Interface to Network Security Functions for Cloud-Based Security Services

Sangwon Hyun, Jinyong Kim, Hyoungshick Kim, Jaehoon (Paul) Jeong, Susan Hares, Linda Dunbar, and Adrian Farrel

The authors present the design and development of an I2NSF architecture and propose improving its efficiency by integrating it with SDN. In this work, they implement the SDN-integrated I2NSF architecture and its security applications. They also discuss several standardization and research challenges for I2NSF.

Abstract

Network functions virtualization and cloud-based security services will become increasingly common in enterprise network systems to reduce the system operation costs and take advantage of the diverse network security functions (NSFs) developed by multiple vendors. In such a network environment, standardizing the interfaces to the NSFs of different vendors is essential to simplify the management of these heterogeneous NSFs. In addition, software-defined networking can be imposed to optimize the security service process in such cloud-based service environments by enforcing some types of packet filtering rules at the SDN switches, instead of NSFs possibly placed in remote clouds. The Interface to Network Security Functions (I2NSF) Working Group, which is part of the Internet Engineering Task Force, is currently developing a set of standard interfaces to such heterogeneous NSFs. In this article, we present the design and development of an I2NSF architecture and propose improving its efficiency by integrating it with SDN. In our work, we implement the SDN-integrated I2NSF architecture and its security applications. This article also discusses several standardization and research challenges for I2NSF.

Introduction

Recent advances in cloud computing and network functions virtualization (NFV) technologies [1] have made it feasible to provide network services through virtual service functions running on cloud servers. It is also possible to outsource these virtual service functions to third-party solution vendors [2]. This cloud-based service provisioning model offers several advantages such as cost saving and flexible and efficient resource utilization. This service model is particularly useful for providing network security services to the users. For example, in the scenario of a distributed denial of service (DDoS) attack, it is possible to rapidly and flexibly respond to the intensive attack traffic by dynamically increasing the number of DDoS mitigation instances. Moreover, this cloud-based security service model facilitates the deployment of various security functions developed by multiple security solution vendors. This is well suited to satisfy the growing requirement of enterprise network systems to integrate these security functions to build more secure systems. Consequently, the demand for cloud-based security services is increasing [3].

Network security functions (NSFs) developed by different vendors have different interfaces for their configuration and management because there is no industry standard of interfaces to NSFs. This heterogeneity introduces complexity in the management of the NSFs of multiple vendors, resulting in increased management costs. Therefore, standardization is essential to successfully deploy the NSFs offered by various vendors. Recently, some standardization activities have been in progress in the Internet Engineering Task Force Interface to Network Security Functions (IETF I2NSF) Working Group [4, 5] to satisfy these demands (i.e., to develop standard interfaces for the NSFs).

In this article, we present the design and development of an I2NSF architecture that provides standard interfaces for the efficient control and management of the NSFs offered by diverse vendors in cloud-based security service environments. Being active members of the IETF I2NSF Working Group, we are developing a set of standard information and data models that are the key to building the standard interfaces in the I2NSF architecture. Based on these standard information and data models, the I2NSF architecture is able to provide an efficient and flexible security service environment that is driven by the security policies expressed in the information and data models.

Software-defined networking (SDN) [6] allows dynamic and flexible changes in network behavior by controlling and managing the configurations of the network resources, such as switches and routers, programmatically. This capability makes it feasible to enforce some packet filtering rules at the switches by controlling their packet forwarding rules.

In this article, we propose to integrate an I2NSF architecture with SDN, and divide the application of the security policy rules among the SDN switches and NSFs by taking advantage of the SDN technology. Specifically, switches enforce simple packet filtering rules that can be translated into their packet forwarding rules, whereas NSFs enforce NSF-related security rules requiring the security capabilities of the NSFs. Thus, if switches can make decisions on some received packets according to their packet forwarding rules programmed by a switch controller, we can avoid unnecessary latency for the packets taken by an NSF for a time-consuming inspection task. In addition, because all the packets do not necessarily

Digital Object Identifier: 10.1109/MCOM.2018.1700662

Sangwon Hyun, Jinyong Kim, Hyoungshick Kim, and Jaehoon (Paul) Jeong (corresponding author) are with Sungkyunkwan University; Susan Hares and Linda Dunbar are with Huawei; Adrian Farrel is with Juniper Networks.

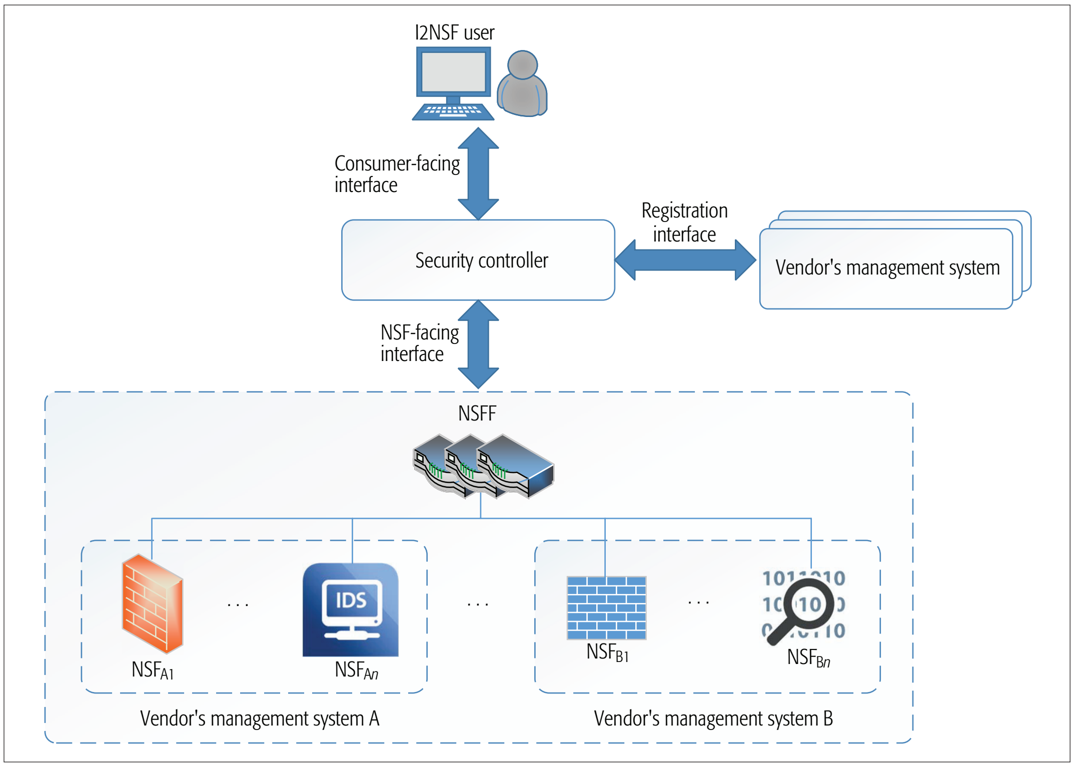

Figure 1. I2NSF architecture.

The major objective of the NSF-Facing interface is to provide standard interfaces to control and manage the NSFs of various vendors. For this objective, we developed generic standard information and data models specifying vendor-neutral syntax rules and data structures to comprehensively express the security policy rules for the NSFs of different vendors that have similar security capabilities.

pass through an NSF, we can reduce the possibility of congestion in an NSF. In this work, we implement an I2NSF architecture integrated with an SDN network.

The contributions of this article are as follows. First, we develop an I2NSF architecture and a set of standard interfaces in that architecture to enable efficient and flexible security service provisioning in cloud-based security service environments. Second, we propose a seamless integration of the I2NSF architecture and SDN to improve the efficiency of the security service enforcement process. Third, we implement the SDN-integrated I2NSF architecture of two example security service scenarios. Finally, we discuss several standardization and research challenges of I2NSF.

The remainder of this article is organized as follows. The next section describes the I2NSF architecture and its integration with SDN. Then we present the proof-of-concept implementation of the SDN-integrated I2NSF architecture and its security applications. Following that, we discuss several standardization and research challenges for I2NSF. Finally, we conclude the article.

I2NSF Architecture

Components

Figure 1 shows the I2NSF architecture consisting of the following five components: *I2NSF user, security controller, vendor's management system (VMS), NSF forwarder (NSFF),* and *NSF*.

I2NSF User: The I2NSF user in the figure represents the administrator of the enterprise network where a security policy should be enforced. The I2NSF user describes a high-level security policy without considering the underlying NSFs, and sends a request for the high-level security policy to the security controller.

Network Operator Management Security Controller (Security Controller): The network operator management security controller is the most critical component, which orchestrates and manages the NSFs so that they cooperatively enforce the security policy requested by an I2NSF user. After receiving a high-level security policy request from an I2NSF user, the security controller determines the types of NSFs that are required to enforce the security policy, generates low-level security policy rules for each of the required NSFs, and finally updates the configuration of each NSF with the generated policy rules. In addition, the security controller monitors the NSFs running in the system and maintains various information related to each NSF (e.g., network access information and workload status).

Vendor's Management System (VMS): The VMS is managed by a third-party security vendor offering NSFs. The VMS performs the dynamic life cycle management of its NSF instances in response to the requests of the security controller. As depicted in Fig. 1, there can be multiple VMSs (e.g., VMS A and B) for different security vendors, and this enables an I2NSF user to take advantage of the various security functions offered by multiple vendors.

NSF and NSF Forwarder (NSFF): An NSF, such as a firewall, deep packet inspection (DPI), or DDoS attack mitigator, performs the security inspection of the network traffic according to the security policy rules provided by the security controller. A basic NSF (e.g., firewall) within the I2NSF architecture can trigger an advanced security inspection (e.g., DPI and DDoS attack mitigation) with a different type of NSF based on its own security inspection result; for example, a firewall triggers a further inspection of suspi-

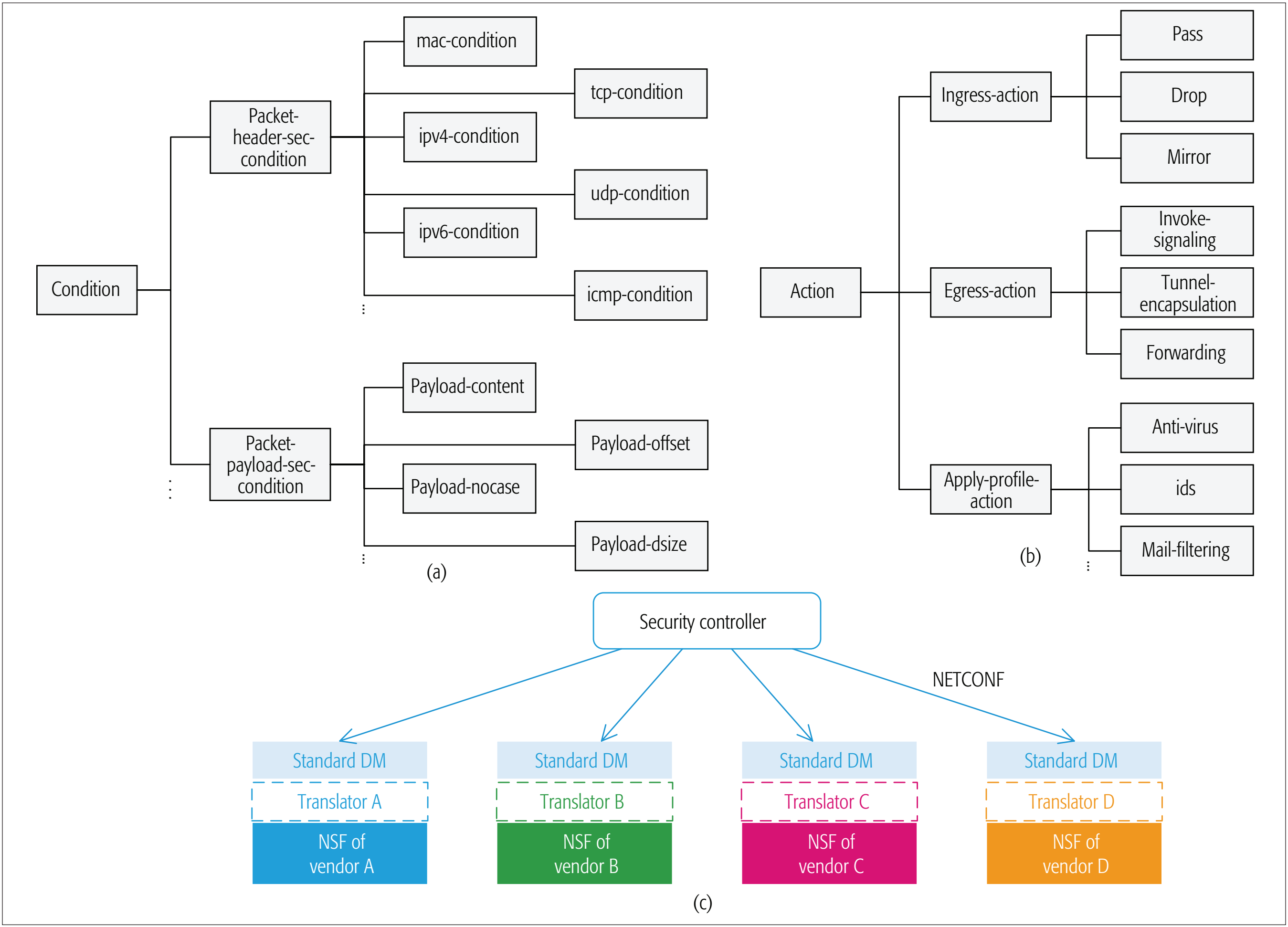

Figure 2. Standardizing interfaces to the NSFs based on a generic standard information model and a data model: a) condition subtree of the information model for the network security group of NSFs; this subtree describes inspection rules of various header fields and payloads of packets; b) action subtree of the information model for the network security group of NSFs; this subtree describes an action to be taken when a given packet meets the specified conditions; c) standard interfaces to the NSFs of multiple vendors using the YANG data model.

cious traffic with a DPI. In such cases, the NSFF forwards the suspicious traffic from the current NSF (e.g., firewall) to the successor NSF (e.g., DPI or DDoS attack mitigator) to realize an advanced and composite inspection. The NSFF is a logical component in that it can coexist with either the security controller or the NSF.

INTERFACES

This section describes the three interfaces of the I2NSF architecture shown in Fig. 1, namely, *consumer-facing interface*, *NSF-facing interface*, and *registration interface*.

Consumer-Facing Interface: As shown in Fig. 1, the consumer-Facing interface is the only interface of the I2NSF user to the I2NSF system. Such a design hides the underlying NSFs from the users and allows them to specify a user-friendly high-level security policy without considering the details of the NSFs. The following is an example of a high-level security policy of time-dependent web access control: *Block the access of company employees to Facebook from 9am to 6pm* [7]. This high-level security policy does not require specific information about network resources and protocols, and thus allows the users to define their security requirements in a user-friendly manner.

The major purpose of this interface is to allow the users to request security services from the I2NSF system by sending a high-level security policy to the security controller. In addition, the security controller notifies the users of critical security alerts received from the NSFs via this interface. By analyzing the received alerts, the users can identify new attacks and update (or generate) a high-level security policy to respond to the new attacks. We developed standard information and data models of the consumer-facing interface to enable these functionalities (draft-jeong-i2nsf-consumer-facing-interface-dm-05).

NSF-Facing Interface: The major objective of the NSF-facing interface is to provide standard interfaces to control and manage the NSFs of various vendors. For this objective, we developed generic standard information and data models specifying *vendor-neutral* syntax rules and data structures to comprehensively express the security policy rules for the NSFs of different vendors that have similar security capabilities. (See below for more details of the information and data models.) In addition, the security policy rules described according to the standard are delivered using a standard protocol such as Network Configuration Protocol (NETCONF) [8]). Consequently, the security controller does not have to consider vendor-specific differences between the NSFs when configuring them.

Each NSF also uses this interface to periodically notify the security controller of its current status (e.g., workload level and congestion) and of the suspicious security events/alerts detected by it. In addition, the security controller delivers the forwarding configuration information of the NSFs running in the system to the NSFF via this interface.

Registration Interface: The main purpose of the registration interface between the security controller and VMS is to perform dynamic life cycle management of the NSF instances and register new NSF instances into the system. We developed standard information and data models for introducing these functionalities. If a new NSF instance is required by the system, the security controller requests the VMS to create it. Such a request from the security controller contains the description of the security capability and service capacity that should be provided by the requested NSF instance. After creating it, the VMS informs the security controller of its identifying information (IP address, port number, etc.). In contrast with the above case, if some existing NSF instance is underutilized, the security controller may request the VMS to remove it via this interface.

NSF-Triggered Traffic Steering

In the I2NSF architecture, an NSF can trigger an advanced security action (e.g., DPI and DDoS attack mitigation) on a packet based on the result of its own security inspection of the packet. For example, a firewall triggers further inspection of a suspicious packet with DPI. For this advanced security action to be realized, the suspicious packet should be forwarded from the current NSF to the successor NSF. Service function chaining (SFC) [9] is a technology that enables this triggering by steering a packet with multiple service functions (e.g., NSFs). The I2NSF architecture can adopt similar approaches to support the advanced security action. To trigger an advanced security action, the current NSF appends the metadata describing the security capability required for the advanced action to the suspicious packet and sends the packet to the NSFF.

The NSFF maintains a forwarding information table of the NSF instances available in the system. Each entry of the table contains the network access information and security capability description of the NSF. When receiving a packet to be forwarded for an advanced security action, the NSFF searches its local table for a matching NSF instance that can perform the required security action on the packet. If the NSFF cannot locate any matching entry in its local table, it asks the security controller of an NSF instance with the required security capability, and the security controller responds to the NSFF with the information of a matching NSF instance. Finally, the NSFF forwards the packet to the discovered NSF instance using the forwarding information either from its local table or from the security controller.

Information Model and Data Model for Standard Interfaces to NSFs

Standardizing interfaces to the heterogeneous NSFs developed by different security vendors is highly essential for reducing management costs and improving system interoperability. To standardize the interfaces, first of all, the NSFs of the various vendors are classified into three groups according to their security capabilities: *network security* (e.g., firewall filter), *content security* (e.g., anti-malware scanner), and *attack mitigation* (e.g., DDoS mitigator). The *network security* group of NSFs performs the inspection of layer 2-4 (L2–L4) headers and payloads of packets, whereas the *content security* group of NSFs performs the inspection of packet contents in the application layer, for example, to detect malware and malicious URLs. The *attack mitigation* group of NSFs performs detecting and mitigating DDoS attacks.

We define a generic standard *information model* (IM) for each group of NSFs that can comprehensively express security policy rules of the NSFs. Figures 2a and 2b show the key parts of the information model defined for the *network security* group of NSFs that performs the inspection of various header fields and payloads of packets. The information model defines information items and a link to them to express the core semantics of the security policy rules of the NSFs (of the various vendors) belonging to the same security capability group. For example, the different vendors' NSFs in the same network security group are expected to use different formats of the security policy rules. However, these security policy rules in diverse formats can be generalized to provide the same security services with the description of a set of conditions to be evaluated for various headers and payloads of a packet and a set of actions to be taken when the conditions are satisfied for the packet. Such an abstraction and generalization result in a generic information model for the same security capability group of NSFs.

Figure 2a shows the condition subtree of the IM for the *network security* group of NSFs, and this condition subtree is used to define a set of conditions that specify attributes and/or values that are to be compared to the headers and payloads of a given packet. The action subtree in Fig. 2b is used to define the action (i.e., ingress and egress actions) to be taken when a given packet satisfies the specified conditions. In particular, we can specify an advanced security action that triggers a further in-depth analysis using another type of NSF, such as an anti-virus and intrusion detection system (IDS), in addition to simple actions such as pass and drop.

The next step is to implement the IM using a data modeling language (e.g., YANG [10]), and the resulting implementation is referred to as a *data model* (DM). Our implementation of the IM using the YANG data modeling language (i.e., the YANG data model) is described in an Internet-Draft (draft-kim-i2nsf-nsf-facing-interface-data-model-04). As illustrated in Fig. 2c, each vendor (represented in a different color) implements the standard DM for its NSF(s) and pre-configures the relevant NSF(s) with the DM implementation. Then the NSFs can understand the security policy rules described according to their DM.

In the I2NSF architecture, an NSF can trigger an advanced security action (e.g., DPI and DDoS attack mitigation) on a packet based on the result of its own security inspection of the packet. For example, a firewall triggers a further inspection of a suspicious packet with a DPI. For this advanced security action to be realized, the suspicious packet should be forwarded from the current NSF to the successor NSF.

In this system, we divide the enforcement of security policy rules into the SDN switches and NSFs. In particular, SDN switches enforce simple packet filtering rules that can be translated into their packet forwarding rules, whereas NSFs enforce NSF-related security rules requiring the security capabilities of the NSFs.

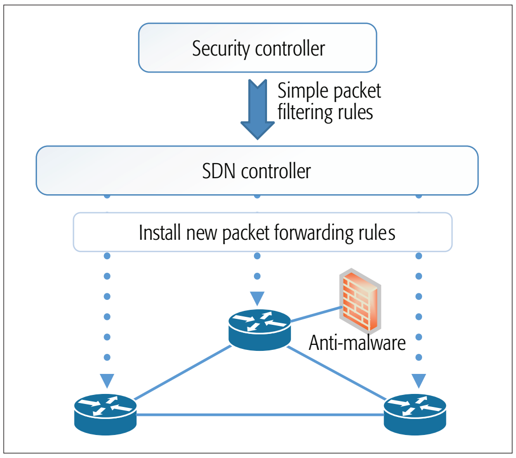
Security controller
Simple packet filtering rules
SDN controller
Install new packet forwarding rules
Anti-malware

Figure 3. Integrating the I2NSF system into an SDN network.

When the security controller is required to configure an NSF with a new security policy rule, it first describes the security policy rule using XML according to the syntax definition in the corresponding DM. The XML-encoded security policy rule is delivered to the NSF using NETCONF protocol [8]. The NSF decodes the received policy rule based on the pre-configured matching DM and extracts the rule data from it. Optionally, the NSF can perform an additional process (denoted as *Translator* in Fig. 2c) of translating the received rule data into its vendor-specific format if necessary. Finally, the NSF registers the new security policy rule in its own policy rule table to perform the required security service.

Integrating SDN with I2NSF Architecture

SDN enables some packet filtering rules to be enforced in the network switches by controlling their packet forwarding rules [11–13]. This section presents how we can take advantage of this capability of SDN to optimize the process of security service enforcement in the I2NSF system.

Figure 3 shows the system configuration in which the I2NSF architecture is integrated into the SDN network.

In this system, we divide the enforcement of security policy rules into the SDN switches and NSFs. In particular, SDN switches enforce simple packet filtering rules that can be translated into their packet forwarding rules, whereas NSFs enforce NSF-related security rules requiring the security capabilities of the NSFs.

As an example, let us consider two different types of security rules: *Rule 1* is a simple packet filtering rule that checks only the IP address and port number of a given packet, whereas *rule 2* is a time-consuming packet inspection rule for determining whether an attached file delivered through a flow of packets contains malware. *Rule 1* can be translated into packet forwarding rules of SDN switches and thus be enforced by the switches. In contrast, *rule 2* cannot be enforced by switches, but it can be enforced by NSFs with anti-malware capability. Specifically, a flow of packets is forwarded to and reassembled by an NSF to reconstruct the attached file stored in the packets. Then the NSF scans the file to check the existence of malware. If the file contains malware, the NSF drops the packets.

In this way, if the switches can make decisions on some received packets according to their packet forwarding rules programmed by the switch controller, we can avoid unnecessary latency for the packets taken by an NSF for a time-consuming task, which may be placed in a remote cloud system. In addition, we can reduce the possibility of congestion in an NSF because all the packets do not necessarily pass through an NSF.

Proof-of-Concept Implementation

This section describes our proof-of-concept (PoC) implementation of the proposed architecture using various open source software, and two security service scenarios using the implementation: VoIP security service and time-dependent web access control service. (The details of these two scenarios will be presented below.) To set up an SDN network of our system components, we used OpenDaylight (https://www.opendaylight.org) and Mininet (http://mininet.org). Figure 4 shows the network configuration of our system and the VoIP security service procedure. Although the current PoC includes only two types of NSFs (i.e., firewall and a VoIP intrusion prevention system, IPS), it can be extended with other types of NSFs (depicted by the dashed rectangles in Fig. 4) according to security service requirements. When various types of NSFs exist, the firewall initially performs a basic security inspection of a packet and triggers a further inspection of a suspicious packet with other types of NSFs (e.g., VoIP IPS, anti-malware scanner).

The summary of our implementation is as follows. We implemented a web application for an I2NSF user that provides a user interface to configure and generate a high-level security policy. We implemented the security controller to translate the received high-level security policy into low-level security rules for the NSFs (e.g., firewall and VoIP IPS). We implemented the consumer-facing interface based on the RESTCONF protocol [14] using Express.js (https://expressjs.com) and the NSF-facing interface based on the NETCONF protocol [8] using ConfD (https://developer.cisco.com/site/confD). We also implemented the YANG data model that defines the syntax rules to express the low-level rules in XML, and the low-level rules expressed in XML are sent to the NSFs through the NETCONF-based NSF-facing interface.

Our source codes and documents of the above implementation are available at https://github.com/kimjinyong/i2nsf-framework, and a video demo is also available at https://youtu.be/5iflpVt4l6U.

VoIP Security Service

This service scenario assumes that the administrators of mobile telecommunications service providers want to block malicious VoIP calls to their customers. The following is a hypothetical example of a high-level security policy rule that the administrator (i.e., I2NSF user) requests: *Inspect suspicious VoIP calls of unusual patterns (e.g., incoming calls from unusual foreign countries at unusual times of a day), and block the calls identified as malicious ones*. The administrator sends this high-level security policy to the security

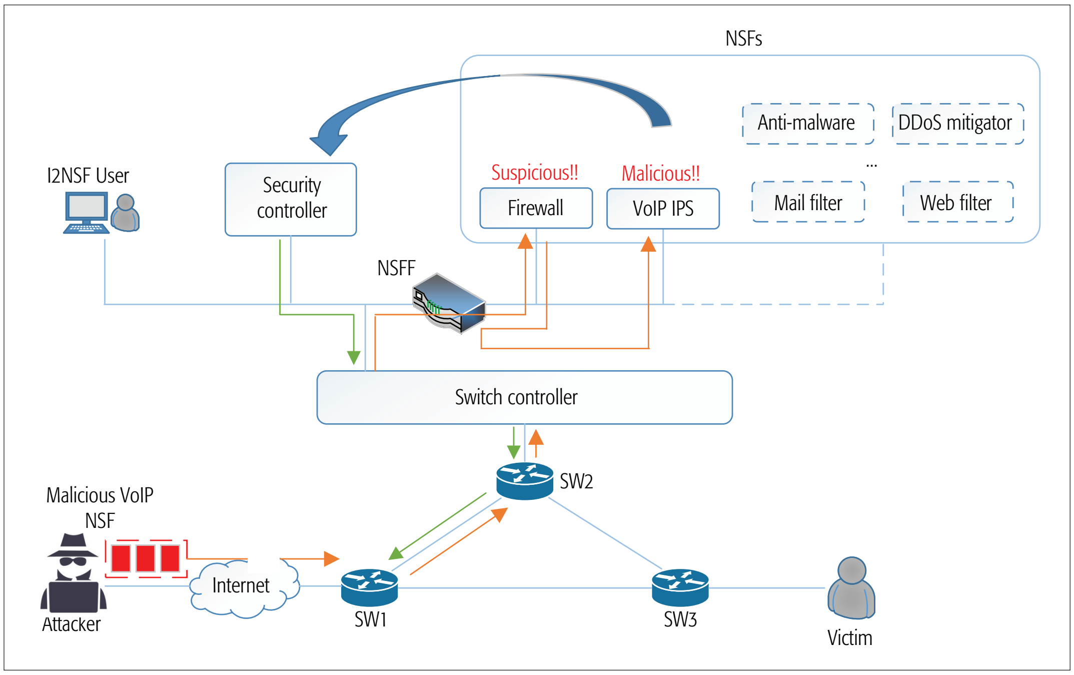

Figure 4. Proof-of-concept implementation and VoIP security service procedure.

This service scenario requires the dynamic update of security rules. To achieve this, the security controller monitors events that require security rule changes, and can automatically update or generate an appropriate rule if any event occurs.

controller through the consumer-facing interface, and the security controller translates the received high-level security policy into low-level security rules for both the firewall and the VoIP IPS. In this scenario, the firewall is configured with security rules to identify VoIP packets of suspicious patterns, and if the firewall detects such a packet, it calls the VoIP IPS for a further inspection of the packet. Then the VoIP IPS analyzes the suspicious packet based on pre-configured security rules to determine whether or not it is malicious.

Figure 4 illustrates the following procedure of the VoIP security service:

1. An unknown flow's packet is received by the switch (SW) 1.
2. SW1 forwards the received packet to the firewall via the switch controller for basic security inspection.
3. The firewall checks the basic header fields (IP address, port number, etc.) of the packet, and identifies that this packet is a VoIP signal packet (e.g., Session Initiation Protocol, SIP, packet) of a suspicious pattern.
4. The firewall requests of the VoIP IPS more in-depth analysis of the suspicious VoIP packet, and the NSFF forwards the packet to the VoIP IPS.
5. The VoIP IPS analyzes the headers and contents of the SIP packet such as the caller's ID and session description headers, and if it identifies the packet as a malicious call packet, it consequently drops the packet.
6. The VoIP IPS also notifies the security controller of the detected malicious call packet.
7. The security controller requests that the switch controller block the subsequent packets from the malicious source (i.e., the attacker).
8. The switch controller installs new forwarding rules into the switches accordingly.

Time-Dependent Web Access Control Service

This service scenario assumes that an enterprise network administrator wants to regulate the access of staff members to Facebook during business hours. The following is a hypothetical example of a high-level security policy rule the administrator requests: *Block access of staff members to Facebook from 9am to 6pm.* The administrator sends the high-level security policy to the security controller, which in turn translates it into low-level packet filtering rules for both the firewall and the switch controller. For this translation, the security controller can interoperate with the enterprise network access control server in order to retrieve the information (e.g., IP address in use, company ID, role) of each employee that is currently using the network. Based on the retrieved information, the security controller generates low-level packet filtering rules to block the access to Facebook from the IP addresses being used by the staff members during the business hours.

This service scenario requires the dynamic update of security rules. To achieve this, the security controller monitors events that require security rule changes, and can automatically update or generate an appropriate rule if any event occurs. For example, if a new staff member joins the enterprise network, the security controller detects this event through the enterprise network access control server, and generates and installs a new rule to enforce the same policy to the new staff member.

Challenges

This section discusses several standardization and research challenges for I2NSF, which are summarized in Table 1.

Standardization Challenges: It is important to monitor the execution status of the NSFs to ensure that the security policies requested by the I2NSF users are being properly enforced and to detect any failure of the NSFs. Thus, a standard

Category		Issue
Standardization challenges		• Standard interfaces to manage the security rule sets of the NSFs • Standard message formats and protocols for monitoring the execution status of the NSFs • Standard method to describe the capabilities of the NSFs • Standard interfaces to import open-source-provided attack profiles into the NSFs
Research challenges	Usability and efficiency	• Automatic selection of the NSFs and rule generation to enforce the user requested security policies • Automatic update of the network configuration and security policies to deal with network attacks • Preventing and resolving conflicts between the security policy rules • Dynamic life cycle management of the NSFs and traffic load balancing
	Security	• Remote attestation to validate the authenticity of the NSFs of third-party vendors • Traffic authentication to prevent unauthorized modification • Fault tolerance of the security controller • Access control to prevent unauthorized access to the NSFs • Key management and distribution of the I2NSF system components

Table 1. Challenges for the I2NSF system.

protocol should be defined for such monitoring. Capability negotiation and discovery is also fundamental to the I2NSF system, where each NSF has different security capabilities. Therefore, a standard method to describe the capabilities of an NSF is required for such capability negotiation and discovery. For example, when triggering an advanced security action, the NSF should describe the security capability required for that advanced action to discover an NSF with that capability. In addition, most NSFs rely on the attack signatures and profiles for their operations; therefore, it is important to maintain the NSFs with extensive and up-to-date attack profiles to ensure their effectiveness. Open-source-provided databases of attack profiles (e.g., by Snort, Suricata, and Kismet) are useful in this regard. Thus, a standard is required for simplifying the process of importing open-source-provided attack profiles into the I2NSF system.

Research Challenges: Based on the standard interfaces discussed above, it is possible to automate various control and management tasks in the I2NSF system. Thus, further research of task automation is essential for the success of the I2NSF. In particular, the security controller should be able to automatically select a proper set of NSFs required to enforce a high-level security policy requested by the user in a cooperative manner, and also automatically generate low-level security policy rules for each of the selected NSFs [15]. Moreover, when the security controller investigates the security events and alerts reported by the NSFs, it should be able to intelligently identify unknown malicious activities through the investigation and to automatically update the network configuration and security rules to deal with the attacks in a timely manner. It is also crucial to prevent and resolve conflicts between a variety of security policy rules by both proactive and reactive approaches. Dynamic life cycle management of the NSF instances and traffic load balancing is also important for efficient and flexible resource utilization.

The security controller performs several critical roles in the I2NSF system, and a failure of the security controller can result in the service disruption of the entire system. Thus, preventing unexpected failures of the security controller is crucial. For example, application isolation to protect the security controller from buggy applications running inside is an important research issue in this regard. Dynamic key management and distribution is important to establish secure and authenticated interfaces between the system components. Secure binding between an NSF and its capability description is also important to ensure the trustworthiness of the capability description.

In cloud-based security service environments, a packet can be delivered to an NSF placed in the cloud system of an external security vendor. In this situation, if the external NSF is compromised by the attacker, the packet would be exposed to various types of attacks such as sniffing and forgery. To prevent these attacks, the user can periodically examine the integrity of the NSF instance through remote attestation. In addition, the user can sign the packet data before sending it to the external NSF to prevent unauthorized modification of the packet. Every access to the NSFs should be carefully authorized to prevent illegal changes in the critical configuration of the NSFs.

Conclusion

This article presents an I2NSF architecture that enables an efficient and flexible security service in cloud-based security service environments. The key contribution of the I2NSF architecture is standardizing the interfaces to the NSFs of different vendors to simplify the management of these NSFs. In addition, we seamlessly integrate the I2NSF architecture and SDN to optimize the process of security service enforcement by the I2NSF system in the cloud-based security service environments. We implement the I2NSF architecture integrated with SDN using various open source software to demonstrate its feasibility and effectiveness. We finally discuss the standardization and research challenges for the I2NSF architecture.

Acknowledgment

This work was supported by the Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korean government, Ministry of Science and ICT (MSIT) (Grant R-20160222-002755) and by the MSIT under the Information Technology Research Center (ITRC) support program (Grant IITP-2017-2017-0-01633) supervised by the IITP.

References

[1] R. Mijumbi *et al.*, "Network Function Virtualization: State-of-the-Art and Research Challenges," *IEEE Commun. Surveys & Tutorials*, vol. 18, no. 1, 2016, pp. 236–62.
[2] J. Sherry *et al.*, "Making Middleboxes Someone Else's Problem: Network Processing as a Cloud Service," *ACM SIGCOMM Computer Commun. Review*, vol. 42, no. 4, 2012, pp. 13–24.
[3] E. Messmer, "Gartner: Cloud-Based Security as a Service Set to Take Off," *Network World*, Oct. 2013; https://www.networkworld.com/article/2171424/data-breach/gart-

ner--cloud-based-security-as-a-service-set-to-take-off.html, accessed Oct. 14, 2017.
[4] IETF, "Interface to Network Security Functions (I2NSF) Working Group"; https://datatracker.ietf.org/wg/i2nsf/charter/, accessed Oct. 14, 2017.
[5] S. Hares *et al.*, "Interface to Network Security Functions (I2NSF): Problem Statement and Use Cases," IETF RFC 8192, July 2017.
[6] ONF, "Software-Defined Networking: The New Norm for Networks," ONF White Paper, vol. 2, 2012, pp. 2–6.
[7] S. Oh *et al.*, "A Flexible Architecture for Orchestrating Network Security Functions to Support High-Level Security Policies," *Proc. 11th ACM Int'l. Conf. Ubiquitous Info. Management and Commun.*, Jan. 2017, pp. 44–48.
[8] R. Enns *et al.*, "Network Configuration Protocol (NETCONF)," IETF RFC 6241, June 2011.
[9] J. M. Halpern and C. Pignataro, "Service Function Chaining (SFC) Architecture," IETF RFC 7665, Oct. 2015.
[10] M. Bjorklund, "YANG — A Data Modeling Language for the Network Configuration Protocol (NETCONF)," IETF RFC 6020, Oct. 2010.
[11] J. Kim *et al.*, "SDN-Based Security Services Using Interface to Network Security Functions," *Proc. 6th IEEE Int'l. Conf. Info. and Commun. Technology Convergence*, 2015, pp. 526–29.
[12] J. Collings and J. Liu, "An OpenFlow-Based Prototype of SDN-Oriented Stateful Hardware Firewalls," *Proc. 22nd IEEE Int'l. Conf. Network Protocols*, 2014, pp. 525–28.
[13] S. Kim *et al.*, "Preventing DNS Amplification Attacks Using the History of DNS Queries with SDN," *Proc. Euro. Symp. Research in Computer Security*, Springer, 2017, pp. 135–52.
[14] A. Bierman, M. Bjorklund, and K. Watsen, "RESTCONF Protocol," IETF RFC 8040, Jan. 2017.
[15] X. Liu, B. Holden, and D. Wu, "Automated Synthesis of Access Control Lists," *Proc. 3rd Int'l. Conf. Software Security and Assurance*, July 2017.

Biographies

Sangwon Hyun is a research fellow at Sungkyunkwan University, South Korea. He received his Ph.D. in computer science from North Carolina State University in 2011. Prior to his Ph.D. study, he received an M.S. in electrical engineering and computer science from Seoul National University, South Korea, in 2004, and a B.S. in electrical and computer engineering from Sungkyunkwan University in 2002. His research focuses on network and system security.

Jinyong (Tim) Kim is a graduate student in the Department of Computer Science and Engineering at Sungkyunkwan University. He received a B.S. degree from the Department of Computer Engineering at Kumoh National Institute of Technology in 2015. His research interests include SDN/NFV-based network security and drone battery charging scheduling.

Hyoungshick Kim received his B.S. degree from the Department of Information Engineering, Sungkyunkwan University, his M.S. degree from the Department of Computer Science, Korea Advanced Institute of Science and Technology, Daejeon, South Korea, and his Ph.D. degree from the Computer Laboratory, University of Cambridge, United Kingdom, in 1999, 2001, and 2012, respectively. He is currently an assistant professor with the Department of Software, Sungkyunkwan University. His research interests include usable security and security engineering.

Jaehoon (Paul) Jeong received his B.S. degree from the Department of Information Engineering, Sungkyunkwan University, in 1999, his M.S. degree from the School of Computer Science and Engineering, Seoul National University, South Korea, in 2001, and his Ph.D degree from the Department of Computer Science and Engineering, University of Minnesota, in 2009. He is currently an assistant professor with the Department of Software, Sungkyunkwan University. His research interests include the Internet of Things, vehicular networks, and SDN/NFV-based network security.

Susan Hares has over 30 years of experience in international standards organizations for Internet and IT technology (IETF, IEEE, BBF, MEF, and MAP/TOP). She chairs the following IETF Working Groups: IDR (BGP), I2RS, and TRILL. She has also founded companies and managed teams developing network hypervisors and secure routing/switching software suites. She is a Ph.D. student at Regent University Business and Leadership School in global leadership.

Linda Dunbar has an M.S. in computer science from the University of Maryland. She has been working in networking industry for over 20 years. Prior to Huawei she worked for Cisco, Fujitsu, and Nortel. Her recent work includes technologies for optimally and securely interconnecting enterprises' branch offices with dynamic workloads and applications. She is very actively in the networking and security industry, has published many IETF RFCs, and co-chairs the I2NSF WG.

Adrian Farrel is an innovator and standards-maker at the IETF where he has co-authored more than 60 RFCs and served as Routing Area Director for six years. With a background in communications software design and implementation, he now runs Old Dog Consulting, specializing in SDN, NFV, traffic engineering, and path computation. He works closely with Juniper Networks on the design of new protocols, and has written many books, including three volumes of fairy tales.