CBSS: Cloud-Based Security System with Interface to Network Security Functions

Jaehoon (Paul) Jeong
Department of Computer Science & Engineering
Sungkyunkwan University
Suwon, Republic of Korea
Email: pauljeong@skku.edu

Patrick Lingga
Department of Electrical & Computer Engineering
Sungkyunkwan University
Suwon, Republic of Korea
Email: patricklink@skku.edu

***Abstract*—This paper proposes a Cloud-Based Security System (CBSS) with Interface to Network Security Functions (I2NSF) as the framework and interfaces. It shows the feasibility of CBSS for flexible and efficient security services in cloud-based network environments such as 5G networks and Internet of Things (IoT) networks. The design and implementation of CBSS are explained along with information and data models of the I2NSF standard interfaces. The architecture of the I2NSF framework is augmented for Intent-Based Networking (IBN) for intelligent security services. Through experiment, it is shown that CBSS can handle various security attacks autonomously.**

***Index Terms*—I2NSF, Interface, Network Security Function, Cloud-Based Security System, Intent-Based Networking.**

I. Introduction

Nowadays, cloud computing has been a dominant computing platform in various domains such as Artificial Intelligence (AI) services, Internet of Things (IoT), and electronic commerce business systems. Especially, 5G cellular networks are leveraging the cloud computing and Multi-access Edge Computing (MEC) for their core networks and Radio Access Networks (RAN) [1]. The various domains adopt Software-Defined Networking (SDN) for supporting softwarized forwarding fabric and Network Functions Virtualization (NFV) for cloud computing and MEC [2]. For safe and secure network services, the cloud computing and MEC systems need to provide security services to various computers and devices such as hosts, servers, smart devices (e.g., smartphone and tablet), and IoT devices.

Many security vendors provide users with security solutions running in the cloud systems. However, there is no standard framework and interfaces for the multi-vendor security solutions in a cloud environment. As a result, it is hard to control and manage those solutions in a cloud system like a data center network and a campus network in a unified and efficient way. That is, each vendor needs its own interface to control and monitor its security function, which works as either a Physical Network Function (PNF) in a middle box or a Virtual Network Function (VNF) in a cloud system.

For cloud-based security service systems with a unified framework and standard interfaces to those security solutions, Interface to Network Security Functions (I2NSF) was proposed by I2NSF Working Group (WG) [3] in Internet Engineering Task Force (IETF) in 2017. I2NSF can accommodate heterogeneous security solutions from multiple security vendors as Network Security Functions (NSF) through the standardized I2NSF framework and interfaces [2], [4]. The I2NSF interfaces can be designed and implemented as a data-driven security approach with the data modeling based on YANG [5] and the remote control and management protocols such as NETCONF [6] or RESTCONF [7]. Thus, with the standardized framework and interfaces, those multi-vendor solutions can work together as either PNFs or VNFs under the control of a unified dashboard in a cloud computing system or MEC system [8].

This paper proposes a Cloud-Based Security System (CBSS) leveraging the I2NSF framework and interfaces for intelligent cloud security services. It explains the design and implementation of an I2NSF-based CBSS along with the information and data models of the I2NSF interfaces. Also, it explains the intelligent security system leveraging the concept of Intent-Based Networking (IBN) [9] with security policy translator and closed-loop security control in an I2NSF system for CBSS. First, the security policy translator can translate an intent of a network administrator (i.e., a high-level security policy) into the corresponding low-level security policy that can be configured into the NSFs for security services. Second, the closed-loop security control can detect a new security attack in a target network under the I2NSF framework, generate a new security policy and apply the policy to the I2NSF framework without any intervention of a network administrator. Also, if it detects the overloading and hardware problems of NSFs, the closed-loop security control can take action on them autonomously. Thus, this security management automation of CBSS may reduce the operational expenditure for security service management in complex cloud systems.

The remainder of this paper is organized as follows. Section II describes the motivation, architecture, and use case of CBSS. Section III describes the I2NSF framework and interfaces. Section IV specifies the information and data models of I2NSF interfaces. Section V articulates the security polity translator and closed-loop security control as IBN features. Section VI shows the performance of the I2NSF framework for the defense against a security attack. Lastly, Section VII concludes this paper along with future work.

978-4-907626-52-5/23/$31.00 ©2023 IPSJ

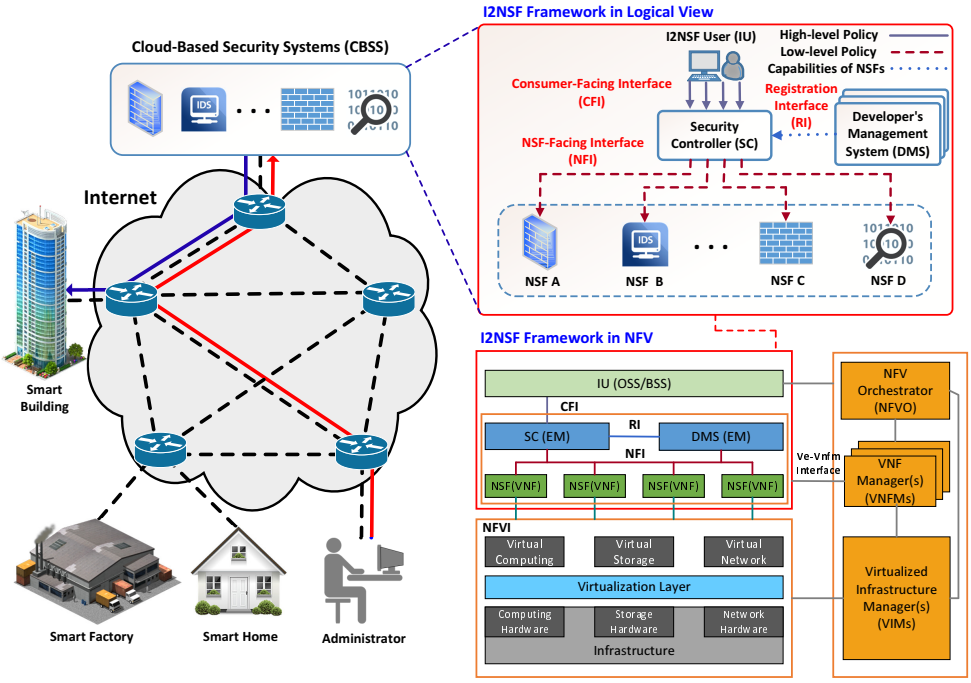


Fig. 1. Cloud-Based Security System (CBSS)

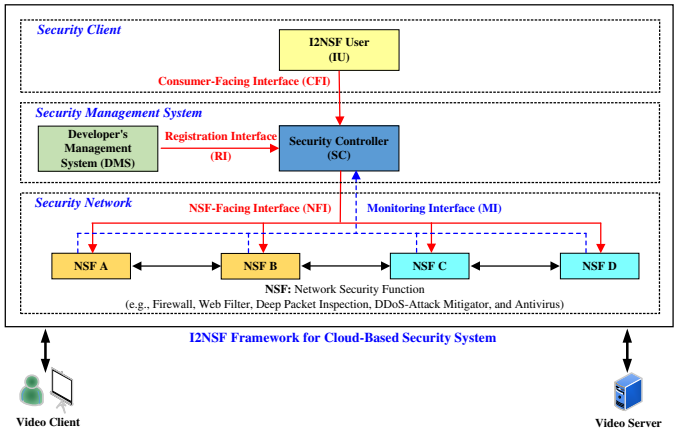


Fig. 2. Legacy Interface to Network Security Functions (I2NSF) Framework

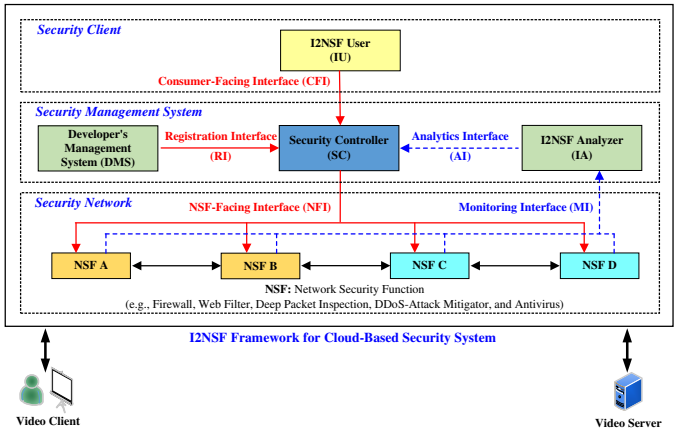


Fig. 3. Extended Interface to Network Security Functions (I2NSF) Framework

II. Cloud-Based Security System (CBSS)

This section describes the Framework of Interface to Network Security Functions (I2NSF). I2NSF is an IETF standard technology for cloud-based security service systems. I2NSF provides network users with a standard framework and standard interfaces for network security services. Fig. 1 shows a Cloud-Based Security System (CBSS) using I2NSF for such cloud-based security services. As shown in this paper, an administrator can configure a security policy for a site (e.g., smart building, smart factory, and smart home) remotely for the sake of the users in the site.

As a use case with Fig. 1, in a smart building, an employer has a high-level security policy that his employees cannot access Social Networking Service (SNS) websites like Facebook, Instagram, and YouTube during work time (from 9am to 6pm). To enforce this high-level security policy in the traffic in the smart building, the employer asks his network administrator to configure this security policy in Network Security Functions (NSF) in a Cloud-Based Security System (CBSS) through which the traffic in the smart building move. It is assumed that the Internet traffic of computers and mobile devices (e.g., smartphones and tablets) are steered to the Cloud-Based Security System (called CBSS) for traffic regulation and security services according to the company's security policy.

The enforcement of a high-level security policy can be realized in CBSS that is empowered with the I2NSF framework. As shown in the right-hand side of Fig. 1, I2NSF Framework is described in a logical view. A network administrator uses a web-based dashboard called an I2NSF User (IU) to make a high-level security policy. The I2NSF User delivers the high-level security policy to Security Controller (SC) which is a main control component in the I2NSF framework and handles

security demands from I2NSF Users. The Security Controller translates the high-level security policy into the corresponding low-level security policy that can be understood by NSFs. The Security Controller selects an appropriate NSF(s) among a pool of NSFs to process the low-level security policy, and then sends the policy to the selected NSF(s) as shown in the figure. After receiving the low-level security policy, the corresponding NSF(s) performs the requested security service for the sake of the I2NSF User.

The I2NSF framework can be built in a cloud using Network Functions Virtualization (NFV) along with Software-Defined Networking (SDN), as shown in the right-hand side of Fig. 1 [8]. NFV is the standard framework from European Telecommunications Standards Institute (ETSI) The security network having NSFs can be constructed by SDN.

III. I2NSF Framework

This section explains the components and interfaces in the I2NSF framework [4], [10]. Fig. 2 shows the legacy I2NSF framework which has been completed by the IETF I2NSF WG. But, for Intent-Based Networking (IBN) [9], as shown in Fig. 3, an extended I2NSF framework is proposed by this paper. Note that the extended I2NSF framework is mentioned as the I2NSF framework in this paper. Table I shows the description of components of the extended I2NSF framework. This I2NSF framework has five components such as Security Controller (SC), I2NSF User (IU), Developer's Management System (DMS), Network Security Function (NSF), and I2NSF Analyzer (IA).

Table II shows the description of interfaces of the I2NSF framework. This I2NSF framework has five interfaces such as Registration Interface (RI) [11], Consumer-Facing Interface (CFI) [12], NSF-Facing Interface (NFI) [13], Monitoring Interface (MI) [14], and Analytics Interface (AI) [15].

As shown in Figs. 2 and 3, the I2NSF framework consists of three layers such as Security Network, Security Management System, and Security Client. Security Network consists of multiple NSFs and forwarding elements (e.g., switch and router). It performs data forwarding through NSFs' security services in data networks (e.g., Internet and data center). Security Management consists of Security Controller (SC), multiple Developer's Management Systems (DMS), and I2NSF Analyzer (IA). It performs control and monitoring functions for security services for network traffic passing through the Security Network. Security Client is I2NSF User which makes a high-level security policy based on a network administrator's security demands.

In the legacy I2NSF framework in Fig. 2, Security Controller collects NSF monitoring data from NSFs as an NSF data collector via Monitoring Interface and can analyze the data [14]. On the other hand, our extended I2NSF framework in Fig. 3, I2NSF Analyzer collects NSF monitoring data from NSFs as an NSF data collector via Monitoring Interface and analyzes the data for the sake of Security Controller. This separation of Security Controller and I2NSF Analyzer can release the workload of Security Controller as an NSF data analytics function.

TABLE I
DESCRIPTION OF COMPONENTS IN I2NSF FRAMEWORK

Component	Description
Security Controller (SC)	An orchestrator to govern the I2NSF system by translating a security service demand into a configurable, low-level security policy.
I2NSF User (IU)	A dashboard to construct a high-level security policy and deliver it to Security Controller.
Developer's Management System (DMS)	A vendor management system to provide an NSF and register its capability with Security Controller.
Network Security Function (NSF)	A security service function to process security requirements as either a Virtual Network Function (VNF) or a Physical Network Function (PNF).
I2NSF Analyzer (IA)	A data analyzer to analyze monitoring data from NSFs for the detection of security attacks and NSF diagnosis.

TABLE II
DESCRIPTION OF INTERFACES IN I2NSF FRAMEWORK

Interface	Description
Registration Interface (RI)	An interface used by DMS to register an NSF's capability with SC.
Consumer-Facing Interface (CFI)	An interface used by IC to deliver a high-level security policy to SC.
NSF-Facing Interface (NFI)	An interface used by SC to deliver a low-level security policy to an NSF.
Monitoring Interface (MI)	An interface used by NSFs to deliver NSF monitoring data to IA.
Analytics Interface (AI)	An interface used by IA to delivery a reconfigured policy or feedback report to SC.

IV. I2NSF Information and Data Models

This section explains the information models and YANG data models of NSF capability and I2NSF interfaces in the I2NSF framework. YANG [5] is a data modeling language used to model configuration and state data, which are manipulated by the Network Configuration Protocol (NETCONF) [6] or the RESTCONF Protocol [7].

We define an information model and a data model as follows.

- **Information Model:** A representation of concepts of interest to an environment in a form that is independent of data repository, data definition language, query language, implementation language, and protocol [16].
- **Data Model:** A representation of concepts of interest to an environment in a form that is dependent on data repository, data definition language, query language, implementation language, and protocol [16].

Fig. 4 shows the registration of an NSF's capability with Security Controller in the I2NSF framework. DMS registers the capability of a new NSF with Security Controller along with the NSF's access information (e.g., IP address and port number) via Registration Interface [11]. The NSF's capability includes the functions of firewall (denoted as FW) and web

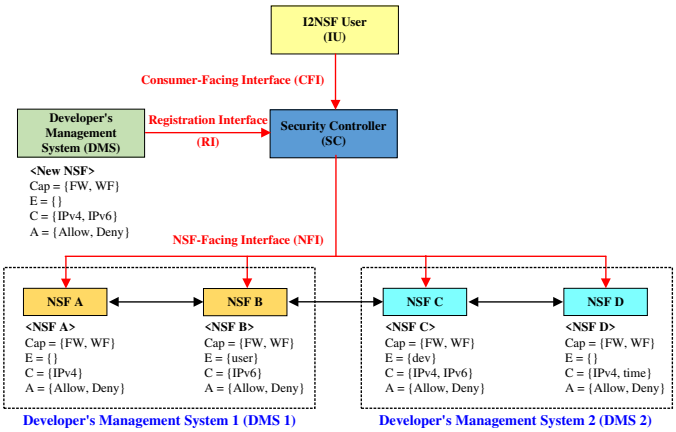
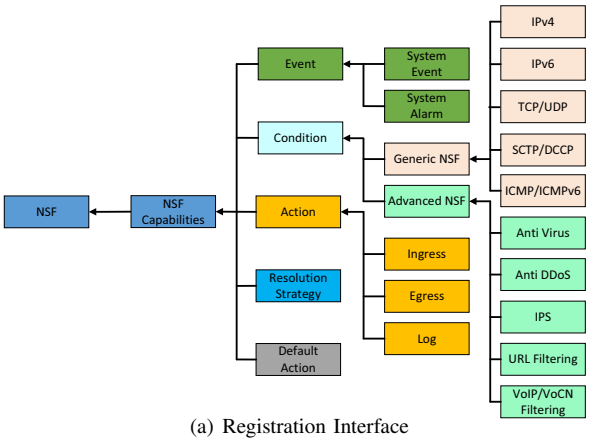
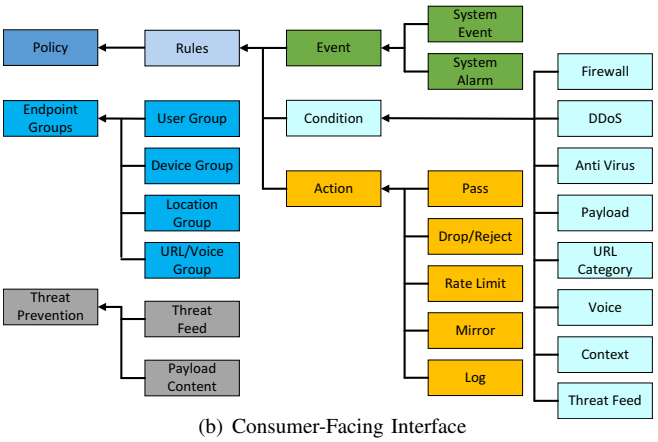


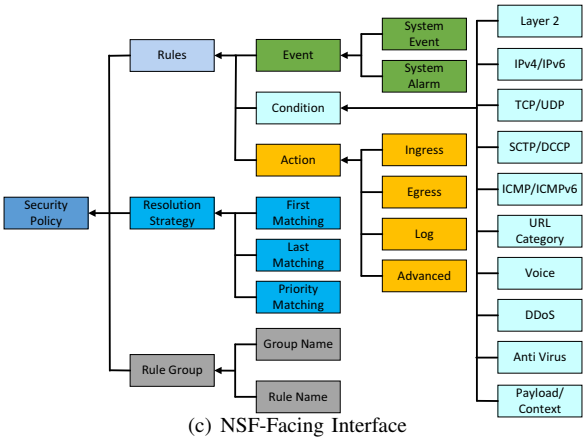
Fig. 4. Registration of an NSF's Capability with Security Controller



(a) Registration Interface



(b) Consumer-Facing Interface



(c) NSF-Facing Interface

Fig. 5. Information Models of I2NSF Interfaces

filter (denoted as WF) in Fig. 4. This NSF supports an Event-Condition-Action (ECA) policy where 'E', 'C', and 'A' stand for "Event", "Condition", and "Action", respectively. "$E = \{\}$" means that the event boolean is always evaluated as true. "$C = \{IPv4, IPv6\}$" means that the condition handles either IPv4 datagrams or IPv6 datagrams. "$A = \{Allow, Deny\}$" means that the action allows or denies a packet according to the condition. As shown in Fig. 4, the capability information of four NSFs (i.e., NSF A, NSF B, NSF C, and NSF D) is shown in the figure, and this information is registered with Security Controller for security services demanded by I2NSF User via Consumer-Facing Interface.

A. Information and Data Models of Registration Interface

The information and data models of Registration Interface (RI) aims at supporting NSF capability registration and query via I2NSF RI [11]. Note that the I2NSF framework in this paper uses NETCONF [6] to deliver an XML file for either NSF capability registration or query via I2NSF RI.

Fig. 5 shows the information models of I2NSF interfaces such as Registration Interface (RI), Consumer-Facing Interface (CFI), and NSF-Facing Interface (NFI). In the information model of RI in Fig. 5(a), an NSF may have multiple capabilities such as event capabilities, condition capabilities, action capabilities, resolution-strategy-capabilities, and default-action-capabilities. Fig. 6 shows a YANG tree of NSF Capability that corresponds to the information model of the NSF capability of RI in Fig. 5(a) [17]. A YANG tree shows the visual representation of a YANG data model. In this figure, the name of the NSF capability module is ietf-i2nsf-capability. The root node of this module is nsf as a container which is a data node in YANG. This nsf node has nsf-name (i.e., the NSF's name), directional-capabilities, event-capabilities, condition-capabilities, action-capabilities, and resolution-strategy-capabilities.

Event capabilities specify the capabilities that describe an event to trigger the evaluation of the condition clause of an I2NSF policy rule [17]. An event may be either a system event or a system alarm [14]. The system event is an event to related to the operations of an NSF, such as access violation, configuration change, session table event, and traffic flows [14]. The system alarm is an event to related to computer hardware's exceeding usage and problem, which draws an attention to make the I2NSF system work properly [14]. As system alarms, there are memory alarm, CPU alarm, disk (storage) alarm, and hardware alarm.

Condition capabilities specify the capabilities of a set of attributes, features, and values to be compared with a set of known attributes, features, and values in order to determine whether a set of actions needs to be executed or not. As

```
module: ietf-i2nsf-capability
 +--rw nsf* [nsf-name]
  +--rw nsf-name               string
  +--rw directional-capabilities*      identityref
  +--rw event-capabilities
  | +--rw system-event-capability*  identityref
  | +--rw system-alarm-capability*  identityref
  +--rw condition-capabilities
  | +--rw generic-nsf-capabilities
  | | +--rw ethernet-capability*  identityref
  | | +--rw ipv4-capability*    identityref
  | | +--rw ipv6-capability*    identityref
  | | +--rw icmpv4-capability*   identityref
  | | +--rw icmpv6-capability*   identityref
  | | +--rw tcp-capability*    identityref
  | | +--rw udp-capability*    identityref
  | | +--rw sctp-capability*    identityref
  | | +--rw dccp-capability*    identityref
  | +--rw advanced-nsf-capabilities
  | | +--rw anti-ddos-capability*       identityref
  | | +--rw ips-capability*          identityref
  | | +--rw anti-virus-capability*       identityref
  | | +--rw url-filtering-capability*     identityref
  | | +--rw voip-vocn-filtering-capability*  identityref
  | +--rw context-capabilities
  |   +--rw time-capabilities*         identityref
  |   +--rw application-filter-capabilities*  identityref
  |   +--rw device-type-capabilities*     identityref
  |   +--rw user-condition-capabilities*    identityref
  |   +--rw geographic-capabilities*      identityref
  +--rw action-capabilities
  | +--rw ingress-action-capability*  identityref
  | +--rw egress-action-capability*  identityref
  | +--rw log-action-capability*    identityref
  +--rw resolution-strategy-capabilities*  identityref
```

Fig. 6. YANG Tree of NSF Capability

conditions, there are matching attributes of a packet or flow, and comparing the internal state of an NSF with a desired state.

Action capabilities specify the capabilities that describe the control and monitoring aspects of traffic flow-based NSFs when the event and condition clauses are satisfied. As actions, providing intrusion detection or prevention, web filtering (i.e., URL filtering), flow filtering, and deep packet inspection for packets or flows.

Resolution strategy capabilities specify the capabilities that describe conflicts occurring between the actions of the similar or different policy rules that are matched and contained in this NSF. Default action capabilities specify the capabilities that describe how to execute I2NSF policy rules when no rule matches a packet. They include the following actions such as pass, drop, reject, rate-limit, and mirror.

B. Information and Data Models of Consumer-Facing Interface

The information and data models of Consumer-Facing Interface (CFI) aims at supporting the construction of a high-level security policy and the delivery of such a policy via I2NSF CFI [12]. Note that the I2NSF framework in this paper uses RESTCONF [7] to deliver an XML file for a high-level security policy via I2NSF CFI.

The YANG data model of CFI defines various types of managed objects and the relationship among them that is need to build the flow policies from I2NSF Users' perspectives. This YANG data model is based on an "Event-Condition-Action" (ECA) policy defined by the I2NSF Capability YANG data model [17]. It allows different I2NSF Users in an I2NSF system to define, manage, and monitor flow policies with an administrative domain (e.g., user group and device group).

In the information model of CFI in Fig. 5(b), a policy may have multiple rules, and each rule consists of event, condition, and action as an ECA policy rule. The YANG data model of CFI defines endpoint groups such user group, device group, location group, URL group, and voice group. It defines threat prevention such threat feed and payload content for deep packet inspection of packets.

C. Information and Data Models of NSF-Facing Interface

The information and data models of NFS-Facing Interface (NFI) aims at supporting the construction of a low-level security policy and the delivery of such a policy via I2NSF NFI [13]. Note that the I2NSF framework in this paper uses NETCONF [6] to deliver an XML file for a low-level security policy via I2NSF NFI.

The YANG data model of NFI defines an interface between a Security Controller and NSFs in an I2NSF system. It is based on the I2NSF Capability YANG data model [17] as a basis YANG data model for the other I2NSF YANG data models.

In the information model of NFI in Fig. 5(c), a policy may have multiple rules, and each rule consists of event, condition, and action as an ECA policy rule like a policy in CFI. For condition, the YANG data model of CFI can specify more details of the condition of a packet in terms of layer-2 header (e.g., Ethernet frame header), layer-3 header (e.g., IPv4/IPv6 datagram header), layer-4 header (e.g., TCP segment header and UDP datagram header), and packet payload (e.g., payload and context).

V. Intent-Based Networking in I2NSF

This section explains intelligent security services through Intent-Based Networking (IBN) based on I2NSF. The following terms are defined in RFC 9315 about "Intent-Based Networking - Concepts and Definitions" [9]:

- **Intent:** A set of operational goals (that a network should meet), and outcomes (that a network is supposed to deliver) defined in a declarative manner without specifying how to achieve or implement them.
- **Intent-Based Networking (IBN):** A network that can be managed using intent.
- **Intent-Based Management**: The concept of performing management based on the concept of intent.

This paper proposes an Intent-Based Management for Cloud-Based Security System. In this paper, a high-level security policy corresponds to an intent as a declarative configuration. On the other hand, a low-level security policy corresponds to an imperative configuration. For IBN-based security services, two functions are provided as follows. First, an automatic translation from a high-level security policy to the corresponding low-level security policy is needed. Second,

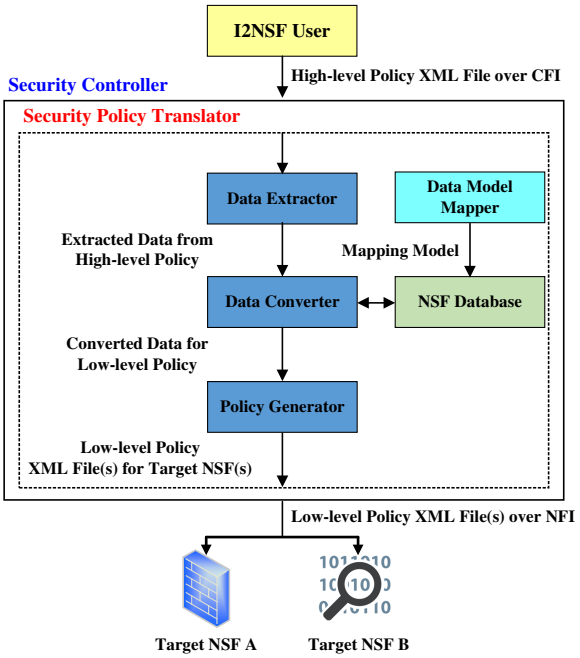


Fig. 7. Security Policy Translator (SPT) Architecture

a closed-loop security control is needed for security management automation in the I2NSF system. These two functions are explained in the following subsections.

A. Security Policy Translator

This subsection explains a Security Policy Translator (SPT) that translates a high-level security policy into a low-level security policy in the I2NSF system. Fig. 7 shows the architecture of SPT which is a part of Security Controller. SPT consists of Data Model Mapper, NSF Database, Data Extractor, Data Converter, and Policy Generator. Table III describes the components in the SPT. As shown in Fig. 7, I2NSF User sends Security Controller a High-level Policy XML File over CFI. SPT in Security Controller translates this XML file into the corresponding Low-level Policy XML Files and sends them to appropriate NSFs over NFI.

Fig. 8 shows an example of Security Policy Translation for time-based firewall (FW) and web filter (WF) services in the I2NSF framework. A high-level security policy is that the employees in a company cannot access SNS websites (e.g., facebook and instagram) during working time (i.e., 9am to 6pm). During the translation, the high-level, abstract data like Employee and SNS Sites are converted to the corresponding low-level, concrete data like IPv4 address range (e.g., 10.0.0.2 to 10.0.0.10) and SNS URLs (e.g., facebook.com and instagram.com).

B. Closed-Loop Security Control

This subsection explains a closed-loop security Control for IBN-based security management automation in an I2NSF system. This closed-loop control can handle two kinds of work such as security policy reconfiguration for new security attacks and feedback report for handling system issues of an

TABLE III
DESCRIPTION OF COMPONENTS IN SECURITY POLICY TRANSLATOR

Component	Description
Data Model Mapper	A component to make a mapping model mapping the elements (i.e., variables) of CFI YANG module into the corresponding elements of NFI YANG module.
NSF Database	A database which stores data model mapping information and the capabilities of NSFs. The mapping can convert an abstract subject or object into the corresponding concrete subject or object (e.g., IP address and website URL).
Data Extractor	A component to extract high-level data from the given high-level policy XML file delivered via CFI as a Deterministic Finite Automaton (DFA).
Data Converter	A component to convert high-level data in CFI to the corresponding low-level data in NFI. It searches for appropriate NSFs with such low-level data as policy provisioning.
Policy Generator	A component to generate a low-level policy XML file with the low-level data through PyangBind [18] having the YANG tree information for NFI.

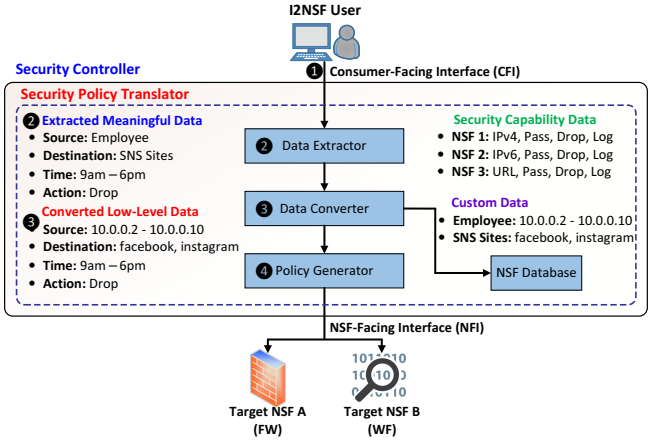


Fig. 8. Example of Security Policy Translation

NSF. First, since a malicious activity by a new security attack can be detected by an NSF or I2NSF Analyzer, this attack needs to be reported to and blocked by the I2NSF system in an automatic, prompt manner. Second, since the hardware overload or malfunction of an NSF is detected by an I2NSF Analyzer, this problem need to be reported to and handled by the I2NSF system in an efficient way.

Fig. 9 shows the closed-loop security control in the I2NSF framework. This closed-loop security control procedure consists of five steps:

1) **Delivery of NSF Monitoring Data:** NSFs deliver their monitoring data to I2NSF Analyzer (IA) via MI.
2) **Machine Learning**: IA analyzes the monitoring data for security attack detection and NSF diagnosis.
3) **Delivery of Reconfigured Policy or Feedback Report**: IA delivers a message (e.g., a reconfigured policy or feedback report) to Security Controller (SC).
4) **Policy Translation**: SC performs policy translation to

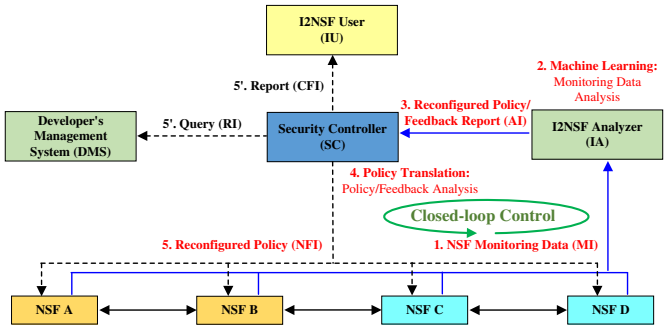


Fig. 9. Closed-Loop Security Control in I2NSF Framework

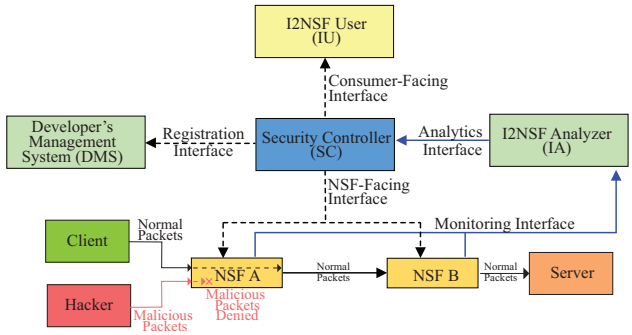


Fig. 10. Experiment Network Topology for I2NSF Framework

analyze the received policy or report and takes appropriate action.

5) **Delivery of Query or Report**: According to the types of the message from IA, SC may send a Query to DMS via RI or send a Report to IU via CFI.

Therefore, CBSS can provide network users with intelligent security services with the intent translation and closed-loop control for self-adaption based on machine learning.

VI. Performance Evaluation

This section explains the performance of our I2NSF system against a hacker's security attack such as a Denial-of-Service (DoS) attack. First, it describes the experiment network topology for performance evaluation and also the open source software for the implemented I2NSF system. Second, it shows the adaptability of a DoS attack in terms of the security detection time according to the number of client hosts in the experiment network.

Fig. 10 illustrates the topology of the experiment. The experiment consists of the following main components:

- **Server:** The central component of the experiment, hosting valuable information and services.
- **Hacker:** An experienced individual attempting to breach the server's defenses using various hacking techniques.
- **Client**: An average client as a user attempting to access the server legitimately.
- **I2NSF Framework**: The main framework used to configure and monitor incoming and outgoing network traffic through NSFs, acting as a barrier between the server and the external network.

TABLE IV
Description of Open Source in I2NSF Framework

Open Source	Version	Role
Node.js [19]	14.17.3	Back-end JavaScript runtime environment.
React [20]	18.2.0	Main library to build interactive user interfaces and web applications.
Python [21]	3.9	The main programming language.
MongoDB [22]	7.0	The database to hold information for the I2NSF framework.
ConfD [23]	8.0.8	Management agent software framework for network elements enabling NETCONF and YANG.
pyang [24]	2.5.3	YANG validator, transformator, and code generator.
PyangBind [18]	0.8.4	Plugin for Pyang that generates a Python class hierarchy from a YANG data model.

Table IV shows the description of open source used in the I2NSF framework. The experiment utilizes the Consumer-Facing Interface YANG data model to deliver the I2NSF User's intent (i.e., a high-level security policy) via RESTCONF in conjunction with a Node.js server environment and the React library to create a user-friendly graphical interface. Within the Security Controller, intent translation (i.e., the translation from a high-level security policy into the corresponding low-level security policy) is executed using Python and MongoDB. The translation output is then enforced through ConfD to use the NSF-Facing Interface YANG data model, which operates the NETCONF server. Additionally, the NSFs feature a subscription-based notification system for I2NSF Analyzer, facilitated by NETCONF event notifications through ConfD. The I2NSF Analyzer subscribes to specific monitoring information from multiple NSFs via Monitoring Interfaces for intent analysis. The I2NSF Analyzer uses a straightforward machine learning technique, such as a decision tree algorithm, and delivers the analytics result to Security Controller via the Analytics Interface that is implemented with NETCONF.

The experiment involves a series of controlled attempts by a hacker's host and a normal client's host to access the server protected by NSFs controlled by the I2NSF framework. Each attempt is monitored and analyzed to evaluate the effectiveness of the protection provided by the I2NSF framework. The hacker attempts a DoS attack on the server, while the normal client accesses the server through authorized channels.

Fig. 11 demonstrates the I2NSF framework's adaptability for our Cloud-Based Security System based on I2NSF. Adaptability refers to the I2NSF framework's capacity to adjust and react to emerging threats. It assesses how swiftly the I2NSF framework can modify its threat detection and protective mechanisms in response to evolving cybersecurity threats. As shown in Fig. 11, the time required for adaptation to a threat like a DoS attack rises with the involvement of more clients. This is due to the increasing complexity of gathering necessary information among a larger client base. However, the impact

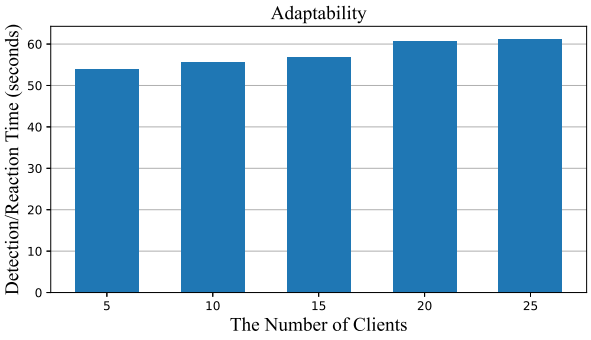


Fig. 11. Detection and Reaction Time of a DoS Attack according to the Number of Clients for Adaptability

on time is not significantly affected by the number of clients.

As a result, the I2NSF framework accommodates diverse types of threats. Whether those threats are a malware outbreak, DoS attack, Distributed Denial-of-Service (DDoS) attack, or novel intrusion technique, the I2NSF framework can adjust its defense against them in a prompt manner. Therefore, this self-defense in the I2NSF framework provides users with a confidence of security in even an environment where the nature of cyber-threats is not only vast but also highly unpredictable.

Note that the I2NSF framework is an open-source project in GitHub (https://github.com/jaehoonpaul/i2nsf-framework), and its demonstration video clip is available in YouTube (https://www.youtube.com/watch?v=ujlicemNo7E).

VII. Conclusion

This paper explains a Cloud-Based Security System (CBSS) with Interface to Network Security Functions (I2NSF). CBSS can be deployed in cloud computing environments such as 5G and IoT networks for intelligent security services. CBSS can be empowered for intent-based security management automation by the features of Intent-Based Networking (IBN), such as security policy translation and closed-loop security control. In this paper, the I2NSF framework and its interfaces are explained to realize the CBSS in the various networks along with the IBN features. Especially, the information and data models of the I2NSF interfaces are explained. As future work, to fully support IBN, security policy validation and optimization in CBSS will be implemented.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government, Ministry of Science and ICT (MSIT)(No. 2023R1A2C2002990), and by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the the Korea MSIT (No. 2022-0-01015).

References

[1] "Multi-access Edge Computing (MEC)," Oct. 2023. [Online]. Available: https://www.etsi.org/technologies/multi-access-edge-computing

[2] S. Hares, D. Lopez, M. Zarny, C. Jacquenet, R. Kumar, and J. P. Jeong, "Interface to Network Security Functions (I2NSF): Problem Statement and Use Cases," *RFC 8192*, Jul. 2017. [Online]. Available: https://datatracker.ietf.org/doc/rfc8192/

[3] "I2NSF Working Group," Oct. 2023. [Online]. Available: https://datatracker.ietf.org/wg/i2nsf/about/

[4] D. Lopez, E. Lopez, L. Dunbar, J. Strassner, and R. Kumar, "Framework for Interface to Network Security Functions," *RFC 8329*, Feb. 2018. [Online]. Available: https://datatracker.ietf.org/doc/rfc8329/

[5] M. Bjorklund, "The YANG 1.1 Data Modeling Language," *RFC 7950*, Aug. 2016. [Online]. Available: https://datatracker.ietf.org/doc/rfc7950/

[6] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, "Network Configuration Protocol (NETCONF)," *RFC 6241*, Jun. 2011. [Online]. Available: https://datatracker.ietf.org/doc/rfc6241/

[7] A. Bierman, M. Bjorklund, and K. Watsen, "RESTCONF Protocol," *RFC 8040*, Jan. 2017. [Online]. Available: https://datatracker.ietf.org/doc/rfc8040/

[8] J. P. Jeong, S. Hyun, T.-J. Ahn, S. Hares, and D. Lopez, "Applicability of Interfaces to Network Security Functions to Network-Based Security Services," *draft-ietf-i2nsf-applicability-18*, Sep. 2019. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-i2nsf-applicability/

[9] A. Clemm, L. Ciavaglia, L. Z. Granville, and J. Tantsura, "Intent-Based Networking - Concepts and Definitions," *RFC 9315*, Oct. 2022. [Online]. Available: https://datatracker.ietf.org/doc/rfc9315/

[10] J. P. Jeong, P. Lingga, P. Jung-Soo, D. Lopez, and S. Hares, "Security Management Automation of Cloud-Based Security Services in I2NSF Framework," *draft-jeong-i2nsf-security-management-automation-06*, Jul. 2023. [Online]. Available: https://datatracker.ietf.org/doc/draft-jeong-i2nsf-security-management-automation/

[11] S. Hyun, J. P. Jeong, T. Roh, S. Wi, and P. Jung-Soo, "I2NSF Registration Interface YANG Data Model for NSF Capability Registration," *draft-ietf-i2nsf-registration-interface-dm-26*, May 2023. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-i2nsf-registration-interface-dm/

[12] J. P. Jeong, C. Chung, T.-J. Ahn, R. Kumar, and S. Hares, "I2NSF Consumer-Facing Interface YANG Data Model," *draft-ietf-i2nsf-consumer-facing-interface-dm-31*, May 2023. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-i2nsf-consumer-facing-interface-dm/

[13] J. T. Kim, J. P. Jeong, P. Jung-Soo, S. Hares, and Q. Lin, "I2NSF Network Security Function-Facing Interface YANG Data Model," *draft-ietf-i2nsf-nsf-facing-interface-dm-29*, Jun. 2022. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-i2nsf-nsf-facing-interface-dm/

[14] J. P. Jeong, P. Lingga, S. Hares, L. Xia, and H. Birkholz, "I2NSF NSF Monitoring Interface YANG Data Model," *draft-ietf-i2nsf-nsf-monitoring-data-model-20*, Jun. 2022. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-i2nsf-nsf-monitoring-data-model/

[15] P. Lingga, J. P. Jeong, and Y. Choi, "I2NSF Analytics Interface YANG Data Model," *draft-lingga-i2nsf-analytics-interface-dm-02*, Jul. 2023. [Online]. Available: https://datatracker.ietf.org/doc/draft-lingga-i2nsf-analytics-interface-dm/

[16] J. Strassner, J. M. Halpern, and S. van der Meer, "Generic Policy Information Model for Simplified Use of Policy Abstractions (SUPA)," *draft-yang-i2nsf-security-policy-translation-15*, May 2017. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-supa-generic-policy-info-model/

[17] S. Hares, J. P. Jeong, J. T. Kim, R. Moskowitz, and Q. Lin, "I2NSF Capability YANG Data Model," *draft-ietf-i2nsf-capability-data-model-32*, May 2022. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-i2nsf-capability-data-model/

[18] R. Shakir, "PyangBind," Oct. 2023. [Online]. Available: https://github.com/robshakir/pyangbind

[19] OpenJS Foundation, "Nodejs," Oct. 2023. [Online]. Available: https://nodejs.org/en

[20] Meta Open Source, "React," Oct. 2023. [Online]. Available: https://react.dev/

[21] Python Software Foundation, "Python," Oct. 2023. [Online]. Available: https://www.python.org/

[22] MongoDB, Inc., "Mongodb," Oct. 2023. [Online]. Available: https://www.mongodb.com/

[23] Tail-f, "ConfD," Oct. 2023. [Online]. Available: https://www.tail-f.com/confd-basic/

[24] M. Bjorklund, "pyang," Oct. 2023. [Online]. Available: https://github.com/mbj4668/pyang