I2NSF Working Group
Internet-Draft
Intended status: Informational
Expires: February 3, 2017

R. Kumar
A. Lohiya
Juniper Networks
D. Qi
Bloomberg
X. Long

August 2, 2016

Client Interface for Security Controller : A Framework for Security Policy Requirements
draft-kumar-i2nsf-client-facing-interface-req-00

Abstract

This document provides a framework and information model for the definition of northbound interfaces for a security controller. The interfaces are based on user-intent instead of vendor-specific or device-centric approaches that would require deep knowledge of vendor products and their security features. The document identifies the common interfaces needed to enforce the user-intent based policies onto network security functions (NSFs) irrespective of how those functions are realized. The function may be physical or virtual in nature and may be implemented in networking or dedicated appliances.

Status of This Memo

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

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

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

This Internet-Draft will expire on February 3, 2017.

Table of Contents

1. Introduction

Programming security policies in a network is a fairly complex task and requires very deep knowledge of the vendors' devices in order to implement a security policy. This has been the biggest challenge for both Service Providers and Enterprise, henceforth known as end-customers, to keep up-to-date with the security of their networks and assets. The challenge is amplified due to virtualization because security appliances come in both physical and virtual forms and are

supplied by a variety of vendors who have their own proprietary interfaces to manage and implement the security policies on their devices.

Even if an end-customer deploys a single vendor solution across its entire network, it is difficult to manage security policies due to the complexity of network security features available in the devices. The end-customer may use a vendor-provided management system that gives some abstraction in the form of GUI and helps in provisioning and managing security policies. The single vendor approach is highly restrictive in today's network as explained below:

- The end-customer cannot rely on a single vendor because one vendor may not be able keep up to date with its security needs.

- The large end-customer may have a presence across different sites and regions and that may mean it is not possible to have a single vendor solution due to technical or business reasons.

- If and when the end-customer migrates from one vendor to another, it is not possible to migrate security policies from one management system to another without complex manual work.

- Due to virtualization within data centers, end-customers are using physical and virtual forms of security functions with a wide variety of vendors, including open source, to control their costs.

- The end-customer might choose various devices in the network (such as routers, switches, firewall devices, and overlay-networks) as enforcement points for security policies for any reason (such as network design simplicity, cost, most-effective place, scale and performance).

In order to provide the end-customer with a solution where they can deploy security policies across different vendors and devices whether physical or virtual, the Interface to Network Security Functions (I2NSF) working group in the IETF is defining a set of northbound interfaces. Using these interfaces, a user can write any application e.g. GUI portal, template engine etc. but this is completely out of scope for this working group.

This document discusses the requirements for these northbound interfaces and describes a framework that can be easily used by end-customer security administrators without knowledge of specific security devices or features. We refer to this as "user-intent" based interfaces. To further clarify, the "user-intent" here does not mean some natural lanuguage input or an abstract intent such as "I want my traffic secure" or "I don't want DDoS attcks in my

network"; rather the user-intent here means that policies are described using client-oriented expressions such as application groups, device groups, user groups etc. instead of using standard n-tuples from the packet header.

2. Conventions Used in this Document

BSS: Business Support System.

CMDB: Configuration Management Database.

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

CRUD: Create, Retrieve, Update, Delete.

FW: Firewall.

IDS: Intrusion Detection System.

IPS: Intrusion Protection System.

LDAP: Lightweight Directory Access Protocol.

NSF: Network Security Function, defined by [I-D.ietf-i2nsf-problem-and-use-cases].

OSS: Operation Support System.

RBAC: Role Based Access Control.

SIEM: Security Information and Event Management.

URL: Universal Resource Locator.

vNSF: Refers to NSF being instantiated on Virtual Machines.

3. Security Provisioning Framework

The IETF I2NSF working group has defined a framework for Interfaces to Network Security Functions that defines following terminology:

Client: A client could be a GUI system used by a security administrator, an OSS/BSS system used by an end-customer, or a security controller system or application in the end-customer's management system.

Client-Facing Interface: A client-facing interface is an interface used to configure and manage security a framework across the entire network independent of device-specific interface so that same interface can be used for any device from any vendor.

The "Client Facing Interface" ensures that an end-customer can deploy any device from any vendor and still be able to use same consistent interface. In essence, these interfaces give a framework to manage end-customer's security policies. Henceforth in this document, we "security policy management interface" interchangeably when we refer to these northbound interfaces.

3.1. Client Interface Guiding Principles

Guiding principles in defining the client interfaces are as follows:

- Agnostic of network topology and NSF location in the network.
- Declarative/Descriptive model instead of Imperative/Prescriptive model - How a user would like to see security policy instead of how it would be actually implemented.
- Agnostic of vendor, implementation and form-factor (physical, virtual).
- Agnostic to how NSF is implemented and its hosting environment.
- Agnostic to how NSF becomes operational - Network connectivity and other hosting requirements
- Agnostic to NSF control plane implementation (if there is one) E.g., cluster of NSF active as one unified service for scale and/or resilience.
- Agnostic to NSF data plane implementation i.e. Encapsulation, Service function chains.

3.2. Deployment Models for Implementing Security Policies

This document describes a framework for security policy management interfaces. This document does not describe a framework for southbound interface: those may be defined in another draft.

Traditionally, medium and larger end-customers deploy management systems to manage their security policies. This approach may not be suitable for modern datacenters that are virtualized and manage their resources using controllers.

There are two different deployment models:

a. Management without an explicit management system for control of devices and NSFs. In this deployment, the security controller acts as a NSF policy management system that takes information passed over the northbound policy interface and translates into data on the I2NSF southbound interface. The I2NSF interfaces are implemented by security device/function vendors. This would usually be done by having an I2NSF agent embedded in the security device or NSF. This deployment model is shown in Figure 1.

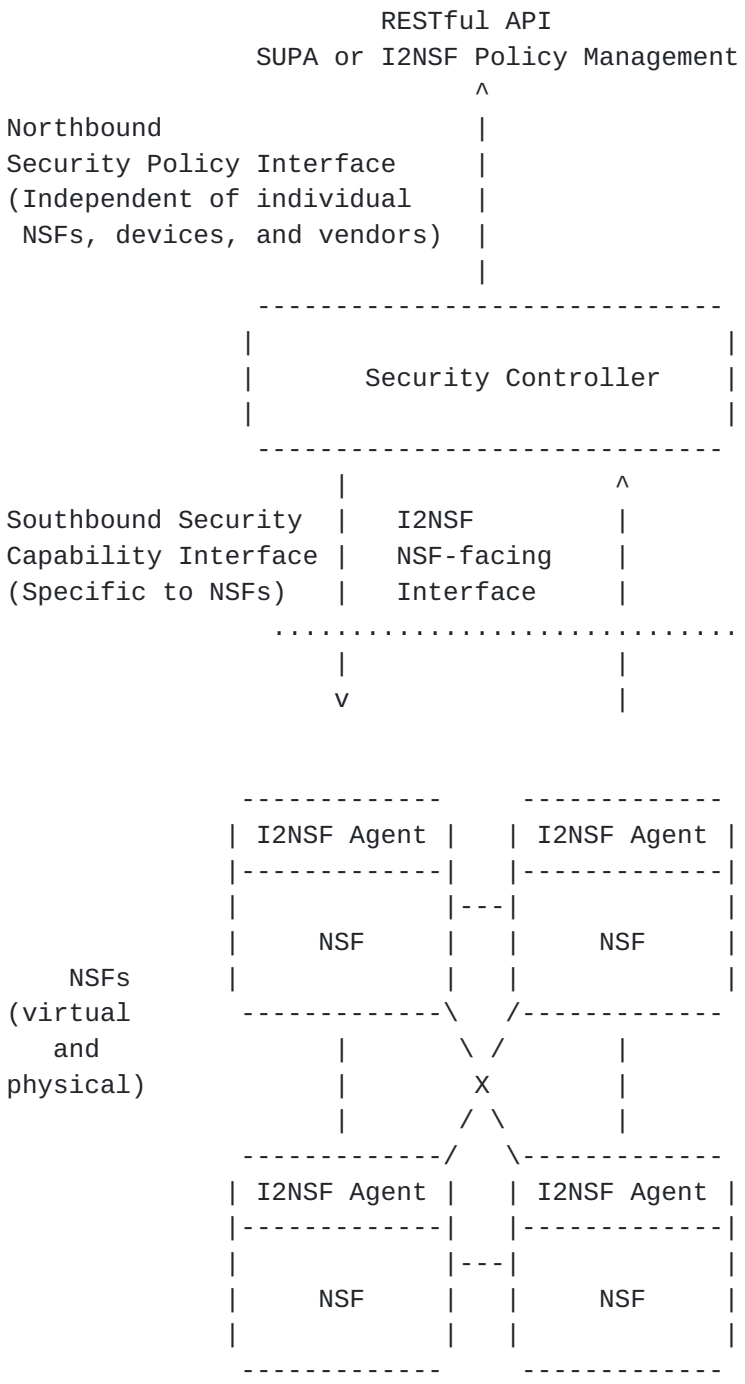

Figure 1: Deployment without Management System

b. Management with an explicit management system for control of devices and NSFs. This model is similar to the model above except that security controller interacts with a dedicated management system which could either proxy I2NSF southbound interfaces or could provide a layer where security devices or

NSFs do not support an I2NSF agent to process I2NSF southbound interfaces. This deployment model is shown in Figure 2.

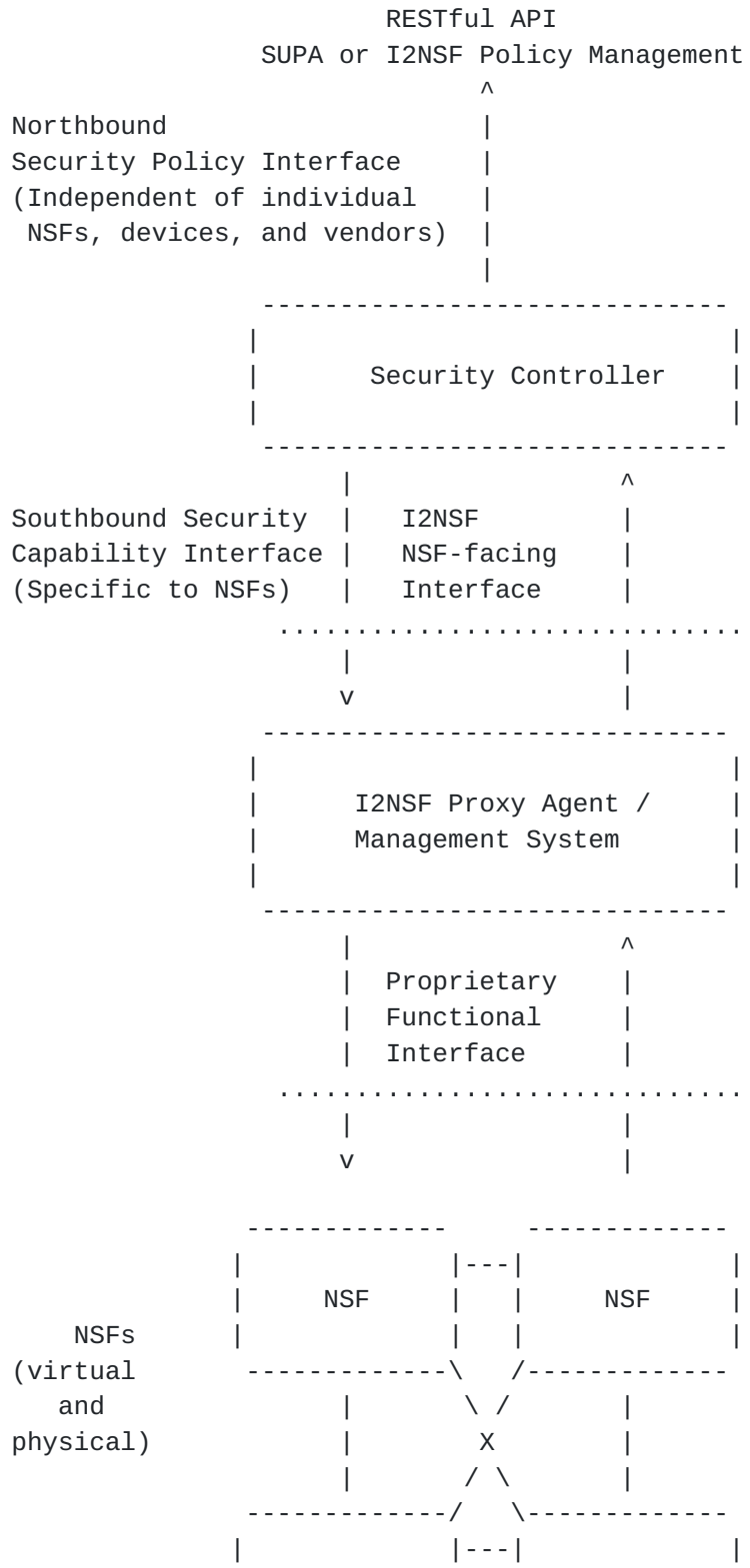

Figure 2: Deployment with Management System or I2NSF Proxy Agent

Although the deployment models discussed here don't necessarily affect the northbound security policy interface, they do give an overall context for defining a security policy interface based on abstraction.

3.3. Client Perspective on Security Policy Configuration and Management

In order to provide I2NSF northbound interface for security policies to client that are not specific to any vendor, device or feature implementation, it is important that security policies shall be configured and managed from a client's perspective. We refer to this as the user-intent based model since it is primarily driven by how security administrators view security policies from the deployment perspective.

The client perspective ensures that policy management is not only easy to understand for them (the actual users), but is also independent of vendor, device, and specific implementation which is the foremost goal for a northbound interface.

4. Functional Requirements for the Client Interface

As mentioned earlier, it is important that the northbound interface be primarily driven by user-intent which is what a client understands well. In order to define this interface, we must understand the requirements and framework used by the security administrator.

A security policy that is based on user-intent is completely agnostic of how this policy is enforced in the end-customer's network. The security controller may choose to implement such a policy on any device (router, switch, firewall) in a physical or virtual form factor. The security controller's implementation is outside the scope of this document and the I2NSF working group.

At a high level, the objects that are required in order to express and build the security policies fall into the following categories:

- Multi-tenancy and RBAC for policy management
- Policy lifecycle management

- Policy endpoint groups
- Policy rules
- Policy actions
- Third party integration
- Telemetry data

The above categories are by no means a complete list and may not be sufficient for all use-cases and all end-customers, but should be a good start for a wide variety of use-cases in both Service Provider networks and Enterprise networks.

The following sections provide further details on the above mentioned security policies categories.

4.1. Multi-Tenancy and RBAC for Policy Management

An end-customer that uses security policies may have internal tenants and would like to have a framework wherein each tenant manages its own security policies to provide isolation across different tenants.

An end-customer may be a cloud service provider with multi-tenant deployments where each tenant is a different organization and must allow complete isolation across different tenants.

The RBAC objects and method needed to build such a framework is defined below.

Policy-Tenant: An entity that owns and manages the security policies.

Policy-User: A user within a Policy-Tenant authorized to manage security policies for that tenant.

Policy-Authorization-Role: A role assigned to a Policy-User that determines whether the user has read-write access, read-only access, or no access for certain resources.

Authentication and Authorization Scheme: There must be a scheme for a Policy-User to be authenticated and authorized to use the security controller. There are several authentication schemes avialable such as OAuth, XAuth and X.509 certificate based. The authentication scheme between client and controller may also be mutual instead of one-way. Any specific scheme may be determined

based on organizational and deployment needs and outside the scope of I2NSF.

4.2. Policy Lifecycle Management

In order to provide more sophisticated security framework, there should be a mechanism to express that a policy becomes dynamically active/enforced or inactive based on either security administrator intervention or an event.

One example of dynamic policy management is when the security administrator pre-configures all the security policies, but the policies get activated/enforced or deactivated based on dynamic threats faced by the end-customer. Basically, a threat event may activate certain inactive policies, and once a new event indicates that the threat has gone away, the policies become inactive again.

The northbound interface should support the following mechanisms for policy enforcement:

Admin-Enforced: The policy, once configured, remains active/enforced until removed by the security administrator.

Time-Enforced: The policy configuration specifies the time profile that determines when policy is activated/enforced.

Event-Enforced: The policy configuration specifies the event profile that determines when policy is activated/enforced.

4.3. Policy Endpoint Groups

Typically, when the security administrator configures a security policy, the intention is to apply this policy to certain subsets of the network. The subsets may be identified based on criteria such as users, devices, and applications. We refer to such a subset of the network as a "Policy Endpoint Group".

One of the biggest challenges for a security administrator is how to make sure that security policies remain effective while constant changes are happening to the "Policy Endpoint Group" for various reasons (e.g., organizational changes). If the policy is created based on static information such as user names, application, or network subnets, then every time that this static information changes policies would need to be updated. For example, if a policy is created that allows access to an application only from the group of Human Resource users (the HR-users group), then each time the HR-users group changes, the policy needs to be updated.

Changes to policy could be highly taxing to the end-customer for various operational reasons. The policy management framework must allow "Policy Endpoint Group" to be dynamic in nature so that changes to the group (HR-users in our example) automatically result in updates to its content.

We call these dynamic Policy Endpoint Groups "Meta-data Driven Groups". The meta-data is a tag associated with endpoint information such as users, applications, and devices. The mapping from meta-data to dynamic content could come either from standards-based or proprietary tools. The security controller could use any available mechanisms to derive this mapping and to make automatic updates to the policy content if the mapping information changes.

The northbound policy interface must support endpoint groups for user-intent based policy management. The following meta-data driven groups are typically used for configuring security polices:

User-Group: This group identifies a set of users based on a tag or on static information. The tag to user information is dynamically derived from systems such as Active Directory or LDAP. For example, an end-customer may have different user-groups, such as HR-users, Finance-users, Engineering-users, to classify a set of users in each department.

Device-Group: This group identifies a set of devices based on a tag or on static information. The tag to device information is dynamically derived from systems such as CMDB. For example, an end-customer may want to classify all machines running one operating system into one group and machines running another operating system into another group.

Application-Group: This group identifies a set of applications based on a tag or on static information. The tag to application information is dynamically derived from systems such as CMDB. For example, an end-customer may want to classify all applications running in the Legal department into one group and all applications running under a specific operating system into another group.

Location-Group: This group identifies a set of locations based on a tag or on static information. The tag to location information is dynamically derived from systems such as CMDB. For example, an end-customer may want to classify all sites/locations in a geographic region as one group.

4.4. Policy Rules

The security policy rules can be as simple as specifying a match for the user or application specified through "Policy Endpoint Group" and take one of the "Policy Actions" or more complicated rules that specify how two different "Policy Endpoint Groups" interact with each other. The northbound interface must support mechanisms to allow the following rule matches.

Policy Endpoint Groups: The rule must allow a way to match either a single or a member of a list of "Policy Endpoint Groups".

There must also be a way to express whether a group is a source or a destination so that the security administrator can apply the rule in only one direction of a communication.

There must also be a way to express a match between two "Policy Endpoint Groups" so that a policy can be effective for communication between two groups.

Direction: The rule must allow a way to express whether the security administrator wants to match the "Policy Endpoint Group" as the source or destination. The default should be to match both directions if the direction rule is not specified in the policy.

Threats: The rule should allow the security administrator to express a match for threats that come either in the form of feeds (such as botnet feeds, GeoIP feeds, URL feeds, or feeds from a SIEM) or speciality security appliances.

The threat could be from malware and this requires a way to match for virus signatures or file hashes.

4.5. Policy Actions

The security administrator must be able to configure a variety of actions within a security policy. Typically, security policy specifies a simple action of "deny" or "permit" if a particular rule is matched. Although this may be enough for most of the simple policies, the I2NSF northbound interface must also provide a more comprehensive set of actions so that the interface can be used effectively across various security functions.

Permit: This action means continue processing the next rule or allow the packet to pass if this is the last rule.

Deny: This action means stop further rule processing and drop the packet.

Drop connection: This action means stop further rule processing, drop the packet, and drop connection (for example, by sending a TCP reset).

Log: This action means create a log entry whenever a rule is matched.

Authenticate connection: This action means that whenever a new connection is established it should be authenticated.

Quarantine/Redirect: This action may be relevant for event driven policy where certain events would activate a configured policy that quarantines or redirects certain packet flows.

4.6. Third-Party Integration

The security policies in the end-customer's network may require the use of specialty devices such as honeypots, behavioral analytics, or SIEM in the network, and may also involve threat feeds, virus signatures, and malicious file hashes as part of comprehensive security policies.

The northbound interface must allow the security administrator to configure these threat sources and any other information to provide integration and fold this into policy management.

4.7. Telemetry Data

One of the most important aspect of security is to have visibility into the networks. As threats become more sophisticated, the security administrator must be able to gather different types of telemetry data from various devices in the network. The collected data could simply be logged or sent to security analysis engines for behavioral analysis, policy voilations, and for threat detection.

The northbound interface must allow the security administrator to collect various kinds of data from NSFs. The data source could be syslog, flow records, policy violation records, and other available data.

5. Operational Requirements for the Client Interface

5.1. API Versioning

The northbound interface must support a version number for each RESTful API. This is very important because the client application and the controller application will most likely come from different

vendors. Even if the vendor is same, it is hard to imagine that two different applications would be released in lock step.

Without API versioning, it hard to debug and figure out issues if application breaks. Although API versioning does not guarantee that applications will always work, it helps in debugging if the problem is caused by an API mismatch.

5.2. API Extensiblity

Abstraction and standardization of the northbound interface is of tremendous value to end-customers as it gives them the flexibility of deploying any vendors' NSF. However this might also look like as an obstacle to innovation.

If an NSF vendor comes up with new feature or functionality that can't be expressed through the currently defined northbound interface, there must be a way to extend existing APIs or to create a new API that is relevant for that NSF vendor only.

5.3. APIs and Data Model Transport

The APIs for client interface must be derived from the YANG based data model. The YANG data model for client interface must capture all the requirements as defined in this document to express a security policy. The interface between a client and controller must be reliable to ensure robust policy enforcement. Once such transport mechanism is RESTCONF that uses HTTP operations to provide necessary CRUD operations for YANG data objects, but any other mechanism can be used.

5.4. Notification

The northbound interface must allow the security administrator to collect various alarams and events from the NSF in the network. The events and alarms may be either related to security policy enforcement or NSF operation. The events and alarms could also be used as a input to the security policy for autonomous handling.

5.5. Affinity

The northbound interface must allow the security administrator to pass any additional metadata that a user may want to provide for a security policy e.g. certain security policy needs to be applied only on linux machine or windows machine or that a security policy must be applied on the device with Trusted Platform Module chip.

5.6. Test Interface

The northbound interface must allow the security administrator the ability to test the security policies before the policies are actually applied e.g. a user may want to verify if a policy creates potential conflicts with the existing policies or whether a certain policy can be implemented. The test interface provides such capabilities without actually applying the policies.

6. IANA Considerations

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

7. Acknowledgements

The editors would like to thank Adrian Farrel for helpful discussions and advice.

8. Normative References

[I-D.ietf-i2nsf-problem-and-use-cases]
Hares, S., Dunbar, L., Lopez, D., Zarny, M., and C. Jacquenet, "I2NSF Problem Statement and Use cases", draft-ietf-i2nsf-problem-and-use-cases-01 (work in progress), July 2016.

Authors' Addresses

Rakesh Kumar
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
US

Email: rkkumar@juniper.net

Anil Lohiya
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
US

Email: alohiya@juniper.net

Dave Qi
Bloomberg
731 Lexington Avenue
New York, NY 10022
US

Email: DQI@bloomberg.net

Xiaobo Long
4 Cottonwood Lane
Warren, NJ 07059
US

Email: long.xiaobo@gmail.com