I2NSF WG
Internet-Draft
Intended status: Standards Track
Expires: June 22, 2016

S. Hares
Huawei
D. Zhang

H. Moskowitz
HTT Consulting
H. Rafiee

December 20, 2015

Analysis of Existing work for I2NSF
draft-hares-i2nsf-gap-analysis-01.txt

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on June 22, 2016.

Table of Contents

1. Introduction

This documents provides a gap analysis for I2NSF.

1.1. What is I2NSF

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

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

The Interface to 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:

- 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.

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

For the I2NSF capability layer, the I2NSF work proposes an interoperable protocol that passes NSF provisioning rules and orchestration information between 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).

1.2. Structure of this Document

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

IETF working groups (section 2)

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

OPNFV Open Source Group (section 4)

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

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

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

1.3. Terms and Definitions

1.3.1. Requirements Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119, BCP 14 [RFC2119] and indicate requirement levels for compliant CoAP.

1.3.2. Definitions

NSF: Network security function. An NSF is a function that that detects unwanted activity and blocks/mitigates the effect of such unwanted activity in order to support availability of a network. In addition, the NSF can help in supporting communication stream integrity and confidentiality.

Cloud Data Center (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.

Cloud-based security functions: Network Security Function (NSF) hosted and managed by service providers or different administrative entity.

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

* Client--Provider relationship, i.e. client requesting some network security functions from its provider;

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

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

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

I2NSF 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.

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

I2NSF Management System: I2NSF client operates within an network management system which serves as a collections and distribution point for security provisioning and filter data. This management system is denoted as I2NS management system in this document.

Virtual Security Function: is a security function that can be requested by one domain but may be owned or managed by another domain.

2. IETF Gap analysis

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

2.1. Traffic Filters

2.1.1. Overview

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

- COPS protocol [RFC2748] for controlling access to networks,
- Next steps in Signalling (NSIS) work (architecture: [RFC4080] protocol: [RFC5973]), and
- 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.

2.1.1.1. Data Flow Filters in NETMOD and I2RS

The current work on expanding these filters is focused oncombining 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.

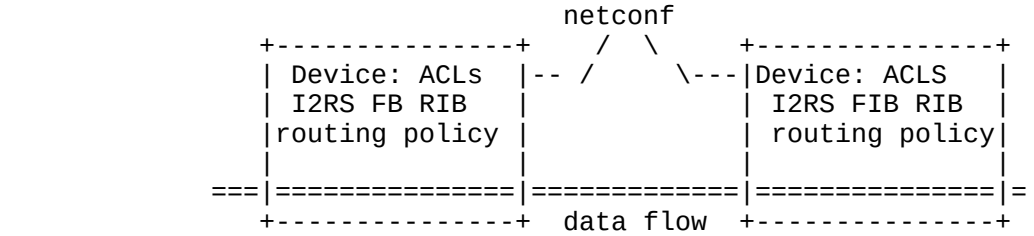


Figure 1

The I2RS protocol is a programmatic interface to the routing system. At this time, the I2RS is targeted to be extensions to the NETCONF/ RESTCONF protocols to allow the NETCONF/RESTCONF protocol to support a highly programmatic interface with high bandwidth of data, highly reliable notifications, and ephemeral state (see [I-D.ietf-i2rs-architecture]). 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].

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

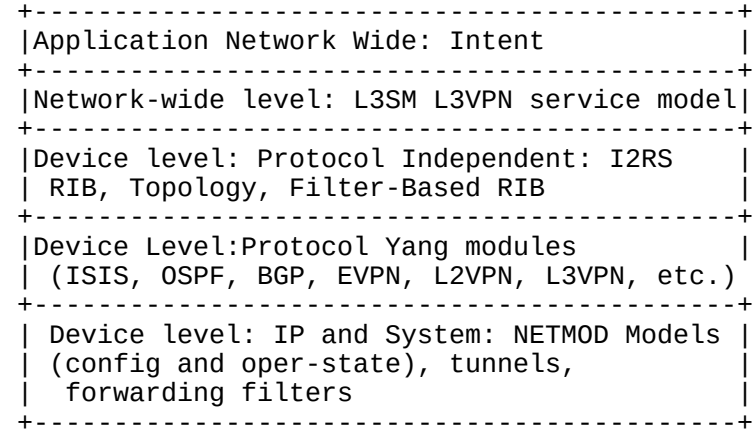


Figure 2 levels of Yang modules

2.1.1.2. I2NSF Gap analysis

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

The I2RS Working group (I2RS WG) is focused on the routing system so 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.

2.1.2. Middle-box Filters

2.1.2.1. Midcom

Midcom Summary: MIDCOM developed the protocols for applications to communicate with middle boxes. However, MIDCOM have not 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

2.1.3. Security Work

2.1.3.1. Overview

Today's NSFs in security devices can handle flow-based security by providing treatment to packets/flows, such as IPS/IDS, Web filtering, flow filtering, deep packet inspection, or pattern matching and 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).

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 x.x.

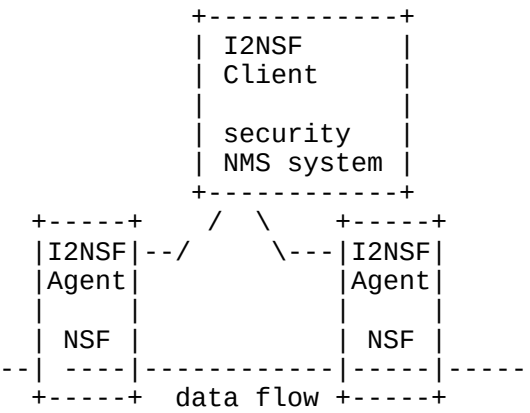


Figure 2

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

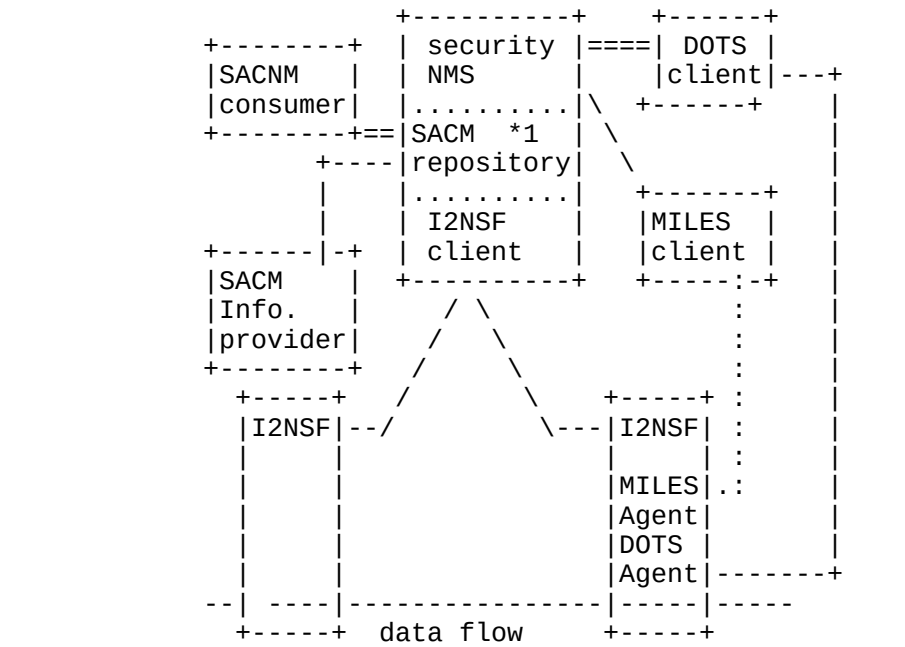
- SACM [I-D.ietf-sacm-architecture] describes an architecture which tries to determine if the end-point security policies and the reality (denoted as security posture) align. [I-D.ietf-sacm-terminology] defines posture as the configuration and/or status of hardware or software on an endpoint as it pertains to an organization's security policy. Filters can be considered on the configuration or status pieces that needs to be monitored.

- 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.

- MILE (Managed Incident LIghtweight Exchange) - is working on creating a standardized format for incident and indicator reports, and creating a protocol to transport this information. The incident information MILE collects may cause changes in data-flow filters on one or more NSFs.

2.1.3.3. I2NSF interaction

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



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

Figure 3

Figure 3 provides a diagram of a system 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:

- 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.

- 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.

- 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.

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.

3. ETSI NFV

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 deployed in carrier networks. One example of these multiple domains is hosting of multiple tenant

domains (telecom service providers) on a single infrastructure domain (infrastructure service) as figure 4 shows. The ETSI Inter inter-VNFM document (aka Ve-Vnfn) between the element management system and the Virtual network function is the equivalent of the interface between the I2NSF client on a management system and the I2NSF agent on the network security feature VNF.

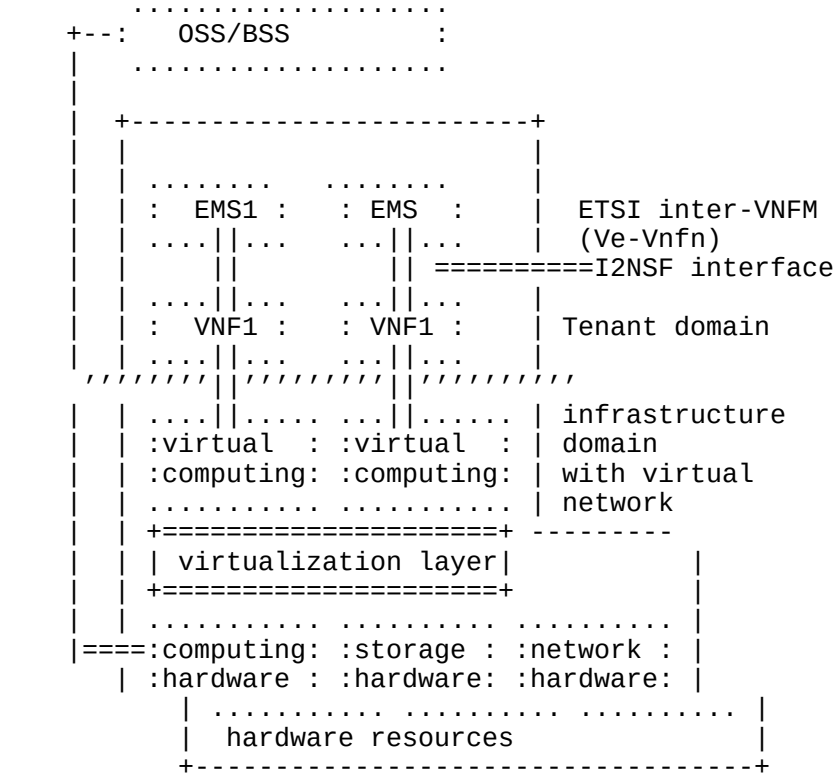


figure 4

The ETSI proof of concept work has worked on the following security proof of concepts:

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

3.2. I2NSF Gap Analysis

The I2NSF will be deployed on top of virtual computing linked together by virtual routers configured by NETCONF/RESTCONF or I2RS which provision and monitoring the L1, L2, l3 and service pathways through the network.

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 in formation.

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.

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.

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 5 shows. This figure also shows how the OPNFV VNF Security project mixes the I2NSF level with the device level.

The Moon project lists the following gaps in OpenStack:

- 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.

- No dynamic control so that if a user obtains the token, the is no way to obtain control over the user.

- No customization or flexibility to allow integration into different vendors,

- 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

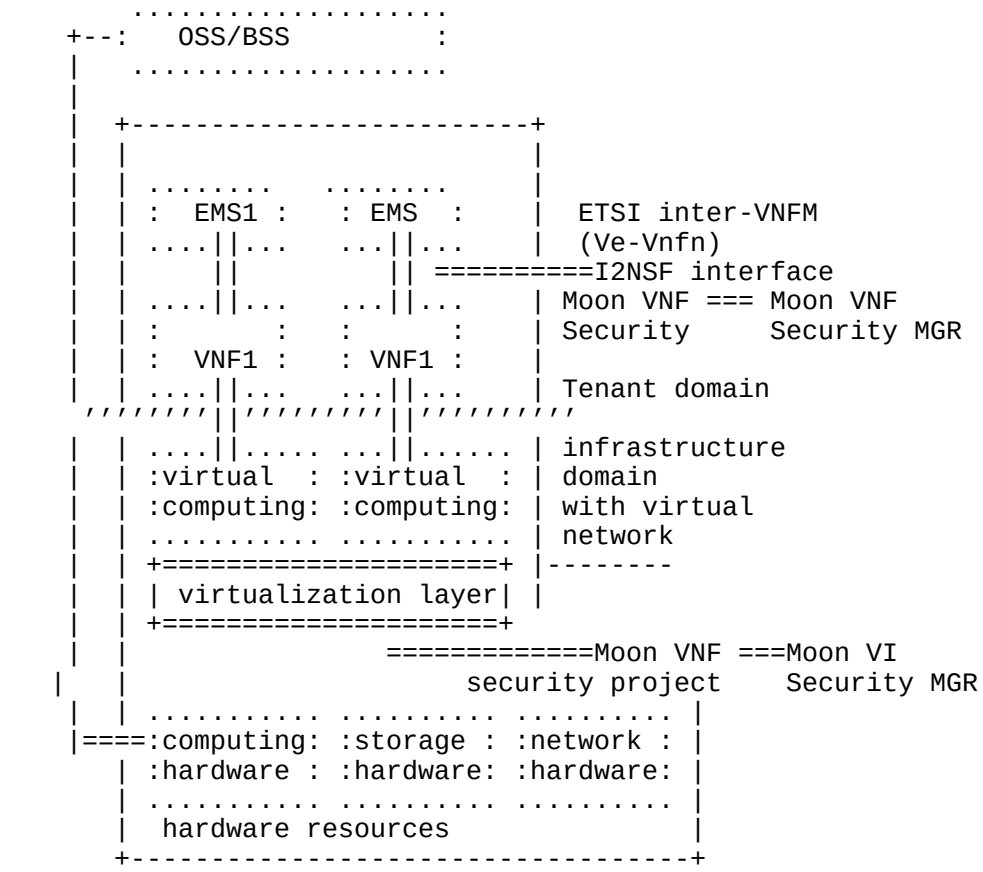


figure 5

4.2. Gap Analysis for OPNFV Moon Project

OpenStack congress does not provide vendor independent systems.

5. OpenStack Security Firewall

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

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

5.1. Overview of API for Security Group

The security group 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.

5.2. 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:

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

The firewall policies include the following information:

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

The firewall table provides the following information:

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

5.3. I2NSF Gap analysis

The OpenStack work is preliminary (security groups and firewall as a service). This work does not allow any of the existing network security vendors provide a management interface. 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.

6. CSA Secure Cloud

6.1. CSA Overview

The Cloud Security Alliance (CSA)(www.cloudsecurityaliance.org) defined security as a service (SaaS) in their Security as a Service working group (SaaS WG) during 2010-2012. The CSA SaaS group defined ten categories of network security (https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_V1_0.pdf) and provides implementation guidance for each of

these ten categories This section provides an overview of the CSA SaaS working groups documentation and a Gap analysis for I2NSF

6.1.1. CSA Security as a Service(SaaS)

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

1. Identity Access Management (IAM) (https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_1_IAM_Implementation_Guidance.pdf)

2. Data Loss Prevention (DLP) (https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_2_DLP_Implementation_Guidance.pdf)

3. Web Security (web) (https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_3_Web_Security_Implementation_Guidance.pdf),

4. Email Security (email) (https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_4_Email_Security_Implementation_Guidance.pdf),

5. Security Assessments (https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_5_Security_Assessments_Implementation_Guidance.pdf),

6. Intrusion Management (https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_6_Intrusion_Management_Implementation_Guidance.pdf),

7. Security information and Event Management (https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_7_SIEM_Implementation_Guidance.pdf),

8. Encryption (https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_8_Encryption_Implementation_Guidance.pdf),

9. Business Continuity and Disaster Recovery (BCDR) https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_9_BCDR_Implementation_Guidance.pdf), and

10. Network Security (https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_10_Network_Security_Implementation_Guidance.pdf).

The sections below give an overview these implementation guidances

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:

- Centralized Directory Services,
- Access Management Services,
- Identity Management Services,
- Identity Federation Services,
- Role-Based Access Control Services,
- User Access Certification Services,
- Privileged User and Access Management,
- Separation of Duties Services, and
- 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 a problem. This 2012 implementation report confirms there is a gap with I2NSF

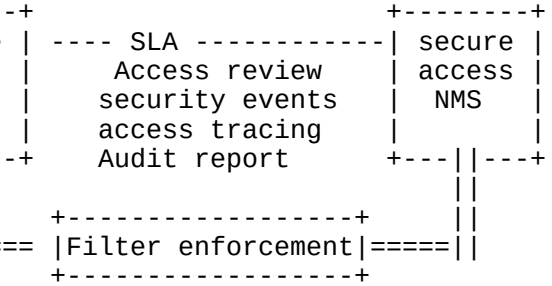


Figure 6

6.1.3. Data Loss Prevention (DLP)

Document:
(https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_2_DLP_Implementation_Guidance.pdf)

The data loss prevention (DLP)services must address:

- origination verification,
- integrity of data,
- confidentiality and access control,
- accountability,
- avoiding false positives on detection, and
- privacy concerns.

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

alert and log data loss,

delete data on system or passing through,

filter out (block/quarantine) data,

reroute data,

encrypt data

Figure 7

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:

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

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

alert and log malware or anti-virus data patterns,

delete data (malware and virus) passing through systems,

filter out (block/quarantine) data,

filter Web URLs,

interact with policy and network management systems,

control bandwidth and QoS of traffic, and

monitor encrypted (SSL enabled) traffic,

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

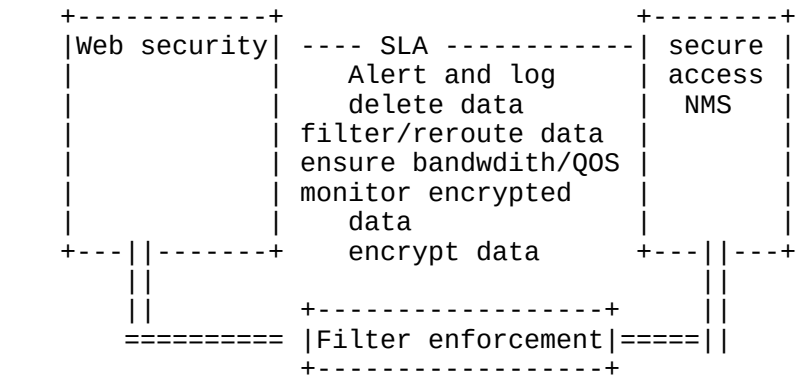


Figure 8

6.1.5. Email Security (email))

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

The CSA Document recommends that email security services must address:

- Common electronic mail components,
- Electronic mail architecture protection,
- Common electronic mail threats,
- Peer authentication,
- Electronic mail message standards,
- Electronic mail encryption and digital signature,
- Electronic mail content inspection and filtering,
- Securing mail clients, and
- 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.

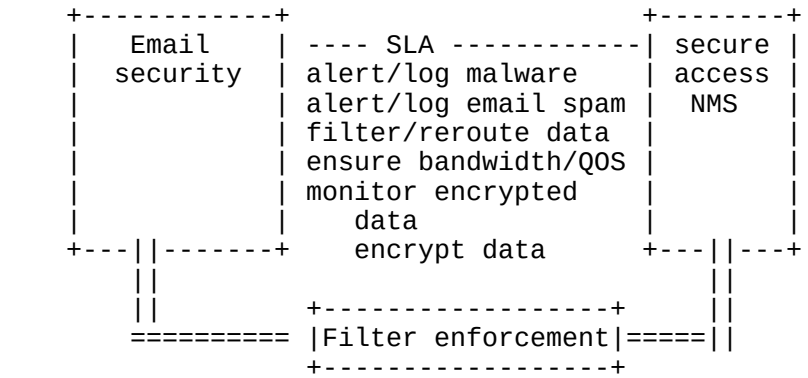


Figure 9

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,

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

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

6.1.7. 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:

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

Intrusion response includes both:

- Automatic, Manual, or Hybrid Mechanisms,
- 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.

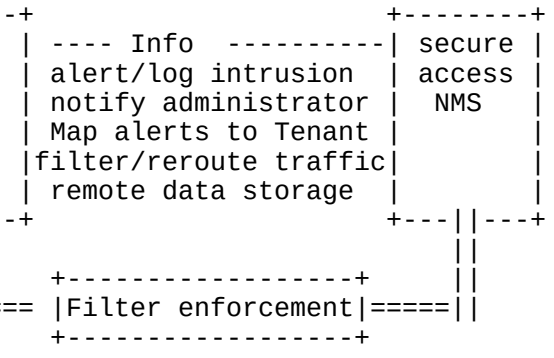


Figure 10

6.1.8. Security Information and Event Management(SEIM)

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

The Security Information and Event Management (SEIM) 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 SEIM 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.

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.

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 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 devices is extremely important to this application. Recovery as a Service (SaaS) indicates that these services need to be able to be remotely management.

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

6.1.11. Network Security Devices

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

The CSA SaaS Network Security implementation recommendation includes advice on:

How to segment networks,

Network security controls,

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

Secure routing and time,

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

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

Threat Management,

Forensic Support, and

Privileged User/Use Monitoring.

These network security systems require provisioning, monitoring, and the ability for the security management system to subscribe to

receive logs, snapshots of capture data, and time synchronization. This document states the following:

> "It is critical to understand what monitoring APIs are available from the CSP, and if they match risk and compliance requirements",
>
> "Network security auditors are challenged by the need to track a server and its identity from creation to deletion. Audit tracking is challenging in even the most mature cloud environments, but the challenges are greatly complicated by cloud server sprawl, the situation where the number of cloud servers being created is growing more quickly than a cloud environments ability to manage them."
>
> A valid threat vector for cloud is the API access. Since a majority of CSPs today support public API interfaces available within their networks and likely over the Internet."

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

6.2. I2NSF Gap Analysis

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

7. In-depth Review of IETF protocols

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

- NETCONF [RFC6242]
- NETCONF monitoring [RFC6022]
- NETCONF over SSH [RFC6242]
- NETCONF over TLS [RFC5539]
- NETCONF system notification> [RFC6470]
- NETCONF access-control (NACM) [RFC6536]
- RESTCONF [I-D.ietf-netconf-restconf]
- NETCONF-RESTCONF call home [I-D.ietf-netconf-call-home]
- RESTCONF collection protocol [I-D.ietf-netconf-restconf-collection]
- NETCONF Zero Touch Provisioning [I-D.ietf-netconf-zerotouch]

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:

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

- o [I-D.ietf-i2rs-pub-sub-requirements] (Publication-Subscription notifications,

- o [I-D.ietf-i2rs-traceability]

- o [I-D.ietf-i2rs-protocol-security-requirements]

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

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]).

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 1. 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.

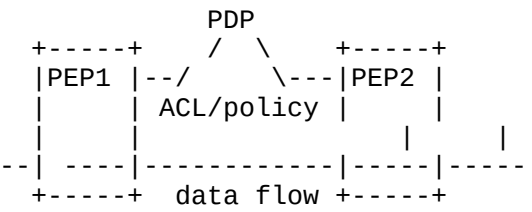


Figure 11

COPS had a design of Policy Enforcement Points (PEP), and policy Decision Points (PDP) as shown in figure 11. These decision points controlled flow from PEP to PEP.

Why COPS is no longer used

Security in the network in 2015 uses specific devices (IDS/IPS, NAT firewall, etc) with specific policies and profiles for each types of device. No common protocol or policy format exists between the policy manager (PDP) and security enforcement points.

COPs RFCs: [RFC4261], [RFC2940], , [RFC3084], , [RFC3483]

Why I2NSF is different COPS

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

7.5. PCP

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

PCP RFCs:

[RFC6887]

[RFC7225]

[I-D.ietf-pcp-authentication]

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

[I-D.ietf-pcp-proxy]

Why is I2NSF different from PCP:

Here are some aspects that I2NSF is different from PCP:

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

- Cover the proxy, firewall and NAT box proposals in I2NSF

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:

- 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.

- 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

- 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 belief I2NSF has a higher chance to be deployed than NSIS:

- 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.

- IETF has the expertise in security architecture and design for interoperable protocols that span controllers/routers, middle-boxes, and security end-systems.

- 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].

8. IANA Considerations

No IANA considerations exist for this document.

9. Security Considerations

No security considerations are involved with a gap analysis.

10. Contributors

The following people have contributed to this document: Hosnieh Rafiee.

11. References

11.1. 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>.

11.2. Informative References

[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-02 (work in progress), October 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-11 (work in progress), December 2015.

[I-D.ietf-i2rs-ephemeral-state]
Haas, J. and S. Hares, "I2RS Ephemeral State Requirements", draft-ietf-i2rs-ephemeral-state-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-protocol-security-requirements]
Hares, S., Migault, D., and J. Halpern, "I2RS Security Related Requirements", draft-ietf-i2rs-protocol-security-requirements-01 (work in progress), September 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-04 (work in progress), November 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-08 (work in progress), October 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-05 (work in progress), December 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-02 (work in progress), December 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-02 (work in progress), October 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

Dacheng Zhang
Beijing
China

Email: dacheng.zdc@aliabab-inc.com

Bob Moskowitz
HTT Consulting
Oak Park, MI 48237

Email: rgm@labs.htt-consult.com

Hosnieh Rafiee
Munich
Germany

Phone: +49(0)17657587575
Email: ietf@rozanak.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.

Table of Contents

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 exisiting 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:

- 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].

- 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].

- Access Control Policy: The set of rules that define the conditions under which access may take place [ITU-T.X.800].

- Access Control Policy Rules: Security policy rules concerning the provision of the access control service [ITU-T.X.800].

- 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.

- 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.

- 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.

- 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.

- 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.

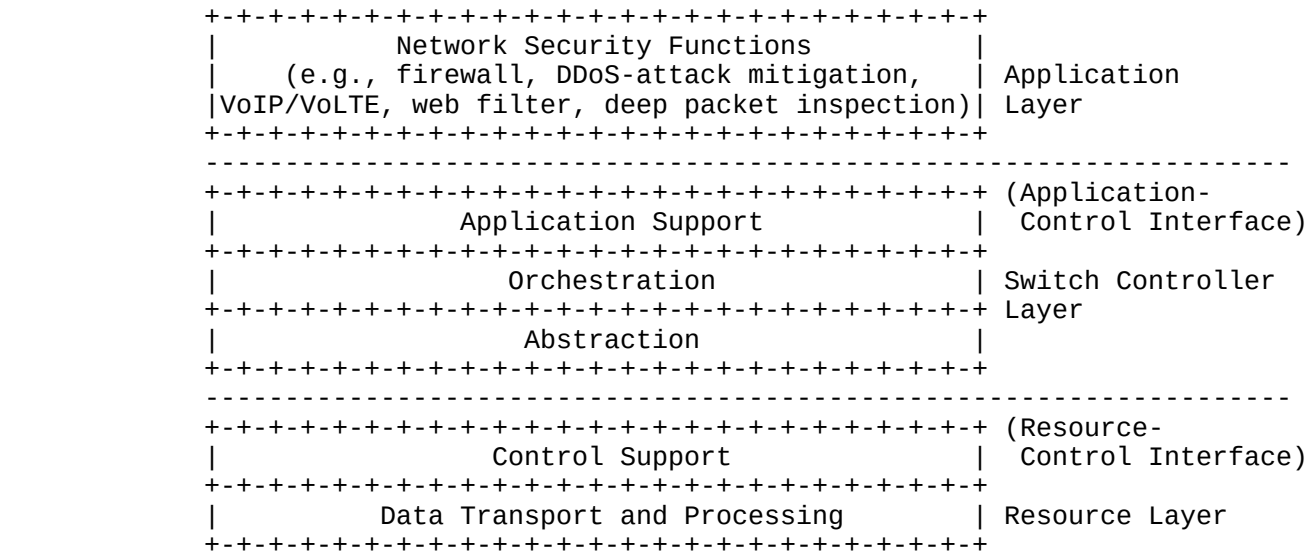


Figure 1: High-level Architecture for SDN-based Security Services

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-mgnt-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

- Prompt reaction to new network attacks: SDN-based security services allow private networks to defend themselves against new sophisticated network attacks.

- 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.

- 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.

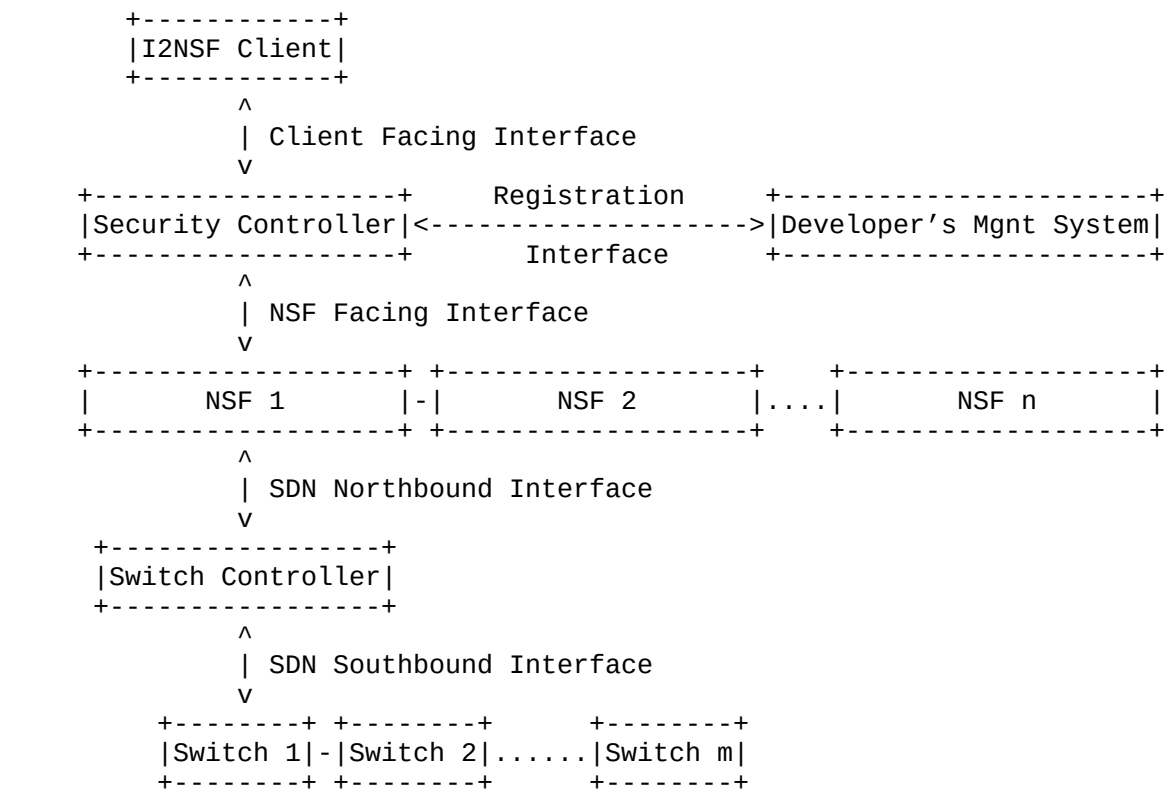


Figure 2: A Framework for SDN-based Security Services using I2NSF

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:

- SDN-based security services are required to support the programmability of network resources to mitigate network attacks.
- SDN-based security services are required to support the orchestration of network resources and SDN applications to mitigate network attacks.
- SDN-based security services are required to provide an application interface allowing the management of access control policies in an autonomous and prompt manner.

- SDN-based security services are required to provide a resource-control interface for the control of network resources to mitigate network attacks.

- SDN-based security services are required to provide the logically centralized control of network resources to mitigate network attacks.

- SDN-based security services are required to support the seamless services to mitigate network attacks.

- 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:

- 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.

- 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.

- 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.

- 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.

- 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 managenent 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:

- 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.

- 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.

- 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.

- 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:

- 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.

- 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.

- 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.

- 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-mgnt-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:

- 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.

- Three use cases described in this document can use a data model corresponding to the information model for the I2NSF capability interface.

- The proposed framework of SDN-based security services can be combined to a security management architecture for handling security policies.

- 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 will expire on September 16, 2016.

Copyright Notice

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

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 NSFs

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 NSFs 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 NSFs utilize profile and/or signature files to achieve more effective threat detection and prevention. It is not uncommon for a NSF to apply different profiles and/or signatures for different flows. Some profiles/signatures do not require any knowledge of past or future activities, while others are stateful, and may need to maintain state for a specific length of time.

The functional profile or signature file is one of the key properties that determine the effectiveness of the NSF, and is mostly 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 NSFs

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

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

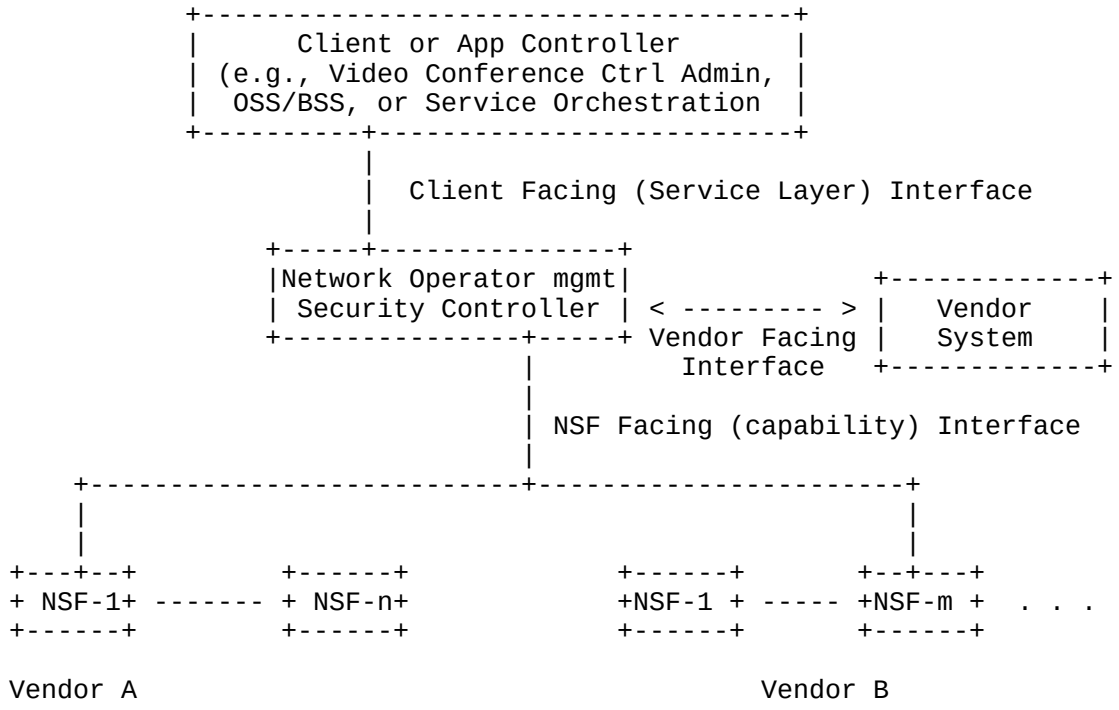


Figure 1: Multiple Interfaces

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 polices to the corresponding instantiations of the NSF, as shown in the Figure 2 below.
- Policies to one vNSF may need to be retrieved and moved to another vNSF of the same type when client flows are moved from one vNSF to another.
- Multiple vNSFs may share the same physical platform
- There may be scenarios where multiple vNSFs collectively perform the security policies needed.

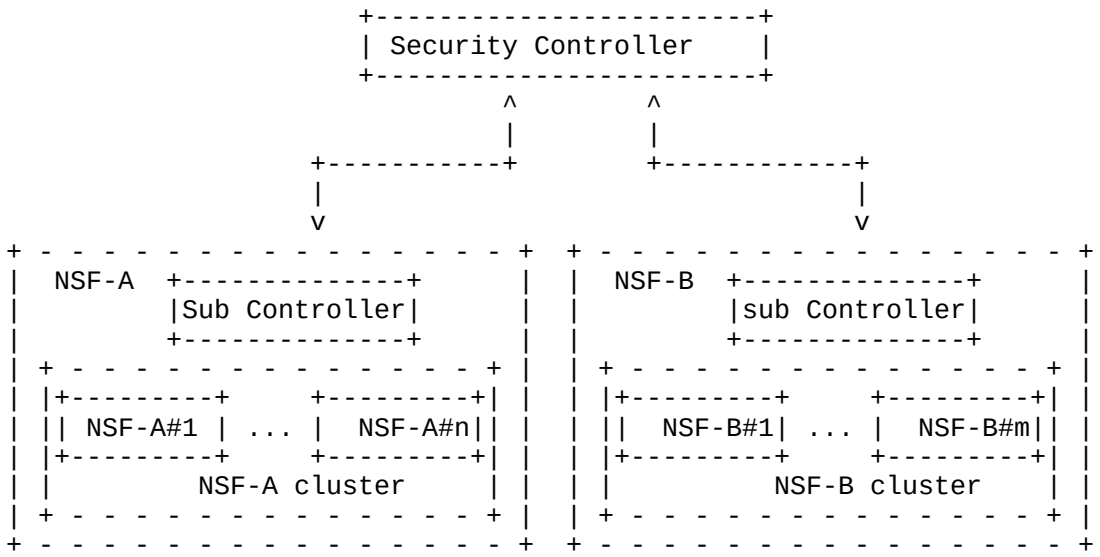


Figure 2: Cluster of NSF Instantiations Management

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 NSFs using Packet Content Match categories. The IDR Flow Specification [RFC5575] has specified 12 different packet header matching types. More packet content matching types have been proposed in the IDR WG. I2NSF should re-use the packet matching types being specified as much as possible. More matching types might be added for Flow-based NSFS. Tables 1-4 below list the applicable packet content categories that can be potentially used as packet matching types by Flow-based NSFs:

Packet Content Matching Capability Index	
Layer 2 Header	Layer 2 header fields: Source/Destination/s-VID/c-VID/EtherType/.
Layer 3	Layer header fields:
IPv4 Header	protocol dest port src port src address dest address dscp length flags ttl
IPv6 Header	addr protocol/nh src port dest port src address dest address length traffic class hop limit flow label dscp
TCP SCTP DCCP	Port syn ack fin rst ? psh

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

Table 1: Subject Capability Index

context	matching Capability Index
Session	Session state, bidirectional state
Time	time span, time occurrence
Events	Event URL, variables
Location	Text string, GPS coords, URL
Connection Type	Internet (unsecured), Internet (secured by VPN, etc.), Intranet, ...

Direction	Inbound, Outbound
State	Authentication State
	Authorization State
	Accounting State
	Session State

Table 2: Object Capability Index

Action Capability Index	
Ingress port	SFC header termination, VxLAN header termination
Actions	Pass Deny Mirror Simple Statistics: Count (X min; Day;..) Client specified Functions: URL
Egress	Encap SFC, VxLAN, or other header

Table 3: Action Capability Index

Functional profile Index	
Profile types Signature	Name, type, or Flexible Profile/signature URL Command for Controller to enable/disable

Table 4: Function Capability Index

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:

- Pass for Subscriber "xxx"
- 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 NSFs (by different vendors) in a network.
  - Third party system

- Security functions send requests for more sophisticated functions upon detecting something suspicious, usually via a security controller.

9. Manageability Considerations

Management of NSFs usually includes:

- life cycle management and resource management of NSFs
- configuration of devices, such as address configuration, device internal attributes configuration, etc,
- signaling, and
- policy rules provisioning.

I2NSF will only focus on the policy rule provisioning part, i.e., the last bullet listed above.

10. Security Considerations

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

11. IANA Considerations

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

12. References

12.1. Normative References

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

[RFC3060] Moore, B, et al, "Policy Core Information Model (PCIM)", RFC 3060, Feb 2001.

[RFC3460] Moore, B. "Policy Core Information Model (PCIM) Extensions", RFC3460, Jan 2003.

[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 Virtualizsation (NFV) Use Cases. ETSI GS NFV 001v1.1.1, 2013.

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

[SC-MobileNetwork] W. Haeffner, N. Leymann, "Network Based Services in Mobile Network", IETF87 Berlin, July 29, 2013.

[I2NSF-Demo] Y. Xie, et al, "Interface to Network Security Functions Demo Outline Design", <draft-xie-i2nsf-demo-outline-design-00>, April 2015.

[ITU-T-X1036] ITU-T Recommendation X.1036, "Framework for creation, storage, distribution and enforcement of policies for network security", Nov 2007.

13. Acknowledgments

Acknowledgements to xxx for his review and contributions.

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

Authors' Addresses

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

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

XiaoJun Zhuang
China Mobile
Email: zhuangxiaojun@chinamobile.com

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

John Strassner
Huawei
John.sc.Strassner@huawei.com

Joe Parrott
BT
Email: joe.parrott@bt.com

Ramki Krishnan
Dell
Email: ramki_krishnan@dell.com

Seetharama Rao Durbha
CableLabs
Email: S.Durbha@cablelabs.com

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

Copyright Notice

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

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:

- 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:

- 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);

- Condition: A set of attributes, features, and/or values that are to be compared with a set of known attributes, features, and/or values in order to make a decision. When used in the context of policy rules for I2NSF, it is used to determine whether or not the set of Actions in that Policy Rule can be executed or not. The following are exemplary types of conditions:

  - Packet content values: Refer to the kind of information or attributes acquired directly from the packet headers or payloads that can be used in the security policy. It can be any fields or attributes in the packet L2/L3/L4 header, or special segment of bytes in the packet payload;

  - Context values: Refer to the context information for the received packets. It can be (and not limited to):

* User: The user (or user group) information to which a network flow is associated. A user has many attributes, such as name, id, password, authentication mode, and so on. The combination of name and id (where id could be a password, a certificate, or other means of identifying the user) is often used in the security policy to identify the user. For example, if an NSF is aware of the IP (or MAC) address associated with the user, the NSF can use a pre-defined or dynamically learned name-address association to enforce the security functions for this given user (or user group);

* Schedule: Time or time range when packet or flow is received;

* Region: The geographic location where network traffic is received;

* Target: The target indicates the entity to which the security services are applied. This can be a service, application, or device. A service is identified by the protocol type and/or port number. An application is a computer program for a specific task or purpose. It provides additional semantics (e.g., dependencies between services) for matching traffic. A device is a managed entity that is connected to the network. The attributes that can identify a device include type (e.g., router, switch, pc) and operating system (e.g., Windows, Linux, or Android), as well as the device's owner;

* State: It refers to various states to which the network flow is associated. It can be either the TCP session state (e.g., new, established, related, invalid, or untracked), the session AAA state (e.g., authenticated but not authorized), or the access mode of the device (e.g., wireline, wireless, or cellular; these could be augmented with additional attributes, such as the type of VPN that is being used);

* Direction: the direction of the network flow.

o Action: NSFs provide security functions by executing various Actions, which at least includes:

- Ingress actions, such as pass, drop, mirroring, etc;

- Egress actions, such as invoke signaling, tunnel encapsulation, packet forwarding and/or transformation;

- Applying a specific Functional Profile or signature - e.g., an IPS Profile, a signature file, an anti-virus file, or a URL filtering file. The functional profile or signature file defines the security capabilities for content security control and/or attack mitigation control; these will be described in sections 3.2 and 3.3, respectively. It is one of the key properties that determine the effectiveness of the NSF, and is mostly vendor-specific today. One goal of I2NSF is to standardize the form and functional interface of those security capabilities while supporting vendor-specific implementations of each.

The above ECA ruleset is very general and easily extensible, thus can avoid any potential constraints which could limit the implementation of the network security control capability.

3.2. Content Security

Content security is another category of security capabilities applied to application layer. Through detecting the contents carried over the traffic in application layer, these capabilities can realize various security functions, such as defending against intrusion, inspecting virus, filtering malicious URL or junk email, blocking illegal web access or malicious data retrieval.

Generally, each type of threat in the application layer has a set of unique characteristics, and requires handling with a set of specific methods. Thus, it can be thought of as a logically independent security capability. Since there are a large number of types of threats in the application layer, as well as new types of threats that occur quickly, there will be a large number of security capabilities. Therefore, some basic principles for security capability management and utilization need to be considered:

o Flexibility: each security capability should be an independent function, with minimum overlap or dependency to other capabilities. This enables each security capability to be utilized and assembled together freely. More importantly, changes to one capability will not affect other capabilities;

- o High level of abstraction: this enables each capability to have a unified interface to make it programmable; this in turn provides a standardized ability to describe and report its processing results and corresponding statistics information. Furthermore, it facilitates the multi-vendor interoperability;

- o Scalability: The system must have the capability to scale up/down or scale in/out. Thus, it can meet various performance requirements derived from changeable network traffic or service requests. In addition, the security capability must support reporting statistics to the security controller to assist its decision on whether it needs to invoke scaling or not;

- o Automation: The system must have the ability to auto-discover, auto-negotiate, and auto-update security capabilities. These features are especially useful for the management of a large number of NSFs.

Based on the above principles, a set of abstract and vendor-neutral capabilities with standard interfaces is needed. The security controller can compare the requirements of clients to the set of capabilities that are currently available in order to choose which NSFs are needed to meet those requirements. Note that this choice is independent of vendor, and instead relies specifically on the capabilities (i.e., the description) of the functions provided. This also facilitates the customization of the functionality of the selected NSFs by setting the parameters of their interfaces. This category of security capability abstracts security as a black box that has selectable features compared with current network security control mechanisms.

Furthermore, when an unknown threat (e.g., zero-day exploits, unknown malware, and APTs) is reported by a network security device, new capabilities may be created, and/or existing capabilities may be updated (e.g., signature and algorithm), to correspond to the new functionality provided by the NSF to handle the threat. The new capabilities are provided from different vendors after their analysis of the new threats and subsequent installation of the functions required to report on (and possibly mitigate) the threat. New capabilities may be sent to and stored in a centralized repository, or stored separately in a local repository. In either case, a standard interface is needed during this automated update process.

3.3. Attack Mitigation

This category of security capabilities is used to detect and mitigate various types of network attacks. Today's common network attacks can be classified into the following sets, and each set further consists of a number of specific attacks:

o DDoS attacks:

- Network layer DDoS attacks: Examples include SYN flood, UDP flood, ICMP flood, IP fragment flood, IPv6 Routing header attack, and IPv6 duplicate address detection attack;

- Application layer DDoS attacks: Examples include http flood, https flood, cache-bypass http floods, WordPress XML RPC floods, ssl DDoS.

o Single-packet attack:

- Scanning and sniffing attacks: IP sweep, port scanning, etc

- malformed packet attacks: Ping of Death, Teardrop, etc

- special packet attacks: Oversized ICMP, Tracert, IP timestamp option packets, etc

Each type of network attack has its own network behaviors and packet/flow characteristics. Therefore, each type of attack needs a special security function, which is advertised as a capability, for detection and mitigation.

Overall, the implementation and management of this category of security capabilities of attack mitigation control is very similar to content security control. A standard interface, through which the security controller can choose and customize the given security capabilities according to specific requirements, is essential.

4. Information Model Design

4.1. Overall Structure

The I2NSF capability interface is in charge of controlling and monitoring the NSFs. This is done using the following approach:

1) User of the capability interface selects the set of capabilities required to meet the needs of the application;

2) A management entity uses the information model to match chosen capabilities to NSFs, independent of vendor;

3) A management entity takes the above information and creates or uses vendor-specific data models to install the NSFs identified by the chosen capabilities;

4) Control and monitoring can then begin.

Based on the analysis above, the information model should consist of at least four sections: capability, network security, content security and attack mitigation. This assumes that an external model, or set of models, is used to define the concept of an ECA Policy Rule and its components (e.g., Event, Condition, and Action objects).

Since Capabilities are determined by the management system, and are not inherent characteristics that differentiate objects, it is also assumed that an external model (or set of models) will define a generic metadata concept. Capabilities are then sub-classed from an appropriate class in the external metadata model.

The capability interface is used for advertising, creating, selecting and managing a set of specific security capabilities independent of the type and vendor of device that contains the NSF. That is, the user of the capability interface does not care whether the NSF is virtualized or hosted in a physical device, the vendor of the NSF, and which set of entities the NSF is communicating with (e.g., a firewall or an IPS). Instead, the user only cares about the set of capabilities that the NSF has, such as packet filtering or deep packet inspection. The overall structure is illustrated in the figure below:

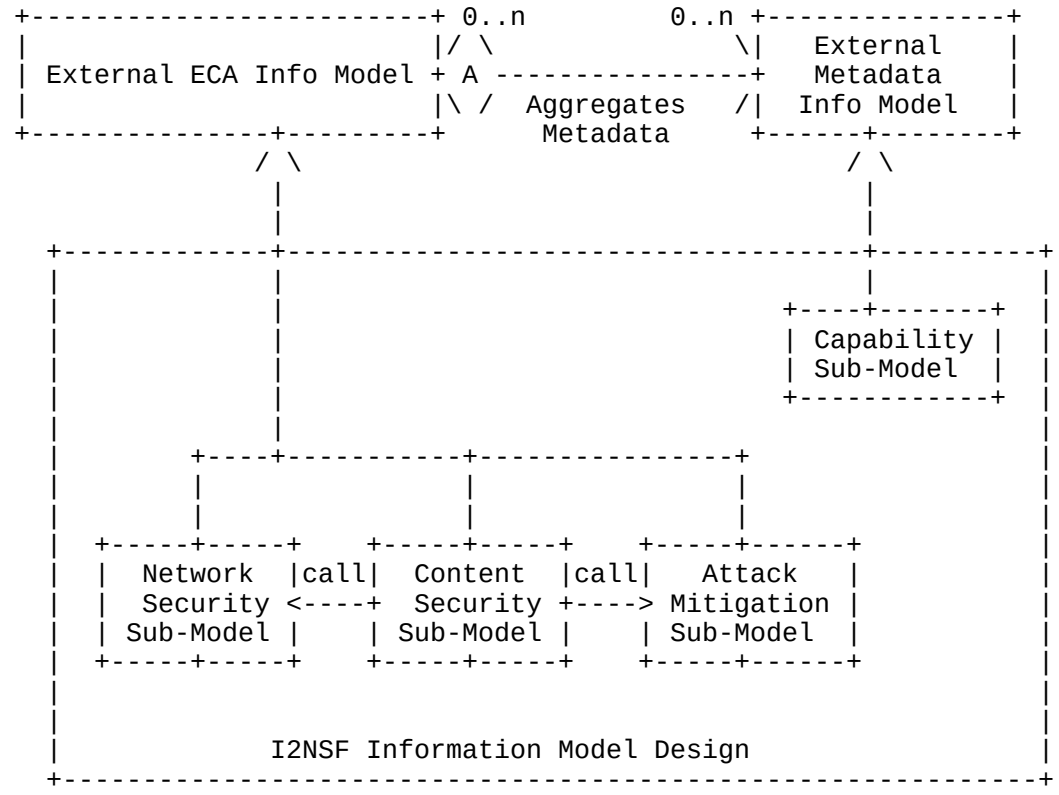


Figure 1. The Overall I2NSF Information Model Design

As illustrated in Figure 1, the network security function is the key. It usually runs as the first step to handle traffic (e.g., packet/flow detection and filtering, etc.) over the network layer. The framework portion of the information model ensures that each of the three domain sub-models (content security, network security, and attack mitigation) can function in collaboration or independently.

The content security and attack mitigation sub-models can be enforced on demand (i.e., once or recursively based on the results of network security function).

This draft defines the four sub-models inside the I2NSF information model shown in Figure 1.This model assumes that another, generic, information model for defining ECA policy rules exists outside of I2NSF. Hence, the Network Security, Content Security, and Attack Mitigation Sub-Models each extend the generic external ECA model to form security policy rules.

It also assumes that Capabilities are modeled as metadata, since a Capability is something that describes and/or prescribes functionality about an object, but is not an inherent part of that object. Hence, the Security Capability Sub-Model extends the generic external metadata model.

Both of these external models could, but do not have to, draw from the SUPA model [I-D.draft-ietf-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:

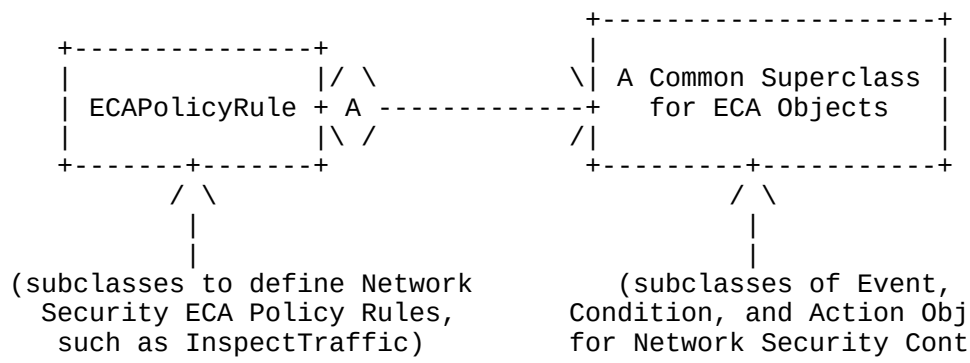


Figure 2. Network Security Information Sub-Model Overview

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.

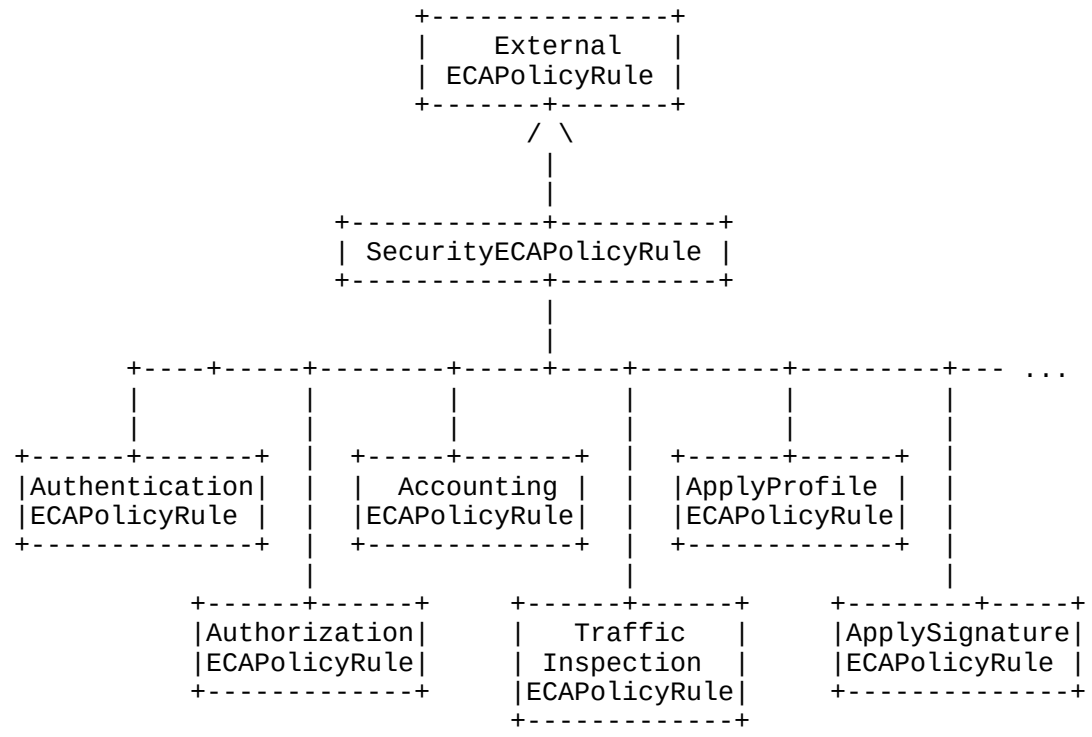


Figure 3. Network Security Info Sub-Model ECAPolicyRule Extensions

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.

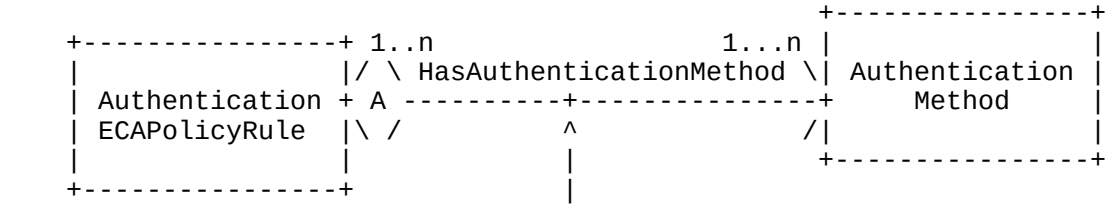
Brief class descriptions of these six ECA Policy Rules are provided in the following sub-sections. Note that there is a common pattern that defines how these ECAPolicyRules operate; this simplifies their implementation. All of these six ECA Policy Rules are concrete classes.

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:



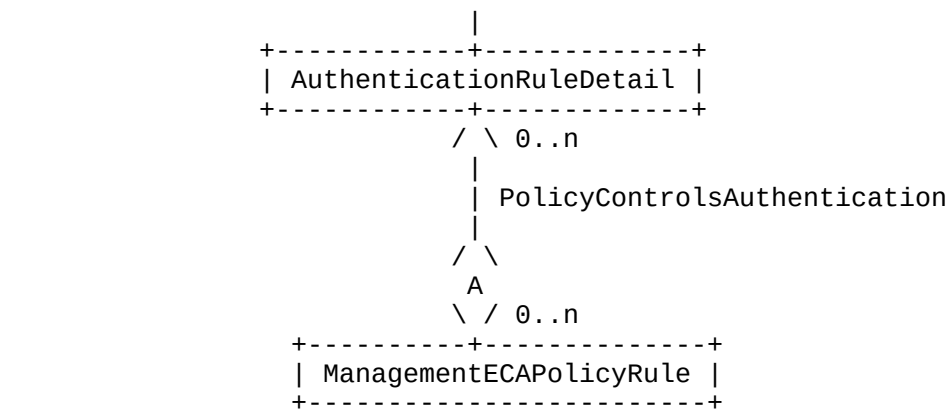


Figure 4. Modeling Authentication Mechanisms

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 externalAuthenticationMethod 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:

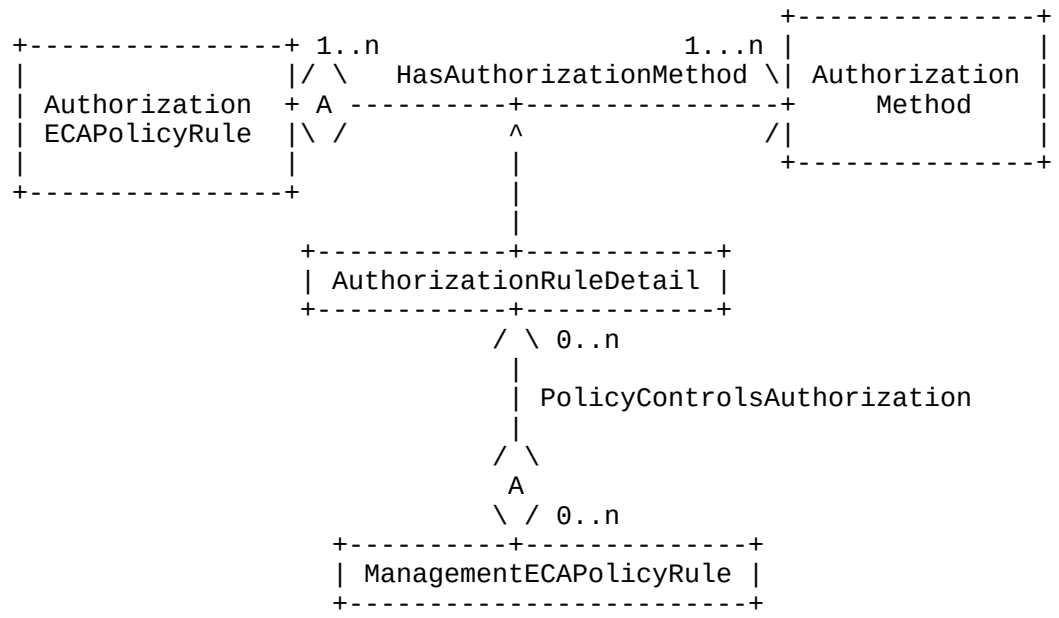


Figure 5. Modeling Authorization Mechanisms

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:

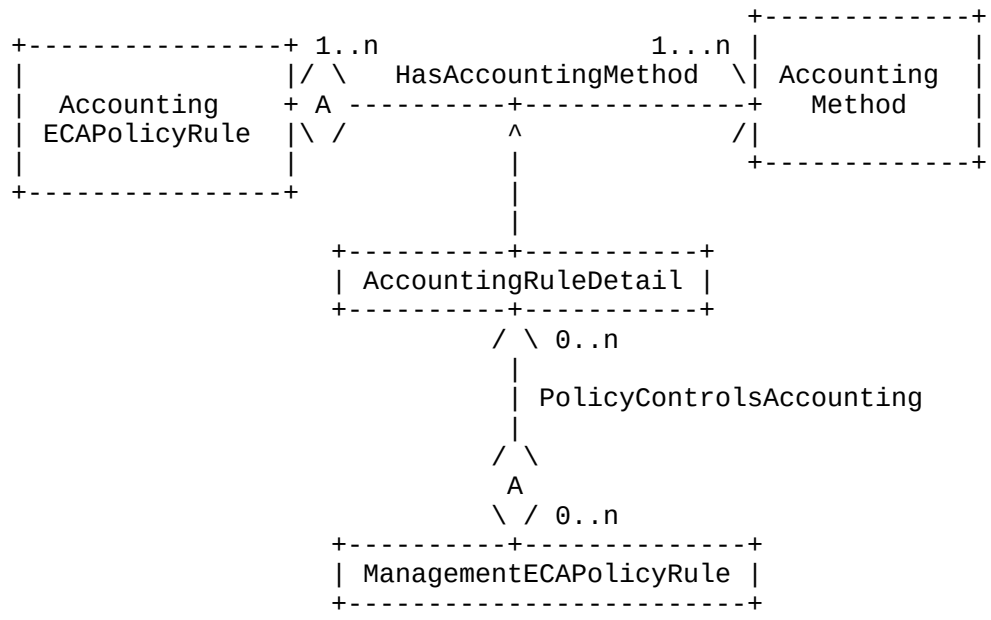


Figure 6. Modeling Accounting Mechanisms

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:

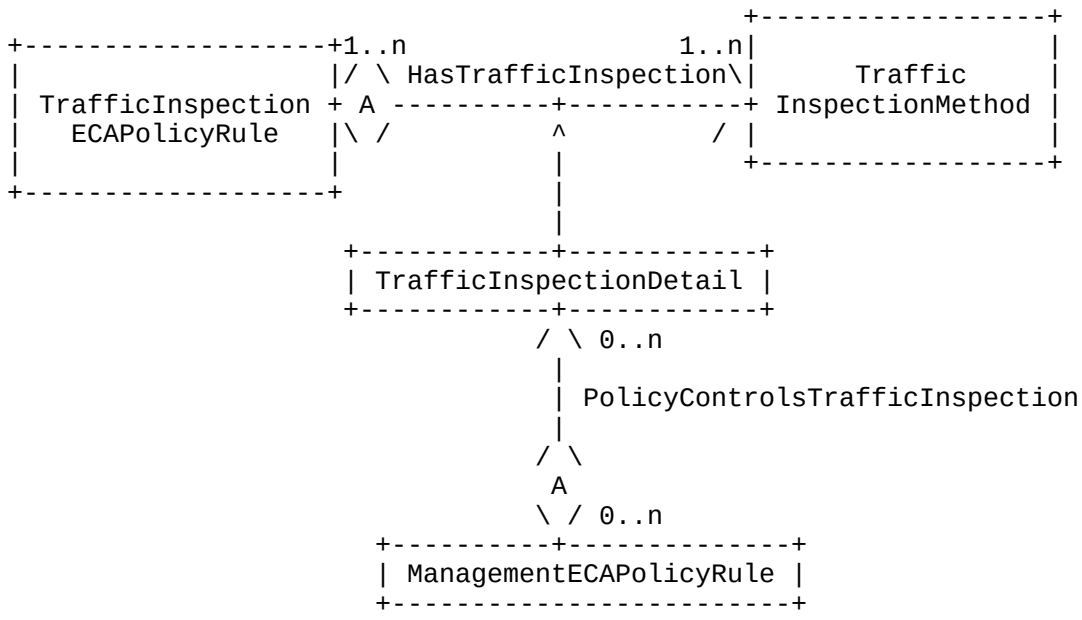


Figure 7. Modeling Traffic Inspection Mechanisms

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:

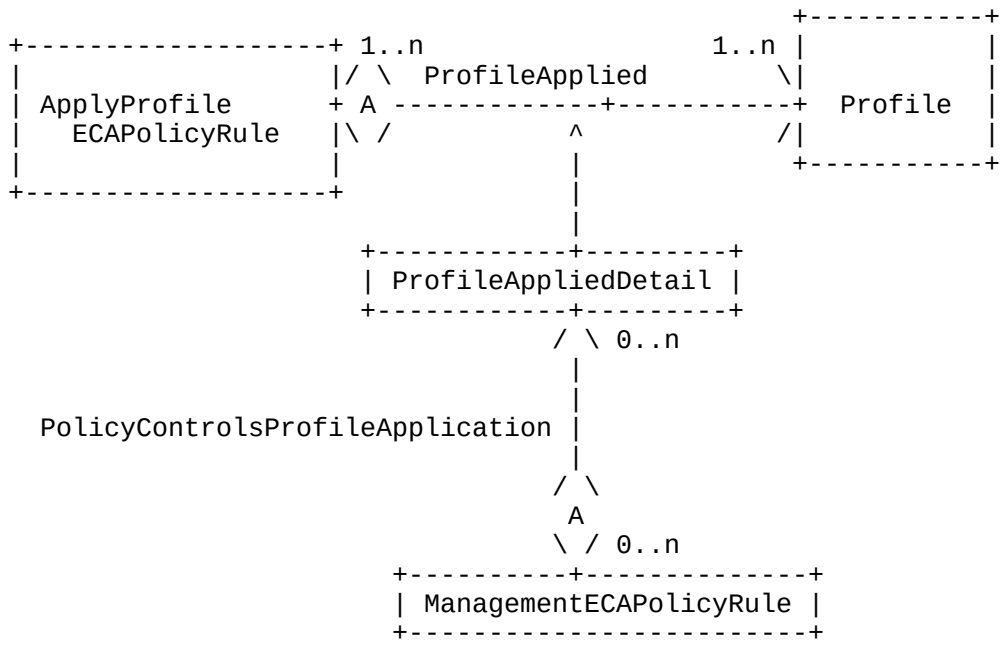


Figure 8. Modeling Profile ApplicationMechanisms

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 Profileis 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:

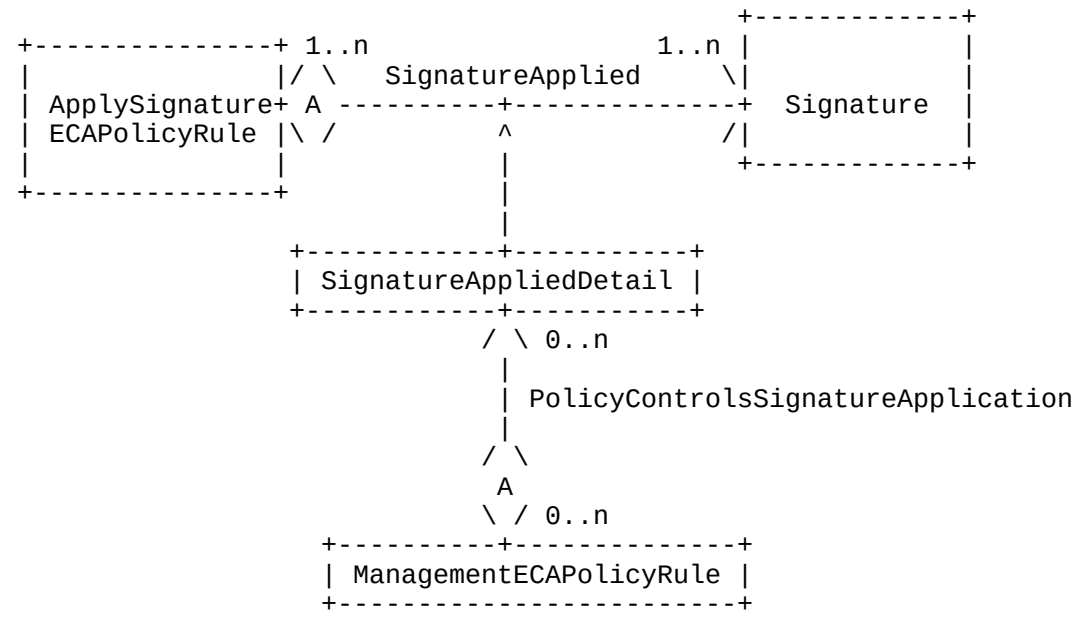


Figure 9. Modeling Sginature Application Mechanisms

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.

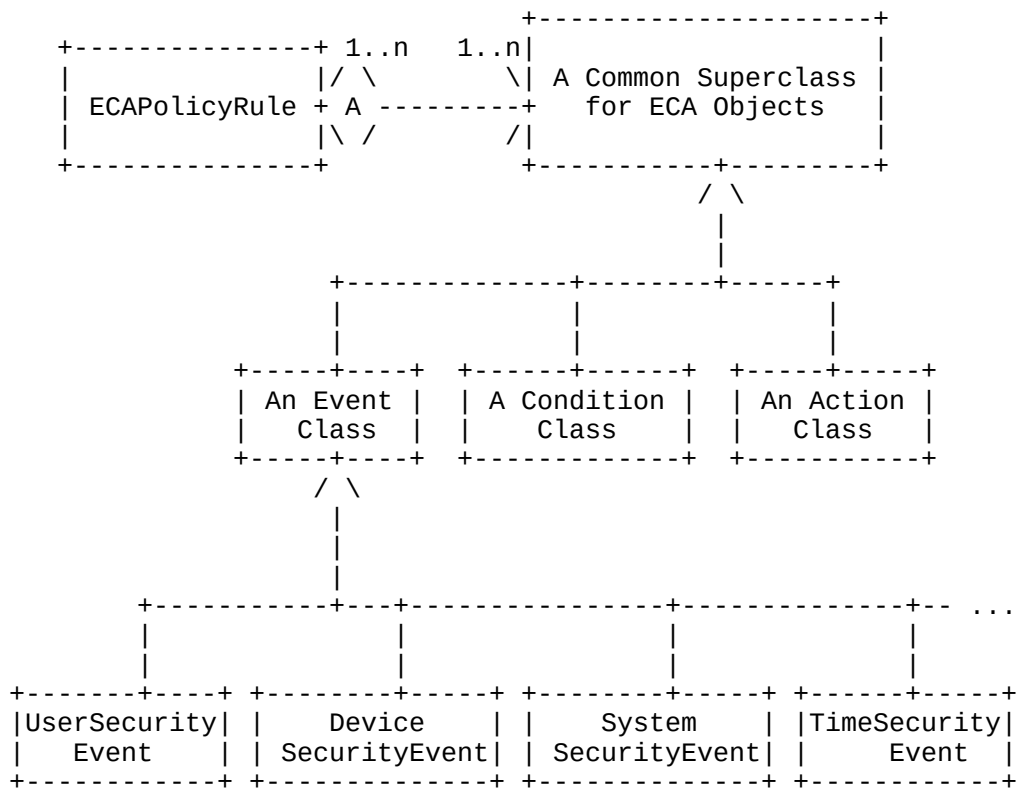


Figure 10. Network Security Info Sub-Model Event Class Extensions

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.

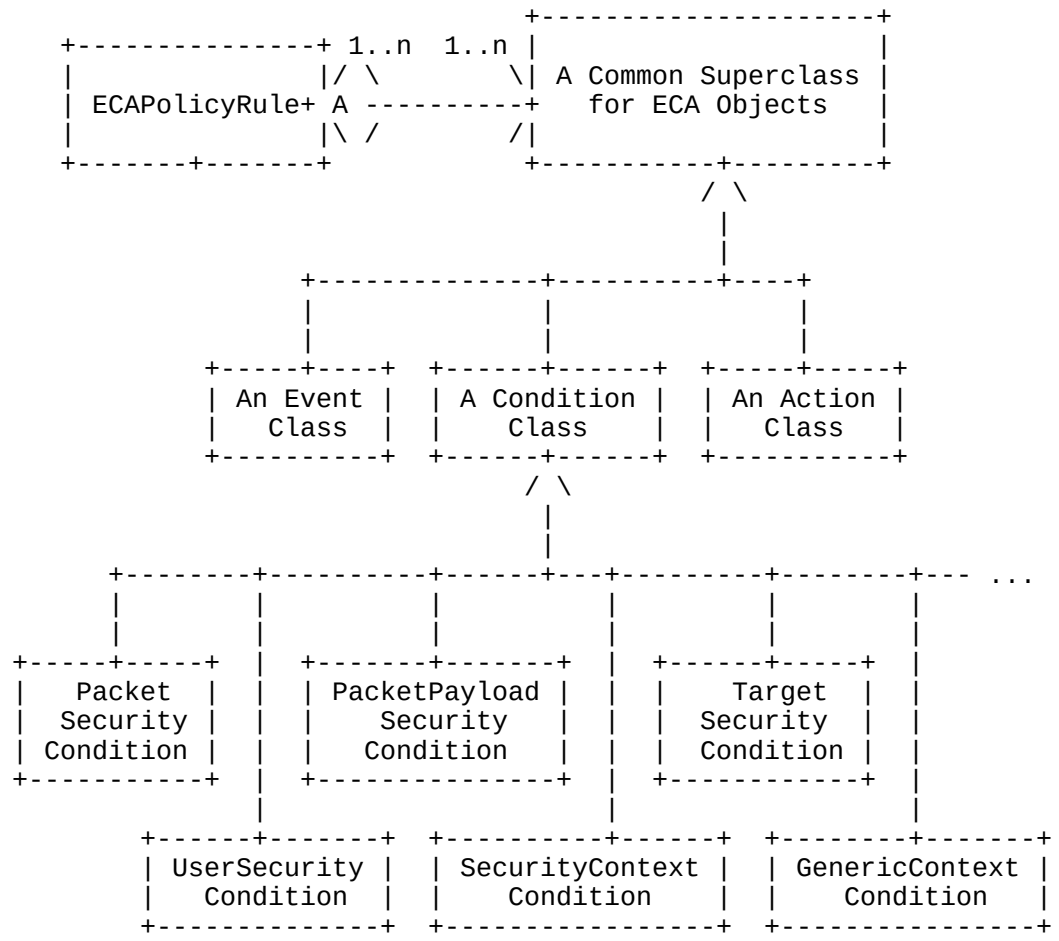


Figure 11. Network Security Info Sub-Model Condition Class Extensions

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.

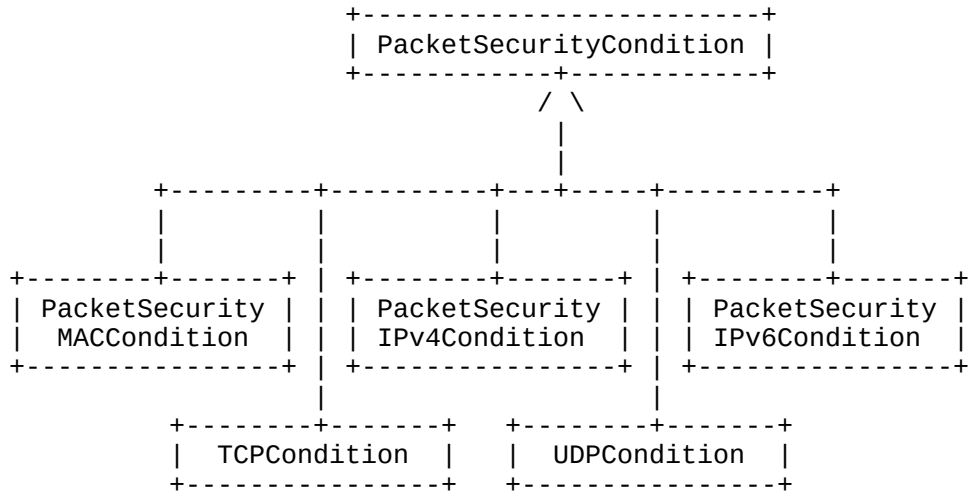


Figure 12. Network Security Info Sub-Model PacketSecurityCondition Class Extensions

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.

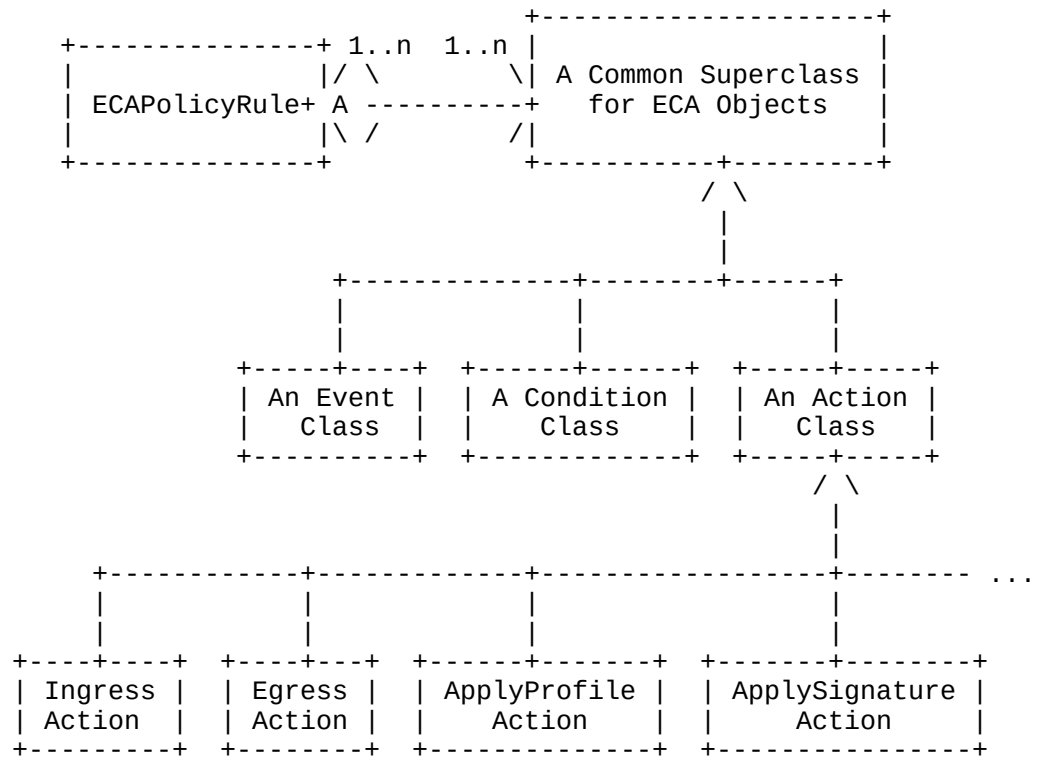


Figure 13. Network Security Info Sub-Model Action Extensions

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:

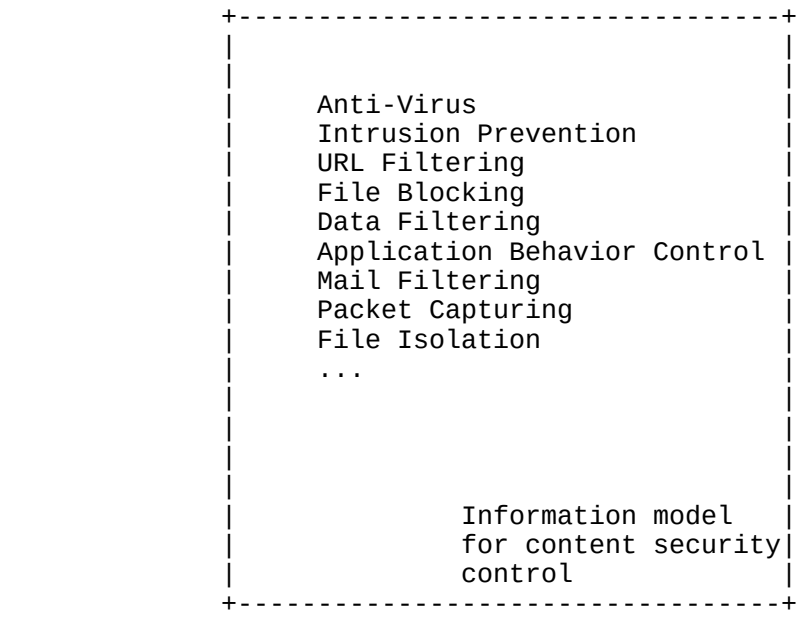


Figure 14. The basic structure of information model for content security control

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:

Please view in a fixed-width font such as Courier.

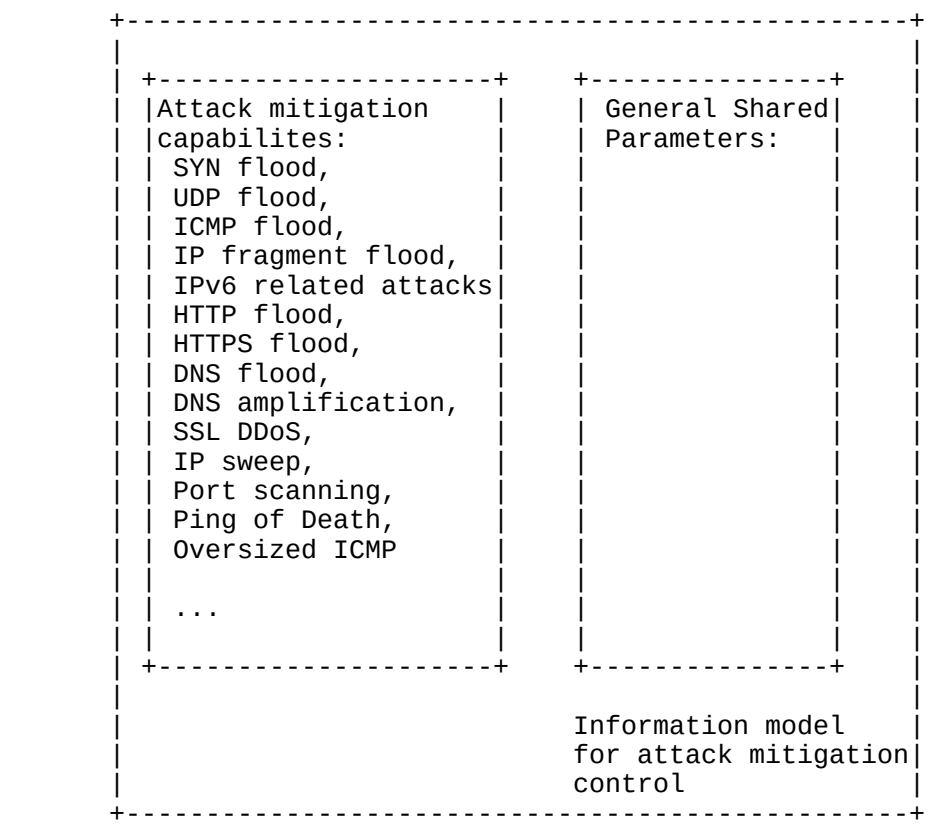


Figure 15. The basic structure of information model for attack mitigation control

The detailed description about the standard interface and the general shared parameters for all the security capabilities of this category are TBD.

5. Security Considerations

TBD

6. IANA Considerations

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-supa-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