INTERNET-DRAFT
Intended Status: Informational
Expires: September 23, 2015

E. Lopez
Fortinet

March 22, 2015

Packet-Based Paradigm For Interfaces To NSFs
draft-lopez-i2nsf-packet-00

Abstract

In the design of interfaces to allow for the provisioning of network-based security functions (NSFs), a critical consideration is to prevent the creation of implied constraints.

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

Table of Contents

1 Introduction

The emergence of networking paradigms based on the use of devices with programmable forwarding planes, has resulted in the need to create application program interfaces (APIs) in support integration of NSFs within software-defined networking (SDN) and network-function virtualization (NFV) environments.

APIs to NSFs can be generally grouped into three types:

1) Configuration - deals with the management and configuration of the forwarding functions of the NSF device itself. Configuration API functions may already be part of ongoing efforts associated with other management protocols such as NETCONF.

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

3) Provisioning - used to control the security functions of NSFs. Due to the lack of standards in the definition and operation of these functions, much of the efforts towards interface development will be in this area.

This draft proposes that a provisioning interface to NSFs can be developed on a packet-based paradigm. While there are many classifications of existing and emerging NSFs, a common trait shared by them is in the processing of packets based on the content (header/payload) and context (session state, authentication state, etc) of received packets.

An important concept is the fact that attackers do not have standards as to how to attack networks, so it is equally important not to constrain NSF developers to offering a limited set of security functions. Therefore, in constructing standards for provisioning interfaces to NSFs, it is equally important to allow support for vendor-specific functions, to allow the introduction of NSFs that evolve to meet new threats. Proposed standards for 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 standardized functions, without

allowing for vendor-specific functions

- Be seen as endorsing a best-common-practice for the implementation of NSFs

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

1.1 Terminology

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

2 Packet-Based Paradigm

Conceptually, the packet-based paradigm is that regardless of functional capabilities, NSFs share that same principal capability as all NBFs in processing packets. NSFs process packets based on information from:

- the packet header- the packet payload- functional state information on NSFs regarding the packets, or the sessions to which packets belong

2.1 Packet Headers

The examination of packet headers is useful in the classification of packets up to the transport layer (IP protocol/port). This information is also used in packet forwarding decisions.

For example, [OPENFLOW-1.5] makes use of some packet header fields as match structures in defining match structures for OpenFlow-compatible switches. In a similar way, a provisioning interface for NSFs can make use of packet header field as a 'subject' value, defining the packet header characteristics matching a particular policy statement.

2.2 Packet Payloads

The examination of packet payloads is useful in the classification of packets at the application layer. It is assumed that technical means

exist in which information can be gleaned by NSFs from packet payloads, which can be used to detect application types beyond IP protocol/port, or to look into additional header fields within encapsulated traffic.

Payload-based characteristics can also be incorporated as a 'subject' value of a provisioning interface. However, a common syntax for expressing how payload values are matched needs to be clearly defined.

2.3 Functional State Information

NSFs not only inspect values within the packet, but also a variety of contexts associated with the packet. Examples include:

- An explicit context, such as a user or device authentication state, which may be correlated to an IP address within the packet.- Time-based information, which can be explicitly used to determine the validity of a packet relative to a policy schedule, or implicitly relative to a timer-based psuedo state of a session using a stateless protocol like UDP- An implicit context, such as the session state of a stateful IP protocol such as TCP or SCTP

The contextual states employed by an NSF in evaluating packets can be considered to be 'object' values of a provisioning interface. Since many of these are dependant of the capabilities of the NSF, a means of performing a capabilities exchange of 'object' values which can utilized in provisioning policy onto NSFs.

4 Provisioning Interface Structural Overview

It is not the intention of this draft to provide technical detail on a provisioning interface to NSFs. Instead, it is intended to suggest that a packet-based paradigm can be used to describe policy statements applied to NSFs. Such policy statements would be based upon four root values:

Subject.Object.Function.Action

Where:

- Subject = Match values based on information carried within the packet header or payload itself.- Object = Match values based on contextual information associated with received packets.- Function = Values with invoke specific security functions provided by the NSF.- Action = Values that determine how packets are handled post security function processing.

Each of these root values can be defined with additional sub-values to provide required details. For example, a 'function' may have additional object values to provide granular information in how packets are processed. If a NSF function was to perform web filtering, a functional statement may not only state that web-filtering is to be performed, but also which defined filter object is to be employed (Function=<function>:<instance>). It is intended that this form of root:branch structure can be easily integrated into normative API structures, such as JSON or RESTful APIs

This implies that there are four types of communications used within this provisioning interface:

- Exchange - a capabilities exchange between the NSF and the provisioning application- Configure - statement that provide a provisioning application the ability to create and configure objects use in provisioning policy statements- Provision - Policy statements used to provision security functions within NSFs- Query - statements which may either be requests by provisioning applications to query policy statement data from NSFs, or gratuitous information (such as counters) from NSFs to provisioning applications

5 Security Considerations

As this draft is focused on the creation of interfaces to NSFs, the security considerations are based on:

- Preventing the imposition of constraints which would limit the functionality of NSFs, or the ability to deploy them onto networks.

- Ensuring the creation of such interfaces does not create additional security vulnerabilities, including risks associated with their passive surveillance.

6 IANA Considerations

This draft does not impose requirements onto IANA.

7 References

7.1 Normative References

[KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

7.2 Informative References

[USECASES] Pastor, A. & Lopez,D., "Access Use Cases for an Open OAM Interface to Virtualized Security", draft-pastor-i2nsf-access-usecases-00, October 2014, <http://datatracker.ietf.org/doc/draft-pastor-i2nsf-access-usecases>

[PCI-DSS] PCI Security Standards Council, "Payment Card Industry (PCI) Data Security Standard - Requirements and Security Assessment Procedures - Version 3", November 2013, <https://www.pcisecuritystandards.org/documents/PCI_DSS_v3.pdf>

[OPENFLOW-1.5] Open Networking Foundation, "OpenFlow Switch Specification, Version 1.5.0", December 2014, <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf>

8 Acknowledgements

The author wishes to thank and acknowledge the support of Linda Dunbar, Diego Lopez Garcia, Dan Romascanu, and Kathleen Moriarty in discussions which led to the creation of this draft. This acknowledgement does not imply agreement with or endorsement of this draft by these individuals.

Authors' Addresses

Edward Lopez
Fortinet
899 Kifer Road
Sunnyvale, CA 94086

Phone: +1 703 220 0988
EMail: elopez@fortinet.com