Network Working Group S. Hyun
Internet-Draft J. Jeong
Intended status: Standards Track S. Woo
Expires: May 17, 2017 Y. Yeo
Sungkyunkwan University
J. Park
ETRI
November 13, 2016

NSF-triggered Traffic Steering in I2NSF Framework
draft-hyun-i2nsf-nsf-triggered-steering-01

Abstract

This document describes an architecture of the framework for Interface to Network Security Functions (I2NSF) which enables traffic steering between Network Security Functions (NSFs) for security policy enforcement. Such traffic steering enables composite inspection of network traffic by steering the traffic through multiple types of NSFs according to the information model for the NSF-facing interface in the I2NSF framework. This document explains the additional components integrated into the I2NSF framework and their functionalities to achieve NSF-triggered traffic steering. It also describes representative use cases to address major benefits from the proposed architecture.

Status of This Memo

This Internet-Draft will expire on May 17, 2017.

Copyright Notice

Table of Contents

1. Introduction

To effectively cope with emerging sophisticated network attacks, it is necessary that various Network Security Functions (NSFs) cooperatively analyze network traffic [sfc-ns-use-cases][RFC7498] [i2nsf-problem][capability-im]. In addition, depending on the characteristics of network traffic and their suspiciousness level, the different types of network traffic need to be analyzed through the different sets of NSFs. An information model in [capability-im] is proposed for NSF-facing interface in the framework for Interface to Network Security Functions (I2NSF) that enables an NSF to trigger further inspection by calling another NSF based on its own analysis results [i2nsf-framework]. However, the current design of the I2NSF framework does not consider network traffic steering fully in order to enable such consecutive inspections through multiple NSFs.

In this document, we propose an architecture that integrates additional components for traffic steering over NSFs into the I2NSF framework. We extend the security controller's functionalities such that it can interpret a high-level policy of NSF-triggered traffic steering into a low-level policy and manage them. It also keeps track of the available NSF instances and their information (e.g., network information and workload), and makes a decision on which NSF instances to use for a given NSF. Based on the forwarding information provided by the security controller, a Network Security Function Forwarder (NSFF) performs network traffic steering through required NSFs. The NSFF is also responsible for interpreting inspection result from an NSF to enforce more advanced inspection. We define an additional packet header format to specify security inspection results and advanced inspection requests.

2. Objective

o Policy configuration for consecutive inspections: NSF-triggered traffic steering architecture allows policy configuration and management of NSF triggering. Based on the triggering policy, relevant network traffic can be analyzed through various NSFs in a composite, cooperative manner.

- o Network traffic steering for consecutive inspection: NSF-triggered traffic steering architecture allows network traffic to be steered through multiple required NSFs based on the triggering policy. Moreover, the I2NSF information model for NSF facing interface [capability-im] requires an NSF to call another NSF for further inspection based on its own inspection result. To meet this requirement, NSF-triggered traffic steering architecture also enables traffic forwarding from one NSF to another NSF.

- o Load balancing over NSF instances: NSF-triggered traffic steering architecture provides load balancing of incoming traffic over available NSF instances by leveraging the flexible traffic steering mechanism. For this objective, it also performs dynamic instantiation of an NSF when there are an excessive amount of requests for that NSF.

3. Terminology

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

- o Network Security Function (NSF): A function that is responsible for specific treatment of received packets. A Network Security Function can act at various layers of a protocol stack (e.g., at the network layer or other OSI layers) [RFC7665]. Sample Network Security Service Functions are as follows: Firewall, Intrusion Prevention/Detection System (IPS/IDS), Deep Packet Inspection (DPI), Application Visibility and Control (AVC), network virus and malware scanning, sandbox, Data Loss Prevention (DLP), Distributed Denial of Service (DDoS) mitigation and TLS proxy.

- o Advanced Inspection/Action: As like the I2NSF information model for NSF facing interface [capability-im], Advanced Inspection/ Action means that an NSF calls another NSF for further inspection based on its own inspection result.

- o Network Security Function Profile (NSF Profile): NSF Profile represents NSF's inspection capabilities. Each NSF has its own

NSF Profile to specify the type of security service it provides and its resource capacity etc.

- Network Security Function Operation Manager (NSF Operation Manager): NSF Operation Manager consistently manages information and state of NSF instances and provides NSF network access information to support advanced inspection request. For example, the information includes the supported transport protocols, IP addresses, and locations for the NSF instances. Also, NSF Operation Manager takes charge of dynamic management of a pool of NSF instances by consulting with Developer's Management System and load balancing over NSF instances.

- Packet Forwarding Header/Encapsulation: Packet Forwarding Header is used to forward a packet from one NSF to another for further inspection. The former NSF constructs a Packet Forwarding Header with the NSF profile of the latter NSF and transmits it to a NSFF. The required fields are the action code, the number of the metadata, and the metadata. In this context, the metadata is a

part of NSF profile.

- Network Security Function Forwarder (NSFF): An NSF forwarder is responsible for forwarding traffic to one or more connected NSFs according to the information carried in the packet forwarding encapsulation when the traffic comes back from an NSF. Additionally, an NSFF is responsible for transporting traffic to another NSFF (in the same or the different type of overlay), and terminating overlay inspection [RFC7665].

4. Architecture

This section describes an NSF-triggered traffic steering architecture and the basic operations of traffic steering. It also includes details about each component of the architecture.

Figure 1 describes the components of NSF-triggered traffic steering architecture. Our architecture enables support a composite inspection of packets in transit. According to the inspection result of each NSF, which is stored in the Packet Forwarding Header, the traffic packets could be steered to another NSF for futher detailed analysis. It is also possible to reflect a high-level advanced inspection policy and a configuration from I2NSF User which is a component of the original I2NSF framwork. Moreover, the proposed architecture provides load balancing, auto supplementary NSF instance generation, and the elimination of unused NSF instances. In order to achieve these design purposes, we integrate several components to the original I2NSF framwork. In the following sections, we explain the details of each component.


```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| I2NSF User                                                      |
|              +-+-+-+-+-+-+-+-+                                    |
|              |User/          |                                    |
|              |App Controller |                                    |
|              +-+-+-+^+-+-+-+-+                                    |
|                     |                                             |
|                     |                                             |
+-+-+-+-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                      | Consumer-Facing Interface
                      |
+-+-+-+-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Security Management System                                        |
|                     |                                             |
|       +-+-+-+-+-+-+-v-+-+-+-+                                     |
```

```
   |       |Security Controller  |                                     |
   |       |    +-+-+-+-+-+-+    | Registration                        |
   |       |    | NSF        |   |   Interface  +-+-+-+-++-+-+-+-+    |
   |       |    | Opeation   |   |<------------>| Developer's     |   |
   |       |    | Manager    |   |              | Mgnt System     |<--+ |
   |       |    +-+-+-+-+-+-+    |              +-+-+-+-++-+-+-+-+   | |
   |       +-+-+-+-+-+-^-+-+-+-+-+                                  | |
   +-+-+-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                       | NSF-Facing Interface                        |
                       |                                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |Security Network   |                                             | |
   |                   +-------------------------------------+       | |
   |                   |                                     |       | |
   |             +-+-+-v-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+       |       | |
   |             |  +---------+      +---------+  |          |       | |
   |             |  | NSFF-1  |  ... | NSFF-n  |  |          |       | |
   |             |  +---------+      +---------+  |          |       | |
   |             +-+-+-+-+-+-+-+-^-+-+-+-+-+-+-+-+-+         |       | |
   |                             |                           |       | |
   |       +-+-+-+-+-+-+-+-+-+-+-v-+-+-+-+-+-+-+-+-+-+-+-+   |       | |
   |       |  +---------+  +---------+      +---------+  |<-+       | |
   |       |  |  NSF-1  |  |  NSF-2  |  ... |  NSF-m  |  |          | |
   |       |  +---------+  +---------+      +---------+  |<---------+ |
   |       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: NSF-triggered Traffic Steering Architecture

4.1. NSF Operation Manager

NSF Operation Manager is a core component in our system. It is responsible for the following three things: (1) Maintaining the information of every available NSF instance such as IP address, supported transport protocol, NSF profile, and load status. (2) Responding the queries of available NSF instances from NSFF so as to help to conduct advanced inspection relevant to a given NSF profile.

(3) Requesting Developer's Management System for the dynamic instantiation of supplementary NSF instances to avoid service congestion or the elimination of an existing NSF instance to avoid resource waste. As Figure 1 describes, NSF Operation Manager is a sub-module of Security Controller.

Whenever a new NSF instance is registered, Developer's Management System passes the information of the registered NSF instance to NSF Operation Manager, so NSF Operation Manager maintains a list of the information of every available NSF instance. NSF Operation Manger will receive the request packet containing NSF profile for advanced inspection from NSFF. Once receiving a query of a certain NSF profile from NSFF, NSF Operation Manager searches for all the available NSF instances applicable for that NSF profile and then finds the best instance with selection criteria like location and load status. After finding the best instance, it returns the search result to NSFF.

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

4.2. Developer's Management System

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

4.3. Packet Forwarding Header

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Outer Encapsulation | Packet Forwarding Header| Origin Packet |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                     /                            \
          +----------+                             +------------+
         /                                                       \
        /                                                         \
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       | Action Code | SpecInfo Num| SpecInfo-0| ... | SpecInfo-n|
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Packet Forwarding Header Format

Packet Forwarding Header is used to convey inspection result and required inspection to an NSFF, so it has variable length of fields like Figure 2. It contains fixed Action and the SpecInfo Num fields and variable SpecInfo fields. Action field has a value out of "allow", "deny", "advanced", and "mirror". SpecInfo Num field represents how many SpecInfos are included in the Packet Forwarding Header and each SpecInfo can include a part of NSF Profile which is required for the next inspection. For instance, SepcInfo can be "syn-flood-mitigate", "udp-flood-mitigate", "content-matching-tcp" etc, which are the service profile of an NSF.

4.4. Network Security Function Forwarder (NSFF)

It is responsible for the following two functionalities: (1) Initiallyu forwarding the incoming traffic/packets to Network Security Sub-Module, as described in the I2NSF information model for NSF facing interface [capability-im]. (2) Forwarding the traffic/packets to the matched NSF with the NSF profile which is specified in a Packet Forwarding Header.

An NSFF takes a gateway functionality, so it receives incoming traffic/packets first and attaches outer encapsulation in order to forward the traffic/packets to Network Sub-Module [capability-im]. The example of Network Sub-Moudle is a firewall which performs packet header inspection. This Network Security Sub-Module attaches a Packet Forwarding Header between the outer encapsulation and the original packet and specifies NSF Profile in that header so that it can be forwarded to Content Security Sub-Module or Mitigate Sub-Module for advanced inspection.

When receiving a packet attached with a packet forwarding header of a specific NSF profile, an NSFF searches for an available NSF instance which provides the network security service corresponding to (matching with) the NSF profile and forward the packet to the NSF instance. If an NSF decides that the packet requires further inspection via another type of NSF, it constructs a packet forwarding header specified with (including) the NSF profile of the advanced NSF, attaches the header to the packet, and then sends the resulting packet to the NSFF. Once receiving the packet, the NSFF checks the NSF profile specified in the packet forwarding header. Then it searches for an NSF instance matching with the NSF profile by consulting with NSF Operation Manager, and finally forwards the packet to the NSF instance.

5. Use Cases

This section introduces two use cases for the NSF-triggered Traffic Steering Framework: (1) Enforcing Different NSFs Depending on a Packet Source's Trust Level, (2) Effective Load Balancing with Dynamic NSF Instantiation.

5.1. Enforcing Different NSFs Depending on a Packet Source's Trust Level

In the proposed architecture, all incoming packets initially arrive at the NSFF. We assume that the current security policy forces all incoming packets to be by default inspected by a firewall in this scenario. Thus the NSFF forwards the received packets to a firewall instance. Then the firewall identifies the source of the traffic and evaluates the trust level of the source. If the traffic comes from a trusted source, it is likely to be benign. In this case, the traffic is just forwarded to the destination without further detailed inspection via different types of NSFs as illustrated in Figure 3.(a). Otherwise if the traffic comes from an untrusted source, the firewall attaches a packet forwarding header including the NSF profile corresponding to DPI to the packet and returns the resulting packet to the NSFF. Once receiving the packet, the NSFF forwards the packet to the DPI instance which will perform detailed inspection for the packet payload. Figure 3.(b) illustrates this case.

```
   +-+-+-+-+-+           +-+-+-+-+-+           +-+-+-+-+-+-+-+
   | Source  |---------->| Firewall|---------->| Destination |
   +-+-+-+-+-+           +-+-+-+-+-+           +-+-+-+-+-+-+-+

             (a) Traffic Flow of Trusted Source


   +-+-+-+-+-+     +-+-+-+-+-+     +-+-+-+-+-+     +-+-+-+-+-+-+-+
   | Source  |---->| Firewall|---->|   DPI   |---->| Destination |
   +-+-+-+-+-+     +-+-+-+-+-+     +-+-+-+-+-+     +-+-+-+-+-+-+-+

             (b) Traffic Flow of Untrusted Source
```

Figure 3: Different Path Allocation according to Traffic Source

5.2. Effective Load Balancing with Dynamic NSF Instantiation

In a large-scale network domain, there typically exist a large number of NSF instances that provide various security services. It is possible that a specific NSF instance experiences an excessive amount of traffic beyond its capacity. In this case, it is required to allocate some of the traffic to another available instance of the same NSF. If there are no additional instances of the same NSF available, we need to create a new NSF instance and then direct the subsequent traffic to the new instance. In this way, we can avoid service congestion and achieve more efficient resource utilization.

This process is commonly called load balancing. In our proposed architecture, NSF Operation Manager performs periodic monitoring of the load status of available NSF instances. In addition, it is possible to dynamically generate a new NSF instance through Developer's Management System. With these functionalities along with the flexible traffic steering mechanism, we can eventually provide load balancing service.

The following describes the detailed process of load balancing when congestion occurs at the firewall instance:

1. NSF Operation Manager detects that the firewall instance is receiving too much requests. Currently, there are no additional firewall instances available.

2. NSF Operation Manager requests Developer's Management System to create a new firewall instance.

3. Developer's Management System creates a new firewall instance and then registers the information of the new firewall instance to

NSF Operation Manager.

4. NSF Operation Manager updates the SFC Information Table to reflect the new firewall instance, and notifies NSF and NSFF of this update.

5. According to the new forwarding information, the NSFF forwards the subsequent traffic to the new firewall instance. As a result, we can effectively alleviate the burden of the existing firewall instance.

6. Security Considerations

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

7. Acknowledgements

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

This document has greatly benefited from inputs by Linda Dunbar, Susan Hares, and Hyoungshick Kim.

8. References

8.1. Normative References

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

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

8.2. Informative References

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

[capability-im] Xia, L., Zhang, D., Li, K., Basile, C., Lioy, A., Lopez, D., Lopez, E., Bouthors, N., and L. Fang, "Information Model of Interface to Network Security Functions Capability Interface", draft-xibassnez-i2nsf-capability-00 (work in progress), November 2016.

[i2nsf-framework] Lopez, E., Lopez, D., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-04 (work in progress), October 2016.

[i2nsf-problem] Hares, S., Dunbar, L., Lopez, D., Zarny, M., Jacquenet, C., and J. Jeong, "I2NSF Problem Statement and Use cases", draft-ietf-i2nsf-problem-and-use-cases-03 (work in progress), November 2016.

[i2nsf-terminology] Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-02 (work in progress), October 2016.

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

Authors' Addresses

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

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

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

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

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

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

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

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