Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2019

S. Hyun
Chosun University
J. Jeong
Sungkyunkwan University
J. Park
ETRI
S. Hares
Huawei

July 2, 2018

Service Function Chaining-Enabled I2NSF Architecture
draft-hyun-i2nsf-nsf-triggered-steering-06

Abstract

This document describes an architecture of the I2NSF framework using security function chaining for security policy enforcement. Security function chaining enables composite inspection of network traffic by steering the traffic through multiple types of network security functions according to the information model for NSFs capabilities in the I2NSF framework. This document explains the additional components integrated into the I2NSF framework and their functionalities to achieve security function chaining. It also describes representative use cases to address major benefits from the proposed architecture.

Table of Contents

1. Introduction

To effectively cope with emerging sophisticated network attacks, it is necessary that various security functions cooperatively analyze network traffic [RFC7498][i2nsf-problem][nsf-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 security functions. In order to meet such requirements, besides security policy rules for individual security functions, we need an additional policy about service function chaining (SFC) for network security which determines a set of security functions through which network traffic packets should pass for inspection. In addition, [nsf-capability-im] proposes an information model for NSFs capabilities that enables a security function to trigger further inspection by executing additional security functions 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 chaining between security functions.

In this document, we propose an architecture that integrates additional components from Service Function Chaining (SFC) into the I2NSF framework to support security function chaining. We extend the security controller's functionalities such that it can interpret a high-level policy of security function chaining into a low-level policy and manage them. It also keeps the track of the available service function (SF) instances for security functions and their information (e.g., network information and workload), and makes a decision on which SF instances to use for a given security function chain/path. Based on the forwarding information provided by the security controller, the service function forwarder (SFF) performs network traffic steering through various required security functions. A classifier is deployed for the enforcement of SFC policies given by the security controller. It performs traffic classification based on the polices so that the traffic passes through the required security function chain/path by the SFF.

2. Objective

- Policy configuration for security function chaining: SFC-enabled I2NSF architecture allows policy configuration and management of security function chaining. Based on the chaining policy, relevant network traffic can be analyzed through various security functions in a composite, cooperative manner.

- Network traffic steering for security function chaining: SFC-enabled I2NSF architecture allows network traffic to be steered through multiple required security functions based on the SFC

policy. Moreover, the I2NSF information model for NSFs capabilities [nsf-capability-im] requires a security function to call another security function for further inspection based on its own inspection result. To meet this requirement, SFC-enabled I2NSF architecture also enables traffic forwarding from one security function to another security function.

- Load balancing over security function instances: SFC-enabled I2NSF architecture provides load balancing of incoming traffic over available security function instances by leveraging the flexible traffic steering mechanism. For this objective, it also performs dynamic instantiation of a security function when there are an excessive amount of requests for that security function.

3. Terminology

This document uses the following terminology described in [RFC7665], [RFC7665][i2nsf-terminology][ONF-SFC-Architecture].

- Service Function/Security Function (SF): A function that is responsible for specific treatment of received packets. A Service Function can act at various layers of a protocol stack (e.g., at the network layer or other OSI layers) [RFC7665]. In this document, SF is used to represent both Service Function and Security Function. Sample 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.

- Classifier: An element that performs Classification. It uses a given policy from SFC Policy Manager.

- Service Function Chain (SFC): A service function chain defines an ordered set of abstract service functions and ordering constraints that must be applied to packets and/or frames and/or flows selected as a result of classification [RFC7665].

- Service Function Forwarder (SFF): A service function forwarder is responsible for forwarding traffic to one or more connected service functions according to information carried in the SFC encapsulation, as well as handling traffic coming back from the SF. Additionally, an SFF is responsible for delivering traffic to a classifier when needed and supported, transporting traffic to another SFF (in the same or the different type of overlay), and terminating the Service Function Path (SFP) [RFC7665].

- Service Function Path (SFP): The service function path is a constrained specification of where packets assigned to a certain service function path must be forwarded. While it may be so constrained as to identify the exact locations for packet processing, it can also be less specific for such locations [RFC7665].

- SFC Policy Manager: It is responsible for translating a high-level policy into a low-level policy, and performing the configuration for SFC-aware nodes, passing the translated policy and configuration to SFC-aware nodes, and maintaining a stabilized network.

- SFC Catalog Manager: It is responsible for keeping the track of the information of available SF instances. For example, the information includes the supported transport protocols, IP addresses, and locations for the SF instances.

- Control Nodes: It collectively refer to SFC Policy Manager, SFC Catalog Manager, SFF, and Classifier.

- Service Path Identifier (SPI): It identifies a service path. The classifier MUST use this identifier for path selection and the Control Nodes MUST use this identifier to find the next hop [RFC8300].

- Service Index (SI): It provides a location within the service path. SI MUST be decremented by service functions or proxy nodes after performing the required services [RFC8300].

- Network Service Header (NSH): The header is used to carry SFC related information. Basically, SPI and SI should be conveyed to the Control Nodes of SFC via this header.

- SF Forwarding Table: SFC Policy Manager maintains this table. It contains all the forwarding information on SFC-enabled I2NSF architecture. Each entry includes SFF identifier, SPI, SI, and next hop information. For example, an entry ("SFF: 1", "SPI: 1", "SI: 1", "IP: 192.168.xx.xx") is interpreted as follows: "SFF 1" should forword the traffic containing "SPI 1" and "SI 1" to "IP=192.168.xx.xx".

4. Architecture

This section describes an SFC-enabled I2NSF architecture and the basic operations of service chaining. It also includes details about each component of the architecture.

Figure 1 describes the components of SFC-enabled I2NSF architecture. Our architecture is designed to support a composite inspection of traffic packets in transit. According to the inspection result of each SF, the traffic packets could be steered to another SF for futher detailed analysis. It is also possible to reflect a high-level SFC-related policy and a configuration from I2NSF Client on the components of the original I2NSF framwork. Moreover, the proposed architecture provides load balancing, auto supplementary SF generation, and the elimination of unused SFs. 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.

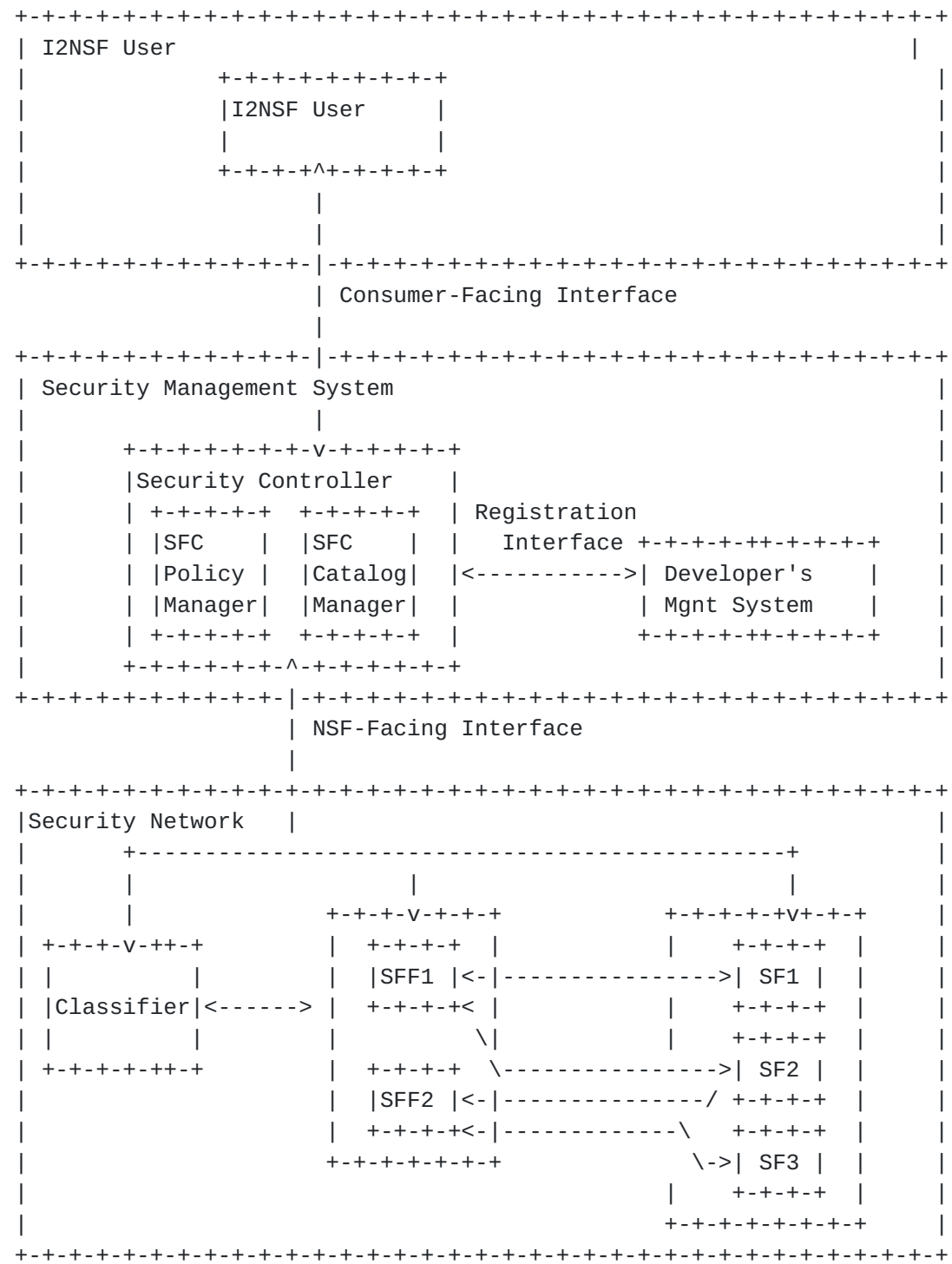


Figure 1: SFC-enabled I2NSF

4.1. SFC Policy Manager

SFC Policy Manager is a core component in our system. It is responsible for the following two things: (1) Interpreting a high-level SFC policy (or configuration) into a low-level SFC policy (or configuration), which is given by I2NSF Client, and delivering the interpreted policy to Classifiers for security function chaining. (2) Generating an SF forwarding table and distributing the fowarding information to SFF(s) by consulting with SFC Catalog Manager. As Figure 1 describes, SFC Policy Manager performs these additional functionalities through Consumer-Facing Interface and NSF-Facing Interface.

Given a high-level SFC policy/configuration from I2NSF Client via Consumer-Facing Interface, SFC Policy Manager interprets it into a low-level policy/configuration comprehensible to Classifier(s), and then delivers the resulting low-level policy to them. Moreover, SFC Policy Manager possibly generates new policies for the flexible change of traffic steering to rapidly react to the current status of SFs. For instance, it could generate new rules to forward all subsequent packets to "Firewall Instance 2" instead of "Firewall Instance 1" in the case where "Firewall Instance 1" is under congestion.

SFC Policy Manager gets information about SFs from SFC Catalog Manager to generate SF forwarding table. In the table generation process, SFC Policy Manager considers various criteria such as SFC policies, SF load status, SF physical location, and supported transport protocols. An entry of the SF forwarding table consists of SFF Identifier, SFP, SI, and next hop information. The examples of next hop information includes the IP address and supported transport protocols (e.g., VxLAN and GRE). These forwarding table updates are distributed to SFFs with either push or pull methods.

4.2. SFC Catalog Manager

In Figure 1, SFC Catalog Manager is a component integrated into Security Controller. It is responsible for the following three things: (1) Maintaining the information of every available SF instance such as IP address, supported transport protocol, service name, and load status. (2) Responding to the queries of available SF instances from SFC Policy Manager so as to help to generate a forwarding table entry relevant to a given SFP. (3) Requesting Developer's Management System to dynamically instantiate supplementary SF instances to avoid service congestion or the elimination of an existing SF instance to avoid resource waste.

Whenever a new SF instance is registered, Developer's Management System passes the information of the registered SF instance to SFC Catalog Manager, so SFC Catalog Manager maintains a list of the information of every available SF instance. Once receiving a query of a certain SFP from SFC Policy Manager, SFC Catalog Manager searches for all the available SF instances applicable for that SFP and then returns the search result to SFC Policy Manager.

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

4.3. Developer's Management System

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

4.4. Classifier

Classifier is a logical component that may exist as a standalone component or a submodule of another component. In our system, the initial classifier is typically located at an entry point like a border router of the network domain, and performs the initial classification of all incoming packets according to the SFC policies, which are given by SFC policy manager. The classification means determining the SFP through which a given packet should pass. Once the SFP is decided, the classifier constructs an NSH that specifies the corresponding SPI and SI, and attaches it to the packet. The packet will then be forwarded through the determined SFP on the basis of the NSH information.

4.5. Service Function Forwarder (SFF)

It is responsible for the following two functionalities: (1) Forwarding the packets to the next SFF/SF. (2) Handling re-classification request from SF.

An SFF basically takes forwarding functionality, so it needs to find the next SF/SFF for the incoming traffic. It will search its forwarding table to find the next hop information that corresponds to the given traffic. In the case where the SFF finds a target entry on its forwarding table, it just forwards the traffic to the next SF/SFF specified in the next hop information. If an SFF does not have an entry for a given packet, it will request the next hop information to SFC Policy Manager with SFF identifier, SPI, and SI information. The SFC Policy Manager will respond to the SFF with next hop information, and then the SFF updates its forwarding table with the response, forwarding the traffic to the next hop.

Sometimes an SF may want to forward a packet, which is highly suspicious, to another SF for futher security inspection. This is referred to as advanced security action in I2NSF. In this situation, if the next SF may not be the one on the current SFP of the packet, re-classification is required to change the SFP of the packet. If the current SF is capable of re-classifying the packet by itself, the SF updates the SPI field in the NSH in the packet to serve the advanced secuity action. Otherwise, if the classifier exists as a standalone, the SF appends the inspection result of the packet to the MetaData field of the NSH and delivers it to the source SFF. The attached MetaData includes a re-classification request to change the SFP of the packet to another SFP for stronger inspection. When the SFF receives the traffic requiring re-classification, it forwards the traffic to the Classifier where re-classification will be eventually performed.

SFC defines Rendered Service Path (RSP), which represents the sequence of actual visits by a packet to SFFs and SFs [RFC7665]. If the RSP information of a packet is available, the SFF could check this RSP information to detect whether undesired looping happened on the packet. If the SFF detects looping, it could notify the Security Controller of this looping, and the Security Controller could modify relevant security policy rules to resolve this looping.

5. Use Cases

This section introduces three use cases for the SFC-enabled I2NSF architecture : (1) Dynamic Path Alternation, (2) Enforcing Different SFPs Depending on Trust Levels, and (3) Effective Load Balancing with Dynamic SF Instantiation.

5.1. Dynamic Path Alternation

In SFC-enabled I2NSF architecture, a Classifier determines the initial SFP of incoming traffic according to the SFC policies. The classifier then attaches an NSH specifying the determined SFP of the packets, and they are analyzed through the SFs of the initial SFP. However, SFP is not a static property, so it could be changed dynamically through re-classification. A typical example is for a certain SF in the initial SFP to detect that the traffic is highly suspicious (likely to be malicious). In this case, the traffic needs to take stronger inspection through a different SFP which consists of more sophisticated SFs.

Figure 2 illustrates an example of such dynamic SFP alternation in a DDoS attack scenario. SFP-1 represents the default Service Function Path that the traffic initially follows, and SFP-1 consists of AVC, Firewall, and IDS/IPS. If the IDS/IPS suspects that the traffic is attempting DDoS attacks, it will change the SFP of the traffic from the default to SFP-2 so that the DDoS attack mitigator can execute a proper countermeasure against the attack.

Such SFP alternation is possible in the proposed architecture with re-classification. In Figure 1, to initiate re-classification, the IDS/IPS appends its own inspection result to the MetaData field of NSH and deliver it to the SFF from which it has originally received the traffic. The SFF then forwards the received traffic including the inspection result from the IDS/IPS to Classifier for re-classification. Classifier checks the inspection result and determines the new SFP (SFP-2) associated with the inspection result in the SFC policy, and updates the NSH with the SPI of SFP-2. The traffic is forwarded to the DDoS attack mitigator.

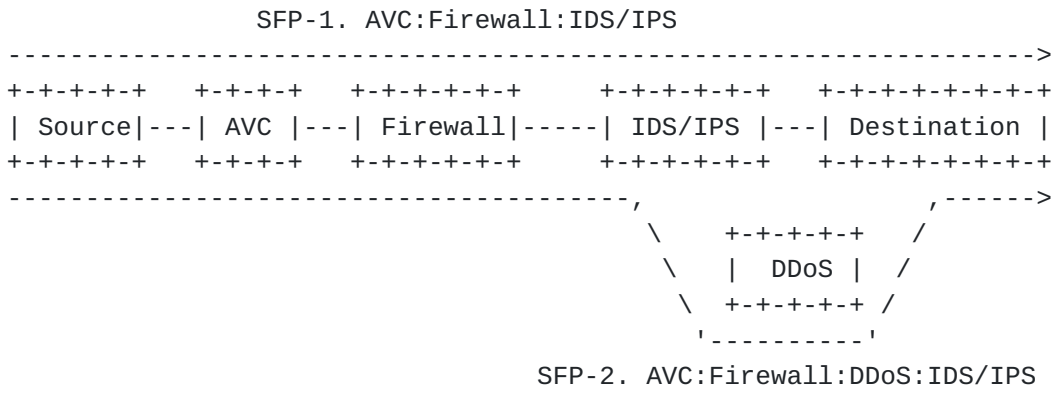
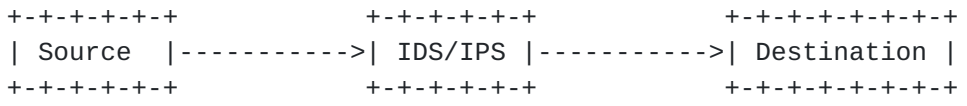


Figure 2: Dynamic SFP Alternation Example

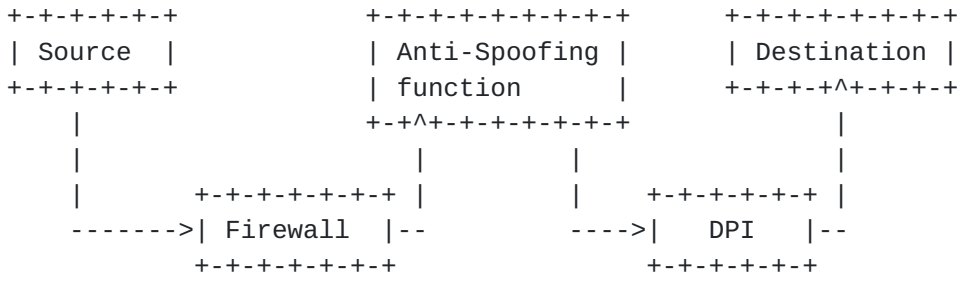
5.2. Enforcing Different SFPs Depending on Trust Levels

Because the traffic coming from a trusted source is highly likely to be harmless, it does not need to be inspected excessively. On the other hand, the traffic coming from an untrusted source requires an in-depth inspection. By applying minimum required security functions to the traffic from a trusted source, it is possible to prevent the unnecessary waste of resources. In addition, we can concentrate more resources on potential malicious traffic. In the SFC-enabled I2NSF architecture, by configuring an SFC Policy to take into account the levels of trust of traffic sources, we can apply different SFPs to the traffic coming from different sources.

Figure 3(a) and Figure 3(b) represent SFPs applicable to traffic from trusted and untrusted sources, respectively. In Figure 3(a), we assume a lightweight IDS/IPS which is configured to perform packet header inspection only. In this scenario, when receiving the traffic from a trusted source, the classifier determines the SFP in Figure 3(a) such that the traffic passes through just a simple analysis by the lightweight IDS/IPS. On the other hand, traffic from an untrusted source passes more thorough examination through the SFP in Figure 3(b) which consists of three different types of SFs.



(a) Traffic flow of trusted source



(b) Traffic flow of untrusted source

Figure 3: Different path allocation depending on source of traffic

5.3. Effective Load Balancing with Dynamic SF Instantiation

In a large-scale network domain, there typically exist a large number of SF instances that provide various security services. It is possible that a specific SF 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 security function. If there are no additional instances of the same security function available, we need to create a new SF 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 the SFC-enabled I2NSF architecture, SFC Catalog Manager performs periodic monitoring of the load status of available SF instances. In addition, it is possible to dynamically generate a new SF 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. SFC Catalog Manager detects that the firewall instance is receiving too much requests. Currently, there are no additional firewall instances available.

2. SFC Catalog 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 SFC Catalog Manager.

4. SFC Catalog Manager updates the SFC Information Table to reflect the new firewall instance, and notifies SFC Policy Manager of this update.

5. Based on the received information, SFC Policy Manager updates the forwarding information for traffic steering and sends the new forwarding information to the SFF.

6. According to the new forwarding information, the SFF forwards the subsequent traffic to the new firewall instance. As a result, we can effecively alleviate the burden of the existing firewall instance.

6. Discussion

The information model and data model of security policy rules in the I2NSF framework defines an advanced security action as a type of action to be taken on a packet [nsf-capability-im][nsf-facing-inf-dm]. Through the advanced security action, a basic NSF (e.g., firewall) can call a different type of NSF for more in-depth security analysis of a packet. If an NSF triggers an advanced security action on a given packet, the packet should be forwarded to the NSF dedicated to the advanced action. That is, the advanced action dynamically determines the next NSF where the packet should go through. So if a forwarding component is configured with the network access information (e.g., IP address, port number) of the next required NSF, it can forward the packet to the NSF. With this advanced security action, it is possible to avoid the overhead for configuring and managing the information of security function chains and paths.

In SFC, re-classification is required to support the situation where the security function path of a packet changes dynamically, and the classifier is responsible for re-classification tasks to change the security function path of a packet. But if the classifier exists as a separate component from an NSF, the packet should be first delivered from the NSF to the classifier for re-classification, and this introduces an additional delay. As already mentioned, the advanced security action in the i2nsf framework can omit the requirement of pre-defined security function chain configuration. If there exists no security function chain/path configurations, there is no need of re-classification as well. That is, the forwarder can simply forward the packet to the next required NSF according to the advanced action determiend by the predesessor NSF, without re-classification through the classifier.

7. Implementation Considerations

7.1. SFC Policy Configuration and Management

In the SFC-enabled I2NSF architecture, SFC policy configuration and management should be considered to realize NSF chaining for packets. According to the given SFC policy, a classifier is configured with the corresponding NSF chain/path information, and also an SFF is configured with a forwarding information table.

The following three interfaces need to be considered for SFC policy configuration and management. First of all, the network administrator, typically an I2NSF user, needs to send SFC policy configuration information that should be enforced in the system to the security controller. Thus an interface between the network

administrator and the security controller should be set up for this objective. By analyzing the given SFC policy configuration information, the security controller generates NSF chain/path information required for classifiers and forwarding information table of NSFs that SFFs require for packet forwarding. An interface between the security controller and classifier is required to deliver NSF chain/path information to the classifier. In addition, an interface between the security controller and SFF is also required to deliver forwarding information table of NSFs to SFFs.

When there are multiple instances of classifiers and SFFs, synchronizing the configuration information over them is important for them to have a consistent view of the configuration information. Therefore it should be considered how to synchronize the configuration information over the classifiers and SFFs.

7.2. Placement of Classifiers

To implement the SFC-enabled I2NSF architecture, it needs to be considered where to place the classifier. There are three possible options: NSF, SFF, and a separate component. The first option is integrating a classifier into every NSF. This approach is good for re-classification, because each NSF itself can perform re-classification without introducing any additional network overhead. On the other hand, configuring every NSF with NSF chain/path information and maintaining their consistency introduce an extra overhead. The second option is integrating a classifier into a SFF. In general, since the number of SFFs is much smaller than the number of NSFs, the overhead for configuring and managing NSF chain/path information would be smaller than the first option. In this case, re-classification of a packet should be done at a SFF rather than an NSF. The third option is that a classifier operates as a standalone entity. In this case, whenever re-classification is required for a packet, the packet should first stop by the classifier before going through a SFF, and this is likely to increase packet delivery latency.

7.3. Forwarding Method of SFC

Tunneling protocols can be utilized to support packet forwarding between SFF and NSF or SFC proxy [RFC7665] . In this case, it needs to be considered which tunneling protocol to use, and both SFF and NSF/SFC proxy should support the same tunneling protocols. If an NSF itself should handle the tunneling protocol, it is required to modify the NSF implementation to make it understand the tunneling protocol. When there are diverse NSFs developed by different vendors, how to modify efficiently those NSFs to support the tunneling protocol is an

critical issue to reduce the maintenance cost of NSFs after deployment.

Network Tunneling in SFC is achieved through protocols such as [RFC8300], [RFC3032], [StEERING], and so on. To implement the SFC-enabled I2NSF architecture, it needs to be considered what to use something for the network tunneling.

[RFC8300] is a data plane protocol to deliver the information between entity of sfc-enabled domain; it sends the service path identification and metadata. Its protocol is an independent transport protocol. Therefore, another tunneling protocol([RFC2890] [RFC2003] [RFC3378] [vxlan-gpe]) should be inserted over the NSH header and encapsulation for forwarding within the network. [RFC3032] construct the routing mechanism based on the MPLS label stack. When an MPLS-label packet enters the SFC-enabled domain, a classifier encapsulates the mpls label stack into a packet based on the SFC information. [StEERING] is a traffic triggered steering framework based on software defined network. An SDN controller defines policy control traffic using the subscriber information and requirements, traffic type, and so on. And propagate that control traffic between SFC-enabled domain entities.

7.4. Implementation of Network Tunneling

We implemented network tunnleing based on GRE (Generic Routing Encapsulation) protocol to support packet forwarding between SFF and SFC proxy. For the NSH encapsulation with GRE protocol in layer 3, we referred to the header format defined in [RFC8300]. Figure 4 shows the format of an entire packet in our implementation, and Figure 5 shows the mapping table of service path identifiers used in our implementation.

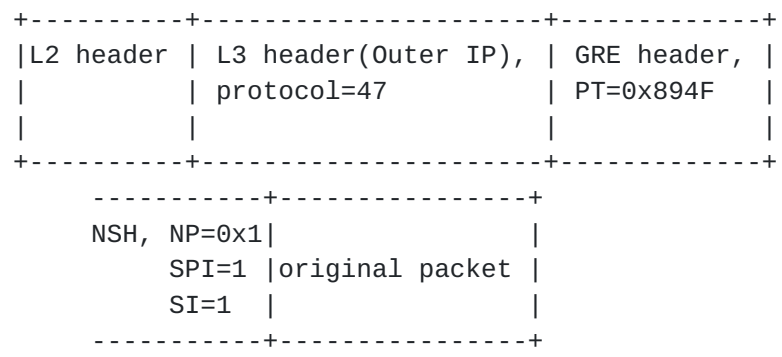


Figure 4: Entire packet format for network tunneling based on GRE protocol

SPI	Network security function
1	Firewall
2	Firewall->DPI
3	Firewall->DPI->DDoS metigation

Figure 5: Mapping table of service path identifiers

7.5. Implementation of SFC using Opendaylight Controller

Traffic steering in I2NSF framework can be implemented by using Opendaylight that supports service function chaining. In such a system where Opendaylight is integrated into I2NSF framework, traffic steering can be performed with three functions as follows. 1) I2NSF Security Controller Function 2) SDN Switch Controller Function 3) SDN Switch Traffic Steering Function. The following describes each of these functions.

What service function chains (SFC) are needed can be determined according to security policy rules of NSFs in I2NSF framework. I2NSF Security Controller Function identifies NSF chains that are required in the system by comprehensively analyzing security policy rules of NSFs. I2NSF Security Controller Function then generates the policies of the identified NSF chains, and requests SDN Switch Controller to enforce the policies of NSF chains.

SDN Switch Controller Function is responsible for creating, updating, and deleting SFCs, while Switch Controller operates to support service function chaining. Opendaylight Switch Controller is able to create key elements for SFC such as SF, SFF, SFC, SFP, and RSP. Once receiving the SFC policies from I2NSF Security Controller Function, SDN Switch Controller Function generates traffic forwarding rules that need to be configured on SFFs and switches, in order for the requested SFC policies to be enforced to relevant traffic. The generated traffic forwarding rules are delivered to relevant SFFs and switches using a data plane protocol ([OpenFlow], [RFC7047],[RFC6241]).

SFFs and switches perform forwarding traffic based on the traffic forwarding rules received from SDN Switch Controller. SDN Switch Traffic Steering Function refers to the Data Plane Elements processes running on SFFs and switches for steering traffic to the destination according to the traffic forwarding rules. To steer packets over

NSFs, the packets are encapsulated with Network Service Header (NSH) [RFC8300].

8. Security Considerations

To enable 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.

9. Acknowledgments

This work was supported by Institute for Information and 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 Sanguk Woo, Yunsuk Yeo, Taekyun Roh, and Sarang Wi.

10. References

10.1. Normative References

[OpenFlow] Open Networking Foundation, "The OpenFlow Switch Specification, Version 1.4.0", OpenFlow https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf, October 2013.

[RFC2003] Perkins, C., "IP Encapsulation within IP", RFC 2003, October 1996.

[RFC2890] Dommety, G., "Key and Sequence Number Extensions to GRE", RFC 2890, September 2000.

[RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, January 2001.

[RFC3378] Housley, R. and S. Hollenbeck, "EtherIP: Tunneling Ethernet Frames in IP Datagrams", RFC 3378, September 2002.

[RFC6241] Enns, R. and M. Bjorklund, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC7047] Pfaff, B. and B. Davie, "The Open vSwitch Database Management Protocol", RFC 7047, December 2013.

[RFC7665] Halpern, J. and C. Pignataro, "Service Function Chaining (SFC) Architecture", RFC 7665, October 2015.

[RFC8300] Quinn, P., Elzur, U., and C. Pignataro, "Network Service Header (NSH)", RFC 8300, January 2018.

[StEERING] Zhang, Y., Beheshti, N., Beliveau, L., Lefebvret, G., Manghirmalani, R., Mishra, R., Patney, R., Shirazipour, M., Subrahmaniam, R., Truchan, C., and M. Tatipamula, "StEERING: A Software-Defined Networking for Inline Service Chaining", October 2013.

[vxlan-gpe] Maino, F., Kreeger, L., and U. Elzur, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-06 (work in progress), April 2018.

10.2. Informative References

[i2nsf-framework] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", RFC 8329, February 2018.

[i2nsf-problem] Hares, S., Lopez, D., Zarny, M., Jacquenet, C., Kumar, R., and J. Jeong, "I2NSF Problem Statement and Use cases", RFC 8192, July 2017.

[i2nsf-terminology] Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-05 (work in progress), January 2018.

[nsf-capability-im] Xia, L., Strassner, J., Basile, C., and D. Lopez, "Information Model of NSFs Capabilities", draft-ietf-i2nsf-capability-01 (work in progress), April 2018.

[nsf-facing-inf-dm]
Kim, J., Jeong, J., Park, J., Hares, S., and L. Xia, "I2NSF Network Security Functions-Facing Interface YANG Data Model", draft-kim-i2nsf-nsf-facing-interface-data-model-04 (work in progress), October 2017.

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

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

Appendix A. Changes from draft-hyun-i2nsf-nsf-triggered-steering-05

The following changes have been made from draft-hyun-i2nsf-nsf-triggered-steering-05:

- Section 7.3 has been added to discuss an forwarding method that supports SFC.
- The references have been updated to reflect the latest documents.

Authors' Addresses

Sangwon Hyun
Department of Computer Engineering
Chosun University
309, Pilmun-daero, Dong-gu
Gwangju, Jeollanam-do 61452
Republic of Korea

EMail: shyun@chosun.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

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

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

Phone: +1 734 604 0332
EMail: shares@ndzh.com