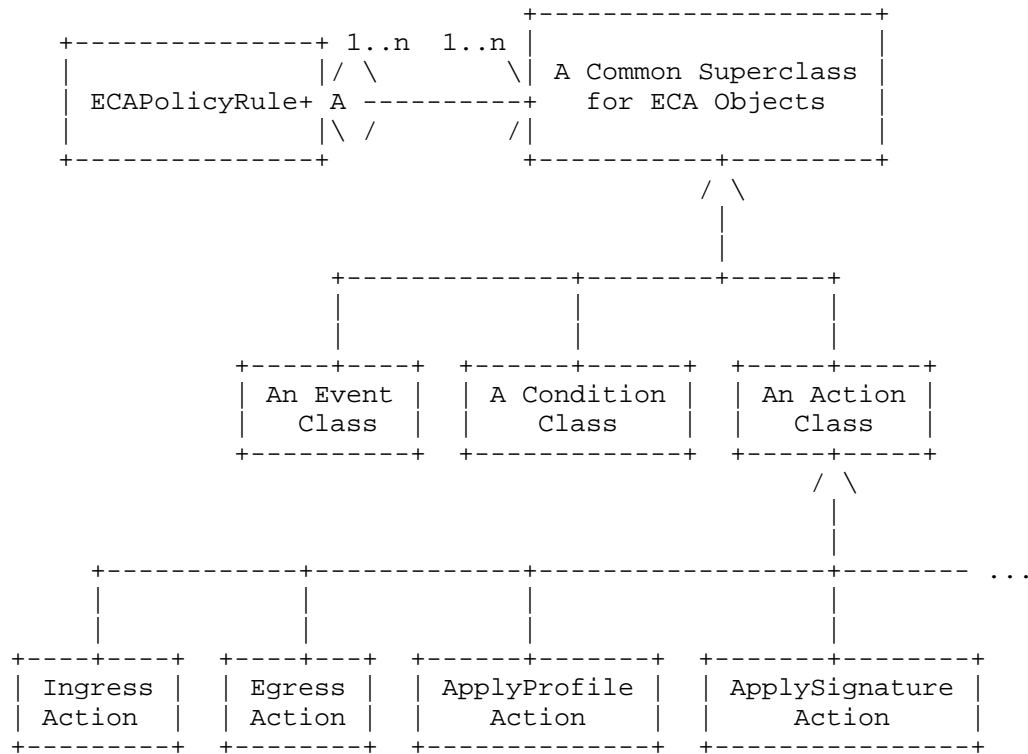


Figure 13. Network Security Info Sub-Model Action Extensions

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:

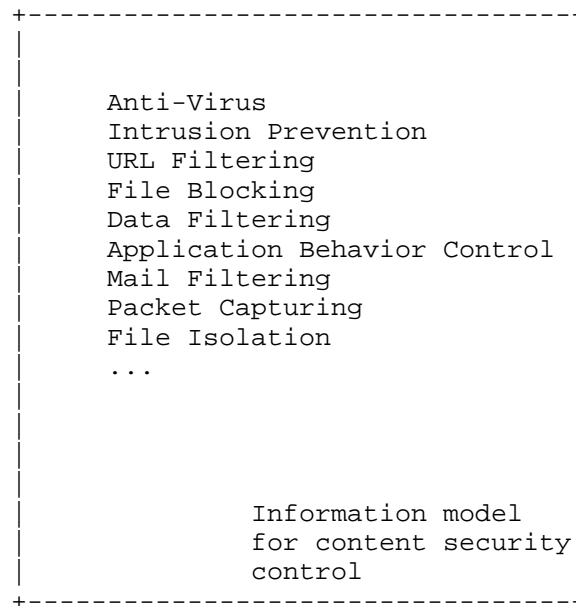


Figure 14. The basic structure of information model for content security control

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.

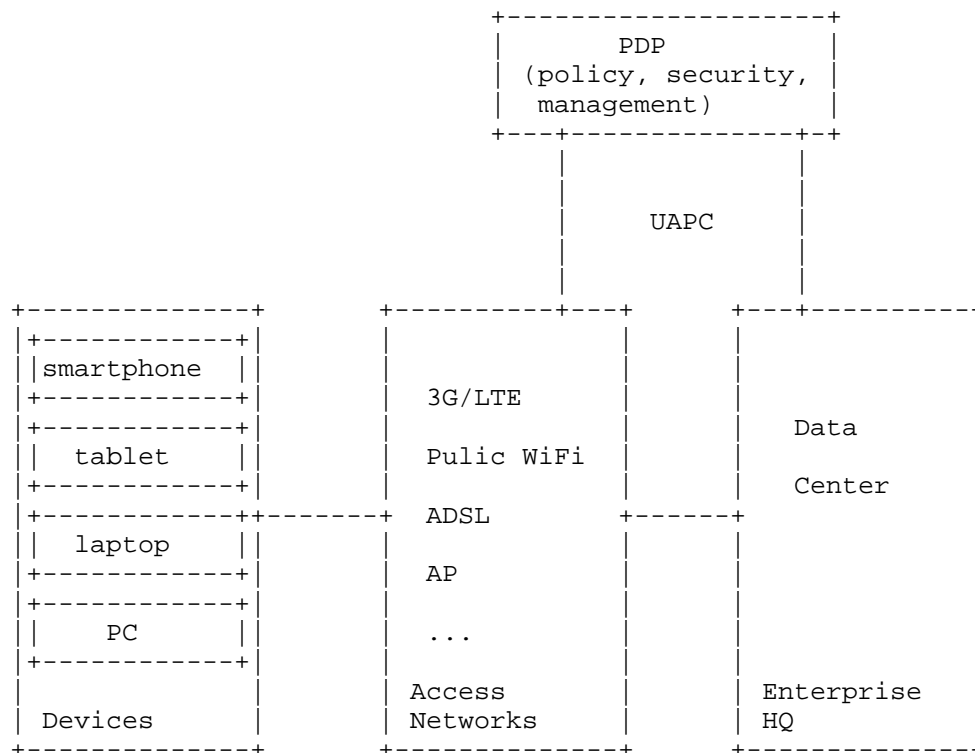


Figure 1: UAPC Framework Example

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.

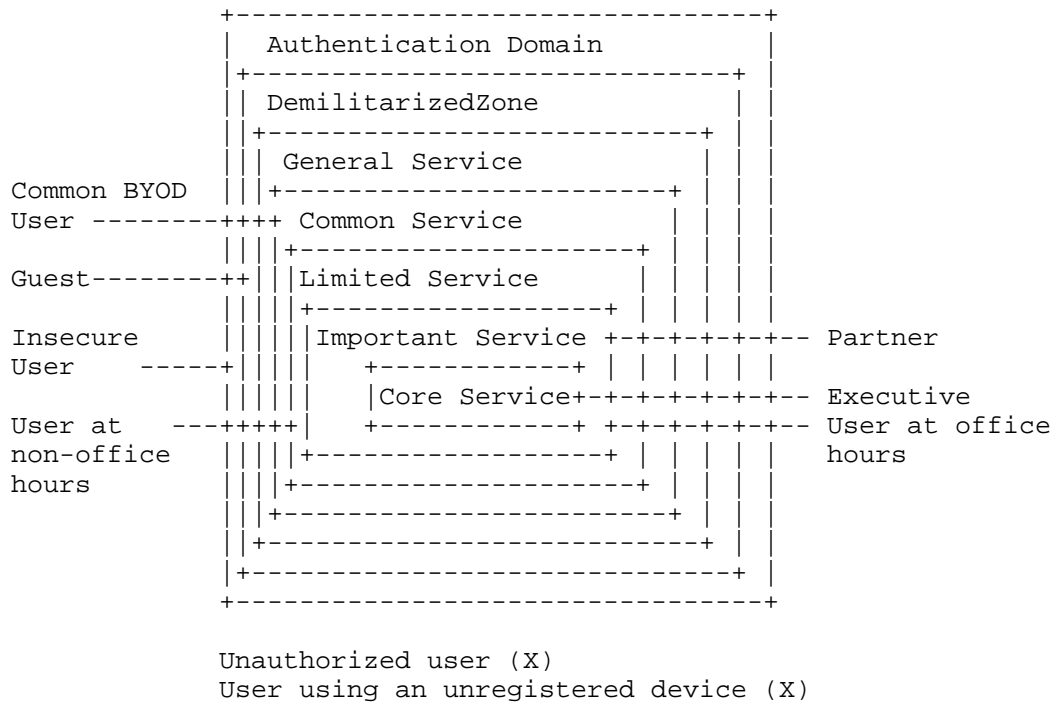


Figure 2: Sample Authorization Rules for User-group Aware Policy Control

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

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 21, 2016

C. Zhou
L. Xia
Huawei Technologies
M. Boucadair
France Telecom
J. Xiong
Huawei Technologies
October 19, 2015

The Capability Interface for Monitoring Network Security Functions (NSF)
in I2NSF
draft-zhou-i2nsf-capability-interface-monitoring-00

Abstract

This document focuses on the monitoring aspects of the flow-based Network Security Functions (NSFs). The NSF Capability interface between the Service Provider's management system (or Security Controller) and the NSFs is meant to enforce the required monitoring capability.

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 April 21, 2016.

Copyright Notice

Copyright (c) 2015 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. The Capability Interface for Monitoring	3
3.1. Overview	3
3.2. Traffic Report	4
3.3. Policy Enforcement	5
4. Security Considerations	5
5. IANA Considerations	5
6. References	6
6.1. Normative References	6
6.2. Informative References	6
Authors' Addresses	6

1. Introduction

Rising security problems bring challenges for the security enterprises and organizations. There are already some software and hardware devices deployed for these problems, e.g., antivirus, firewall, intrusion detection systems (IDS), Web security gateway, and DPI, which build up many safety lines accordingly. However, one individual safety line only defenses the security threats of only one aspect. And more seriously, these safety defense devices are generating large amount of logs and events in the operating procedure, which makes a lot of "information island". With large quantity and isolated security information, it brings low efficiency for the security managers who operate on their own control platform and warning window of various devices.

The network security mechanism would be more efficient if the Security Controller defined in [I-D.merged-i2nsf-framework] could monitor, analyze and investigate the abnormal events and finally produce security reports to the security administrators. The security controller should also be able to collect the traffic and session information from the NSF, in order to steer the suspected attack source or victim traffic to the cleaning center.

The data mining use case defined in [I-D.xia-dots-extended-use-cases] has provided a good example for distributed denial-of-service (DDoS) attack monitoring report. [I-D.reddy-dots-info-model] also describes the information model of flow monitoring to help identify DDOS

attacks in a network. This document aims to cover more cases and more attack types in the network, e.g., botnets, spam, and spyware.

This document describes how to use the capability interface to collect the security information from the NSFs and which security information will be reported using this interface. The protocol and information model will be described for the monitoring aspects of the Capability Interface.

2. Terminology

3. The Capability Interface for Monitoring

3.1. Overview

The capability interface should be able to provide unified event format for the logs and warning information of the overall network, to facilitate the failure discovery and failure locating timely and accurately. With the unified event format, the security managers could run easily and quickly through various security events which guarantees the availability of service information system and service continuity. To achieve this, the Security Controller needs to collect the detailed information of the network status and traffic information from the NSF to make intelligent security decision and to dynamically adjust the sampling and steering policy.

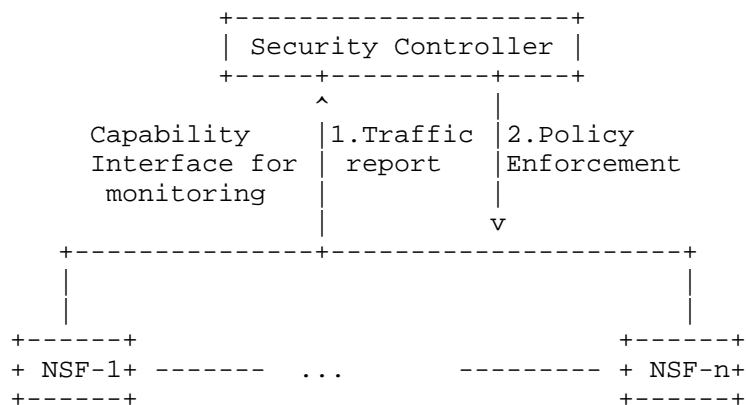


Figure 1: Monitoring Interface

As described in Figure 1, the traffic monitoring procedure involves two network elements: Security Controller (SC) and NSF. The NSF reports the monitoring information to the SC, which provides the specific traffic information, e.g., abnormal flows, security logs, statistics or the suspicious attack sources or destinations. The SC is responsible for monitoring and collecting traffic information from NSFs. Based on the input from the NSF, the SC could make policy enforcement for the specific flows, e.g., traffic steering or adjusting the flow sampling policies.

3.2. Traffic Report

The traffic reported from the NSF may contain the information of IP addresses, security logs, statistics, warnings, and etc. The syslog protocol [RFC5424] could be used to send event notification messages to the SC for traffic collection. The IP Flow Information Export (IPFIX) protocol [RFC5101] may also be adopted for the flow sampling information collection. There are mainly three types of information reported using the capability interface:

- o Traffic Statistics:
 - * Normal traffic statistics based on the source and destination address, including byte per second (bps) and packet per second (pps).
 - * Abnormal traffic statistics based on the source and destination address, including bps and pps.
 - * Traffic statistics based on the network address range (including port usage), including bps and pps.
- o Session Statistics:
 - * Concurrent session statistics based on the source and destination address.
 - * New session statistics based on the source and destination address.
 - * Abnormal session statistics based on the source and destination address, including null link, retransmission session and slow-start link.
- o Abnormal/Attack Event: analysis results of the relevant abnormal/attack event, e.g., time, type, level, detail description, threshold, source IP address and destination IP address.

The NSF could report the accurate source and destination of the attack to the security controller for which to make traffic steering policy, e.g., steering the bad "botnet" traffic to the cleaning center. The type, size and proportion of the abnormal traffic could also be reported to assist the security controller to determine the steering priority, e.g., priority traction, large flow cleaning.

3.3. Policy Enforcement

The Security Controller is responsible for making policy decisions after getting the security information from the NSF (and, typically, other instructions from the operator). It will provide the attack mitigation and defense strategy with the acquired sampling traffic information for attack detection by the way of dynamically adjusting the flow sampling policy, e.g., flow information, sampling ratio, sampling encapsulation method and/or sampling point information. The policies may include: traffic cleaning and sampling adjustment.

- o Traffic cleaning: with the suspicious result of the analyzed sampling traffic, the security controller dynamically sends the steering policy to the related NSF or sends the flow blocking policy to the NSF which is nearest to the attack point, to block the attack traffic from the source. The traffic cleaning policy may include the following three ones:
 - * Access Control List (ACL), to block the attack traffic.
 - * Packet hash digest to block the attack traffic.
 - * Traffic steering policy to clean the traffic.
- o Flow sampling adjustment: after filtering the abnormal object of the sampling traffic, the security controller could dynamically adjust the sampling policy to improve the sampling rate of the TOPN abnormal object, in order to make more accurate attack detection. The abnormal object may include: Top N network address range of the abnormal session, Top N IP address of the abnormal session and the Top N of abnormal sessions.

4. Security Considerations

5. IANA Considerations

This document has no actions for IANA.

6. References

6.1. 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>>.
- [RFC5101] Claise, B., Ed., "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information", RFC 5101, DOI 10.17487/RFC5101, January 2008, <<http://www.rfc-editor.org/info/rfc5101>>.
- [RFC5424] Gerhards, R., "The Syslog Protocol", RFC 5424, DOI 10.17487/RFC5424, March 2009, <<http://www.rfc-editor.org/info/rfc5424>>.

6.2. Informative References

- [I-D.merged-i2nsf-framework]
Lopez, E., Lopez, D., Zhuang, X., Dunbar, L., Parrott, J., Krishnan, R., and S. Durbha, "Framework for Interface to Network Security Functions", June 2015.

Authors' Addresses

Cathy Zhou
Huawei Technologies
Bantian, Longgang District
Shenzhen 518129
P.R. China

Email: cathy.zhou@huawei.com

Liang Xia (Frank)
Huawei Technologies
101 Software Avenue, Yuhuatai District
Nanjing, Jiangsu 210012
P.R. China

Email: Frank.xialiang@huawei.com

Mohamed Boucadair
France Telecom
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Jie Xiong
Huawei Technologies
101 Software Avenue, Yuhuatai District
Nanjing, Jiangsu 210012
P.R. China

Email: emma.xiong@huawei.com