IBCS: Intent-Based Cloud Services for Security Applications

Jinyong (Tim) Kim, Eunsoo Kim, Jinhyuk Yang, Jaehoon (Paul) Jeong, Hyoungshick Kim, Sangwon Hyun, Hyunsik Yang, Jaewook Oh, Younghan Kim, Susan Hares, and Linda Dunbar

Abstract

As the number of network devices is increasing and they are highly connected, network attacks have become more complex and varied. To mitigate these attacks, multiple types of network security equipment are used in combination, requiring considerable security knowledge of each type of network security equipment. Also, the deployment of network security equipment requires considerable cost and time. To solve these problems, this article proposes an IBCS based on automation and virtualization. IBCS provides not only an automated, virtualized security system for security service providers, but also easy, quick, and flexible cloud-based security services for security service consumers based on their intents. Finally, this article demonstrates the feasibility of IBCS with a prototype in a real network environments.

Introduction

As Industry 4.0 is emerging, the number of network devices is increasing, and the devices are highly connected for various application domains such as cyber-physical systems (CPS) and the Internet of Things (IoT) [1].

However, as the smart industry has continued to develop, the network environment has became increasingly diverse, and the amount of network traffic has rapidly increased. This diversity among network environments has provided attackers with increasing chances of network attacks for confidentiality, integrity, and availability. To protect the confidentiality, integrity, and availability of the systems and data in the diverse network environment, security awareness will continue increasing in Industry 4.0, but many difficulties remain, as follows [2].

Dynamic Changes in Network Environments: Whenever new technologies develop, new devices deploy, or new attacks appear, it is difficult to deploy the network security equipment every time because it requires considerable cost and time [3].

Expertise of Network Security Equipment Developed by Various Vendors: The use of multiple types of network security equipment requires significant security knowledge of the various network security equipment. In general, managing such diverse equipment requires a large number of experts, which introduces a high cost of hiring and training them [4, 5].

To address these problems, we propose Intent-Based Cloud Services for Security Applications (IBCS) based on automation and virtualization. This IBCS is a security system that combines the interface to network security functions (I2NSF) framework [4] and the network functions virtualization (NFV) architectural framework [6]. An NSF is defined as either a physical middlebox or a virtualized server that performs network-based security services like firewall, deep packet inspection (DPI), and distributed denial of service (DDoS) attack mitigation. The I2NSF framework is a proposed framework for automating management of a network security function (NSF) in the Internet Engineering Task Force (IETF) I2NSF Working Group (WG). The NFV architectural framework is a proposed framework for automating management and orchestration of an NFV in the European Telecommunications Standards Institute (ETSI). IBCS simplifies the operation process and improves resource utilization for security services through automation and virtualization. The ultimate goal of IBCS is to provide an automated and virtualized security system to security service providers. It is also to provide easy, quick, and flexible cloud-based security services to security service consumers (e.g., network administrators and people who want security services) by offering intuitive ways of using the security services based on the intents of consumers. As an example, intent-based security services allow both the security service providers and the security service consumers to set up security policies without any expert-level security knowledge. These services can decrease configuration effort and time to enforce their required security policies to the relevant network security functions in the system. Figure 1 shows the architecture of our IBCS using the I2NSF framework on top of the NFV architecture.

IBCS-based security service automation consists of four steps: a consumer's intent specification, security policy translation, policy provisioning, and policy enforcement. First, for a customer's intent specification, the IBCS defines NSFs as a set of security capabilities (e.g., IPv4 address, IPv6 address, pass, drop, and log), and then registers the capabilities of NSFs with a secu-

The authors propose an IBCS based on automation and virtualization. IBCS provides not only an automated, virtualized security system for security service providers, but also easy, quick, and flexible cloud-based security services for security service consumers based on their intents. The authors demonstrate the feasibility of IBCS with a prototype in real network environments.

Jinyong (Tim) Kim, Eunsoo Kim, Jinhyuk Yang, Jaehoon (Paul) Jeong (corresponding author), and Hyoungshick Kim are with Sungkyunkwan University; Sangwon Hyun is with Myongji University; Hyunsik Yang, Jaewook Oh, and Younghan Kim are with Soongsil University; Susan Hares is with Hickory Hill Consulting; Linda Dunbar is with Futurewei Technologies.

Digital Object Identifier: 10.1109/MCOM.001.1900476

0163-6804/20/$25.00 © 2020 IEEE

IBCS provides security service providers with an automated and virtualized infrastructure for NSF provisioning. Using the automated and virtualized security system, they can quickly provide security service consumers with the appropriate security services and manage the resources of the cloud servers both efficiently and flexibly without the provider needing to install security equipment.

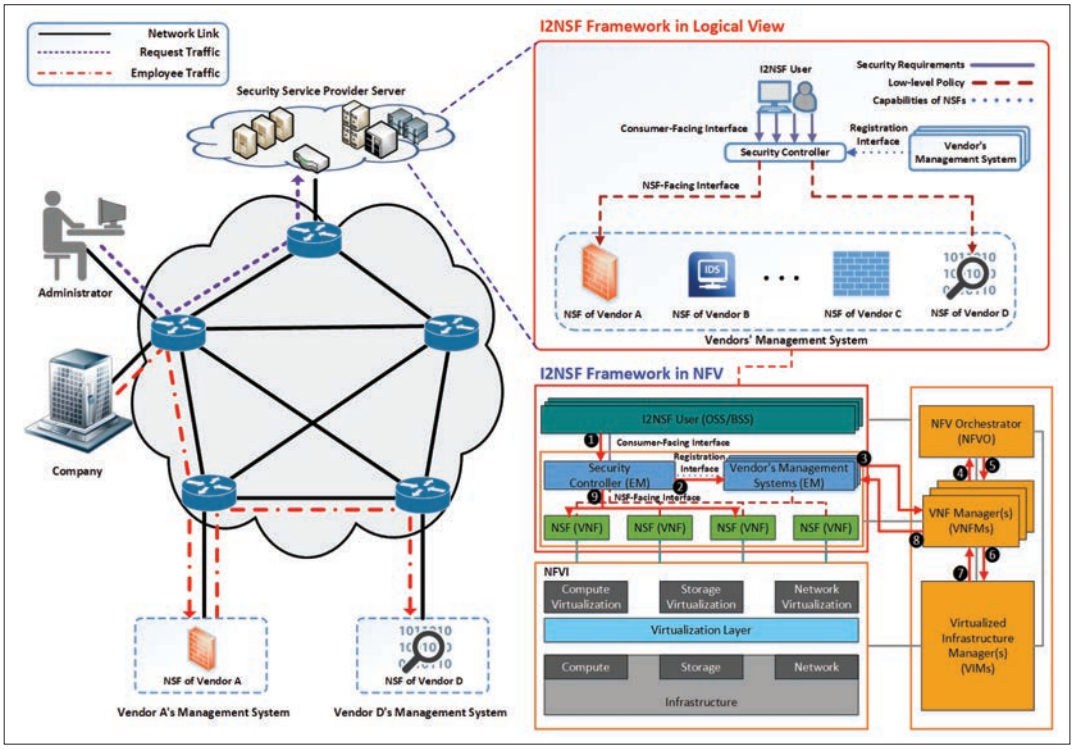

Figure 1. Intent-based cloud security system architecture.

rity controller in a centralized manner. With this setup, based on the registered capabilities, IBCS can provide a security service consumer with an automated security service through the consumer's intent specification as a high-level security policy. That is, the security service consumer can ask for a security service simply by delivering its intent as a high-level security policy to the security controller without any expertise in network security or the underlying NSFs. Second, for security policy translation, the security controller translates the requested intent of the high-level security policy (e.g., block malicious IP addresses) into a low-level security policy (e.g., drop IP packets from/to IP addresses 10.0.0.1 and 10.0.0.2) as a configuration command of an open source software for network threat detection (e.g., Suricata), which can be understood by NSFs. Third, for policy provisioning, the IBCS searches for the appropriate NSFs that can enforce the policy by matching the policy with the set of security capabilities of the NSFs. As a result, one or more NSFs for policy provisioning are possibly chosen according to the security capabilities of the NSFs registered with IBCS. If the chosen NSF is not activated, the NSF can be instantiated using the NFV architectural framework. Then the security controller delivers the translated security policy to the instantiated NSF to enforce the policy. Last, for policy enforcement, the IBCS provides the security service consumer with security services through the instantiated NSFs.

Note that to the best of our knowledge, IBCS is the first attempt to provide security service providers with an automated and virtualized security system, and also security service consumers with easy, quick, and flexible security services based on their intents without requiring any knowledge of security and the installation of any security equipment. Our previous work on I2NSF [5] explains the framework and standard interfaces of I2NSF for our IBCS along with several standardization and research challenges for I2NSF. On the other hand, this article makes the following contributions.

A Security Service through an Automated and Virtualized Cloud-Based Security System: IBCS provides security service providers with an automated and virtualized infrastructure for NSF provisioning. Using the automated and virtualized security system, they can quickly provide security service consumers with the appropriate security services and manage the resources of the cloud servers (i.e., NSFs) both efficiently and flexibly without the provider needing to install security equipment.

A Security Service Consumer's Intent-Based Security Service: The intent-based security service feature of IBCS provides security service consumers with a user-friendly way to request security services based on their intents without requiring any security expertise. They can easily request their intents, thus reducing the possibility of errors and allowing consumers to receive security services quickly through an I2NSF user (i.e., software or web).

The Combination of Standard Technology from IETF and ETSI: IBCS combines the I2NSF framework in IETF and the NFV architectural framework in ETSI to provide the automation and virtualization of security systems. This system demonstrates the potential synergy by combining standards from different international standards organizations.

The Implementation and Evaluation of the Proposed Security System for Feasibility: In order to show the feasibility of the proposed security system, we have implemented the system in a real network environment using the open source

cloud computing platform OpenStack. The performance of IBCS is evaluated against the conventional framework for security services through simulation.

Intent-Based Cloud Security System Architecture

This section presents the goal, components, and interfaces of IBCS as well as the corresponding security service procedure. The goal of IBCS is to provide security service providers with an automated and virtualized security system as well as to provide security service consumers with easy, quick, and flexible security services based on the specific intents of the security service consumers without requiring any knowledge of security and the installation of security equipment. Figure 1 shows an example of IBCS to block the social network service (SNS) access of employees during business hours. First of all, vendors (i.e., security solution companies) register the security capabilities of NSFs to the security controller through the vendor's management systems (VMSs). In Fig. 1, the employees are those who are under the security policies for the company. As shown in Fig. 1, in order to block the SNS access of employees in a company, the administrator (e.g., security service consumer) of the company requests security services by issuing a high-level description of the intents for SNS traffic filtering to the security service provider through the I2NSF user. This high-level description of the intent is called a high-level security policy. Upon delivery of the intent to the security service provider server, the security controller converts the high-level security policy into the corresponding low-level security policy. Note that a low-level policy is the detailed policy meant to configure the NSFs to perform the high-level security policy corresponding to the required security service. Then the security controller asks the NFV architectural framework to dynamically generate the appropriate network security functions (i.e., NSFs of vendor A and vendor D) to perform the low-level security policy in the NFV architectural framework [6]. Next, the security controller creates low-level security policies according to the generated NSFs (i.e., NFVs), and delivers the policies to each NSF. Finally, the NSFs configure the received low-level security policies in their own respective system. The generated NSFs can be maintained by vendors through the VMSs. A detailed explanation of the security service procedure in IBCS is provided below. If an employee attempts to access SNS, his/her traffic will be forwarded to the NSFs of vendor A's management system and vendor D's management system, and then the traffic will be dropped according to the security policy of the administrator. In other words, by steering traffic into the NSFs provided by the vendors, the administrator can be provided with automated and virtualized security services. We will describe the components and interfaces of IBCS used for automation and virtualization in the next subsection.

Components

In this subsection, we describe the main components of IBCS, which combines the I2NSF framework [4] in the IETF and NFV architectural framework [6] in ETSI in order to provide the automation and virtualization of security systems. The I2NSF framework consists of the following main components: I2NSF user, vendor's management systems, security controller, and network security functions. This I2NSF framework is implemented on top of the ETSI NFV architectural framework [6].

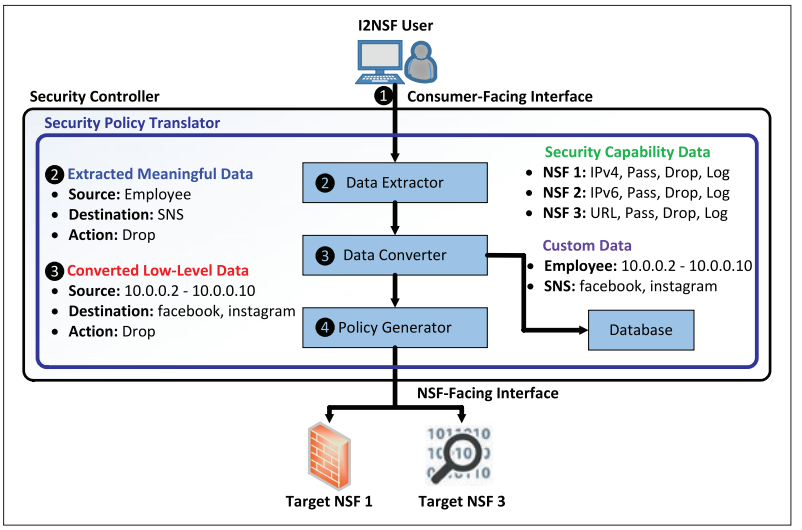

Figure 2. Structure of a security policy translator.

I2NSF User: An I2NSF user is a software or web page implemented by a security service provider or provided from vendors in the IBCS. For normalized data transfer to the security controller, the I2NSF user is implemented according to the consumer-facing interface [7]. Through the I2NSF user, security service consumers can easily request their intents to the security service providers without needing any security expertise or consideration of the underlying NSFs. The intents generated by the I2NSF user are then delivered to the security controller via the consumer-facing interface.

Vendor's Management Systems: VMSs are the systems to which vendors (i.e., security solution companies) can register their NSFs and the security capabilities of the NSFs with the security controller. For normalized data transfer to the security controller, the VMSs are implemented according to the registration interface [8, 9]. In addition, through the VMSs, NSFs are created and deleted without any intervention of vendors.

Security Controller: The security controller is the most critical component in the proposed security system. The security controller is responsible for three major roles.

Security Capability Management of NSFs: The security controller centrally manages the NSFs, and their security capabilities are registered with the security controller by VMSs via the registration interface. The security controller is also capable of monitoring the NSFs running in the system and maintains various information related to each NSF (e.g., network access information and workload status).

Translation from an Intent to a Low-Level Security Policy: The security controller receives an intent from a security service consumer through the I2NSF user. Upon delivery of the intent, the security controller translates the intent into a

low-level policy. The detailed procedure of the security policy translation is explained below. The security policy translation consists of three steps: data extraction, data conversion, and policy generation. Figure 2 shows an example of the translation process of an SNS web-filtering policy.

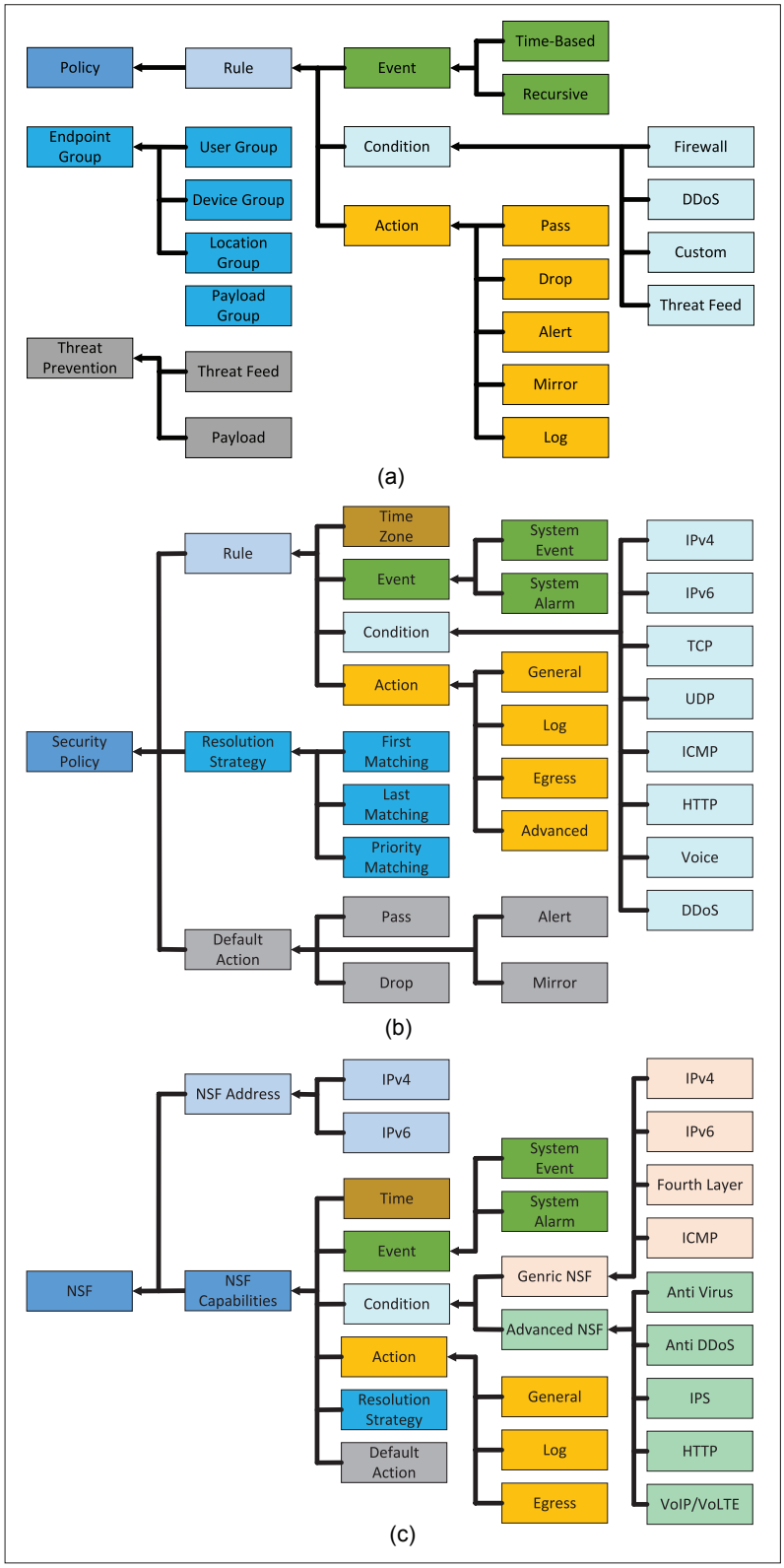

Figure 3. A high-level abstraction of interfaces in IBCS: a) a high-level abstraction of a consumer-facing interface; b) a high-level abstraction of an NSF-facing interface; c) a high-level abstraction of a registration interface.

Security Policy Provisioning: With the translated low-level policies, the security controller seeks appropriate NSFs that can satisfy the intents of the security service consumers. After identifying these appropriate NSFs, the security controller sends them the low-level policies via the NSF-facing interface [10].

Network Security Functions (NSFs): An NSF is software operated in either a physical middlebox or a virtualized server that performs network security services according to the security policies. Some examples of NSFs include firewall, DPI, and DDoS attack mitigation. The NSFs are provided by the vendors (i.e., security solution companies) through the VMS.

Interfaces

This subsection describes the three main standard interfaces of our proposed security system for the automated security services. For the automation of IBCS, we standardized three interfaces — the registration interface [8, 9], the consumer-facing interface [7], and the NSF-facing interface [10] —in the IETF I2NSF WG. Figure 3 shows a high-level abstraction of the three standard interfaces in IBCS. As shown in Fig. 3, the three standard interfaces are modeled using an Event-Condition-Action (ECA) policy model [7, 8, 10] and a YANG data model language [11] for a generic standard data model of a vendor-neutral syntax policy in order to comprehensively express a security policy for NSFs developed by different security vendors. This ECA policy model consists of three Boolean clauses, an event clause, a condition clause, and an action clause, which are all logical statements that are evaluated as either true or false.

Registration Interface: The registration interface is the interface between the security controller and the VMSs [8, 9]. The main purpose of the registration interface is to provide a standard interface to a security service provider so that the NSFs developed by the vendors can be easily adapted to IBCS. Through this registration interface, vendors can register their NSFs and their corresponding security capabilities with the security controller. For communication, the interface is implemented according to the NETCONF protocol [12].

Consumer-Facing Interface: The consumer-facing interface is the interface between the I2NSF user and the security controller [7]. The main purpose of the consumer-facing interface is to provide a standard interface to I2NSF user developers (e.g., security service providers or vendors) so that the I2NSF user can easily adapt to IBCS. Using the I2NSF user developed according to the consumer-facing interface, security service consumers can send their intents to the security controller without considering the underlying NSFs and without any security knowledge. For communication, the interface is implemented using the RESTCONF protocol [13].

NSF-Facing Interface: The NSF-facing interface is the interface between the security controller and NSFs (e.g., firewall, IDS, IPS, anti-virus, or DDoS mitigation) [10]. The primary purpose of the NSF-facing interface is to provide a standard interface to vendors so that the NSFs developed by the vendors can easily adapt to IBCS.

Through this NSF-facing interface, the security controller can control and monitor aspects of the heterogeneous NSFs developed by various vendors. For the communication, the interface is implemented according to the NETCONF protocol [12].

The Security Service Procedure in an Intent-Based Cloud Security System

This section explains the security service procedure of the proposed security system. The I2NSF framework in NFV in Fig. 1 shows the security service procedure of the intent-based cloud security system. First, in order to provide a security service using IBCS, vendors should register the security capabilities of their NSFs with the security controller. Virtual machine (VM) images of the NSFs also need to be registered with the virtual network function manager (VNFM) [6] in their own respective cloud servers. During this process, the VMSs manage the respective mapping table for the VM image ID corresponding to the registered NSF. The detailed security service procedure after registering the NSFs is as follows:

- A security service customer delivers their intents through the I2NSF user. At this time, the service customer can easily deliver such intents requests through the I2NSF user without any knowledge of network security or consideration of the underlying NSFs.
- With the delivered intent, the security controller translates the intent into a low-level security policy that can be understood by NSFs. Then the security controller finds the appropriate NSF that can enforce the security policy by matching the translated security policy and the set of the security capabilities of the registered NSFs. If the found NSF is a middlebox (i.e., physical device) or a virtual NSF that has already been instantiated, the security controller delivers the translated low-level security policy directly to the NSF. Otherwise, if the required NSF is not a middlebox and is a virtual NSF that has not been instantiated, the security controller requests that the VMS instantiate the NSF. A detailed explanation of the security policy translation and policy provisioning is described below.
- Upon receiving the instantiation request from the security controller, the VMS searches the mapping table for the VM image ID corresponding to the requested NSF, then sends the VM image ID to the VNFM to instantiate the NSF.
- Upon receiving the instantiation request, the VNFM first asks the NFV orchestrator for permission to allocate resources to the NSF instance.
- The NFV orchestrator analyzes the request in terms of the allocation of virtual resources. If the resources are available, the NFV orchestrator grants permission to the VNFM.
- After obtaining permission, the VNFM asks the VIM to allocate resources for the NSF instance.
- The VIM creates the NSF according to the information of the requested NSF instance based on the allocated resources, and notifies the VNFM of the information of the created NSF (e.g., IP address and port number).
- The VNFM notifies the VMS of the information regarding the created NSF, and the VMS delivers the information to the security controller.
- Finally, when the security controller receives the information of the created NSF, it delivers the low-level security policy to the NSF for policy enforcement, and the NSF configures the received policy on its own system.

Through these procedures, the system can provide security service consumers with quick and flexible security services based on their high-level intent without requiring any sophisticated knowledge of security and any intervention from the security service provider.

Security Policy Translation

This section explains the process of security policy translation from an intent to a low-level security policy. Figure 2 shows the structure of a security policy translator to translate the intents of security service consumers into low-level security policies that can be executed in NSFs. The security policy translator consists of three components: a data extractor, a data converter, and a policy generator. The data extractor is a component for extracting data from an intent using deterministic finite automata (DFA) [14]. The data converter is a component for converting the extracted intent to a low-level security policy and searching for proper target NSFs that can support the converted low-level policy. The policy generator is a component for generating an xml document with the converted low-level security policy using context-free grammar (CFG) [14].

The security policy translation procedure is shown in Fig. 2 as follows:

- The I2NSF user for a consumer delivers a low-level security policy to the security controller over the consumer-facing interface (e.g., SNS web-filtering: "drop the SNS traffic of employees during business time").
- The data extractor of the security policy translator in the security controller extracts meaningful data from the requested intent using DFA (e.g., Source: Employees, Destination: SNS Websites, and Action: Drop).
- The data converter converts the extracted data into low-level data based on a database, then searches for proper target NSF(s) based on the registered security capabilities of NSFs (e.g., Source (Employees): 10.0.0.2 ~ 10.0.0.10, Destination (SNS Websites): facebook and instagram, and Action: Drop).
- Finally, the policy generator generates a low-level security policy based on the obtained NSFs using CFG. Then the security controller delivers the low-level security policy to the selected target NSF(s) over the NSF-facing interface.

Note that our previous work on security policy translation [15] includes the detailed explanation of the security policy translation process. More use cases are explained such as time-based web filter and deep packet inspection (e.g., VoIP/VoLTE security service) in our IETF drafts (i.e., the consumer-facing interface [7] and the NSF-facing interface [10]).

The main purpose of the consumer-facing interface is to provide a standard interface to I2NSF user developers so that the I2NSF user can be easily adapted to IBCS. Using the I2NSF user developed according to the consumer-facing interface, security service consumers can send their intents to the security controller without considering the underlying NSFs and without any security knowledge.

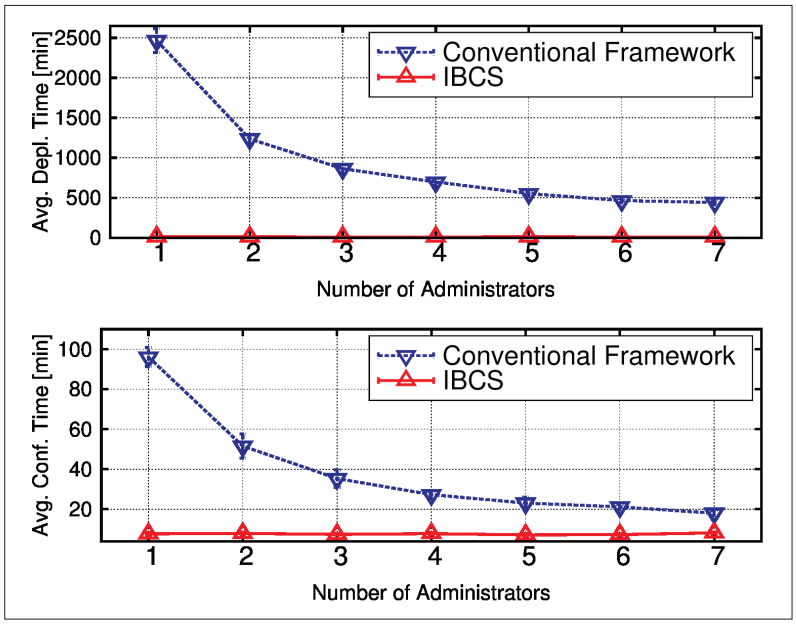

Figure 4. Conventional framework vs. IBCS in performance evaluation.

Implementation

This section describes our implementation of IBCS in order to demonstrate the feasibility of IBCS. We implemented our system under the assumption of an example scenario that controls the SNS access of employees during business time in a company environment. For the IBCS implementation, we used several different types of open source software, such as OpenStack, Tacker, Apache2, PHP, MySQL, Python, Suricata, Jetconf, ConfD, Docker, and Kubernetes. OpenStack and Tacker were used to provide NSFs as virtual network functions (VNFs) and manage them dynamically according to the security service customers' specific intents. We implemented the I2NSF user using Apache2, PHP, and MySQL to provide a web-based interface through which security service consumers could make service requests based on their intents. The security controller was implemented using Python to translate the service consumers' intents into low-level security policies to be sent to the underlying NSFs. The security controller also manages NSFs and their security capabilities. For security services, we implemented NSFs (e.g., time-based firewall and url filter) using Suricata (as open source software for IDS and IPS).

The consumer-facing interface is implemented using Jetconf, which is an implementation of the RESTCONF protocol using Python 3. The NSF-facing interface and the registration interface are implemented using ConfD, which is a powerful software framework developed by Tail-f and Cisco for the NETCONF-based management of network elements. We also implemented the YANG data model for each of the three interfaces (i.e., the consumer-facing interface, NSF-facing interface, and registration interface) according to the specifications being developed by the I2NSF WG [7–10]. Note that our implementation is based on a set of core functionalities implemented with the Docker and Kubernetes.

Note that the source codes and documents of our implementation are available at https://github.com/kimjinyong/i2nsf-framework, and a video demo is also available at https://www.youtube.com/watch?v=WYsPUd1BZXA.

Performance Evaluation

This section shows the performance evaluation of the IBCS against the conventional framework, which does not use automation and virtualization. The evaluation is performed through simulation, where Python is used, with the following performance metrics and parameters:

- **Performance metrics:** Average deployment time and configuration time are used as performance metrics.
- **Parameters:** The impact of the number of administrators for NSF deployment and security policy configuration on the metrics is investigated. The normal distribution $N(\mu, \sigma)$ for the deployment time and configuration time is used for simulation. The time unit is minute for both the deployment time and configuration time. The deployment time using the conventional framework is $N(120, 60)$, and the deployment time using IBCS is $N(6, 2)$. The configuration time using the conventional framework is $N(5, 3)$, and the configuration time using IBCS is $N(4, 2)$. Note that the time setting is based on system operation experience.

Figure 4 shows the deployment time and configuration time according to the number of administrators using the conventional framework and IBCS, respectively. As predicted, the average deployment time and configuration time using the conventional framework decrease as the number of administrators increases. On the other hand, IBCS has much lower average deployment and configuration time than the conventional framework regardless of the number of administrators through automation and virtualization, as shown in Fig. 4. Also, IBCS can save a lot of time and cost in hiring and training administrators for security services.

Research Challenges

This section discusses some research challenges for IBCS to be deployed into the industry.

Natural Language Processing: Security service consumers can only request their intents through an I2NSF user using the consumer-facing interface [7]. For a more flexible intent expression, it is necessary to support a natural-language-like intent specification and voice recognition. These features require natural language processing (NLP) to translate a voice intent into the corresponding security policy for the consumer-facing interface.

Self-Adaptive Policy Provisioning: During the policy provisioning, a new kind of security attack, including virus, spam, DDoS attack, and data loss prevention, is detected by an NSF, and the traffic flow relevant to such an attack can be blocked by a firewall or an SDN switch for a more efficient counterattack. The configuration of a new firewall filtering rule can be initiated by the detecting NSF, which reports such an attack to the security controller. The security controller can configure the firewalls under its control through the NSF-facing interface [10] without the intervention of an I2NSF user.

Packet Path Verification: We need a procedure to check whether a packet has reached an NSF along the intended sequence of NSFs to avoid DDoS attacks related service function chaining (SFC). A security policy can have a pipeline of security services (e.g., firewall, DPI, and DDoS attack mitigation) for sequential packet processing. These pipelined security services can be provisioned by an SFC. A hacker can copy valid in-flight packets passing a service path in the SFC, and later forward the copied packets to the NSFs that will process the redundant packets. To prevent this replay attack as a DDoS attack, a packet with a service path should be verified to check whether it is valid or fake.

Conclusion

As an intent-based cloud security system, IBCS provides security service providers with an automated and virtualized security system using I2NSF and NFV along with a security policy translator. This article demonstrates the feasibility of the IBCS-based security services by implementing a prototype using only open source software. As future work, we will enhance IBCS for self-evolving against new security attacks over time.

Acknowledgment

This work was supported by the IITP Grant funded by the Ministry of Science and ICT, Korea (no. 2016-0-00078, Cloud Based Security Intelligence Technology Development for the Customized Security Service Provisioning).

References

[1] Statista Research Dept., "Internet of Things (IoT) Connected Devices Installed Base Worldwide from 2015 to 2025 (in billions)"; https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/, accessed Apr. 10, 2020.

[2] Grand View Research, Cyber Security Market Size, Share & Trends Analysis Report By Component, By Security Type, By Solution, By Service, By Deployment, By Organization, By Application, And Segment Forecasts, 2019–2025; https://www.grandviewresearch.com/industry-analysis/cyber-security-market, accessed Apr. 10, 2020.

[3] Juniper Networks, "Transforming Service Life Cycle Through Automation With SDN and NFV"; https://www.juniper.net/assets/us/en/local/pdf/whitepapers/2000609-en.pdf, accessed Apr. 10, 2020.

[4] D. Lopez *et al.*, "Framework for Interface to Network Security Functions," IETF RFC 8329, Feb. 2018; https://tools.ietf.org/html/rfc8329.

[5] S. Hyun *et al.*, "Interface to Network Security Functions for Cloud-Based Security Services," *IEEE Commun. Mag.*, vol. 56, no. 1, Jan. 2018, pp. 171–78.

[6] ETSI Industry Specification Group, "Network Functions Virtualisation (NFV); Architectural Framework," ETSI GS NFV 002 V1.2.1, Dec. 2014; https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV%20002v1.2.1%20-%20GS%20-%20NFV%20Architectural%20Framework.pdf.

[7] J. P. Jeong *et al.*, "I2NSF Consumer-Facing Interface YANG Data Model," IETF Internet-Draft draft-ietf-i2nsf-consumer-facing-interfacedm-08, Mar. 2020, work in progress; https://datatracker.ietf.org/doc/draft-ietf-i2nsf-consumer-facing-interface-dm/.

[8] S. Hares *et al.*, "I2NSF Capability YANG Data Model," IETF Internet-Draft draft-ietf-i2nsf-capabilitydata-model-05, July 2019, work in progress; https://datatracker.ietf.org/doc/draft-ietf-i2nsf-capability-data-model/.

[9] S. Hyun *et al.*, "I2NSF Registration Interface YANG Data Model," IETF Internet-Draft draft-ietf-i2nsf-registration-interface-dm-08, Mar. 2020, work in progress; https://datatracker.ietf.org/doc/draft-ietf-i2nsf-registration-interface-dm/.

[10] J. T. Kim *et al.*, "I2NSF Network Security Function-Facing Interface YANG Data Model," IETF Internet-Draft draft-ietf-i2nsf-nsf-facinginterface- dm-08, Nov. 2019, work in progress; https://datatracker.ietf.org/doc/draft-ietf-i2nsf-nsf-facing-interface-dm/.

[11] M. Bjorklund, "The YANG 1.1 Data Modeling Language," IETF RFC 7950, Aug. 2016; https://tools.ietf.org/html/rfc7950.

[12] R. Enns *et al.*, "Network Configuration Protocol (NETCONF)," IETF RFC 6241, June 2011; https://tools.ietf.org/html/rfc6241.

[13] A. Bierman, M. Bjorklund, and K. Watsen, "RESTCONF Protocol," IETF RFC 8040, Jan. 2017; https://tools.ietf.org/html/rfc8040.

[14] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, 3rd Ed., Pearson, July 2006; https://www.amazon.com/Introduction-Automata-Theory-Languages-Computation/dp/0321455363.

[15] J. Yang and J. Jeong, "An Automata-Based Security Policy Translation for Network Security Functions," *Proc. 9th Int'l. Conf. ICT Convergence*, Oct. 2018, pp. 268–72.

Biographies

Jinyong (Tim) Kim is a Ph.D student in the Department of Computer Engineering at Sungkyunkwan University, Korea.

Eunsoo Kim is a Ph.D student in the Department of Computer Engineering at Sungkyunkwan University.

Jinhyuk Yang is an M.S. student in the Department of Computer Engineering at Sungkyunkwan University.

Jaehoon (Paul) Jeong is an associate professor in the Department of Computer Science and Engineering at Sungkyunkwan University.

Hyoungshick Kim is an associate professor in the Department of Computer Science and Engineering at Sungkyunkwan University.

Sangwon Hyun is an assistant professor in the Department of Computer Engineering at Myongji University, Korea.

Hyunshik Yang is a Ph.D student in the Department of Electronic Engineering at Soongsil University, Korea.

Jaewook Oh is an M.S. student in the Department of Electronic Engineering at Soongsil University.

Younghan Kim is a professor in the Department of Electronic Engineering at Soongsil University.

Susan Hares is a consultant with Hickory Hill Consulting in the United States and IETF I2RS Working Group Chair.

Linda Dunbar is a distinguished engineer at Futurewei Technologies in the United States and IETF I2NSF Working Group Chair.

This article demonstrated the feasibility of the IBCS-based security services by implementing a prototype using only open-source software. As future work, we will enhance the IBCS for self-evolving against new security attacks over time.