International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-3, September 2019

Software Defined Network Detection System

Sang Boem Lim

Abstract: ***The ongoing increase in the use of wireless Internet and smartphones has resulted in changing consumer patterns, which has changed the demand for network usage such that existing hardware-centric devices cannot satisfy this demand. One of the fastest growing technologies is software define network, which can solve this problem. An intrusion detection system is a system that detects and responds to network attacks in real time in a network environment based on software define network. The focus of this paper is to present a deep learning-based network detection system. We describe pre-processing for deep learning algorithms and propose an architecture of the detection system. The analysis results of the system are also described.***

Index Terms: Deep Learning, SDN, ONOS, Network Detection.

I. INTRODUCTION

The increasing use of mobile devices, OTT (Over the Top) service, the IoT (Internet of Things), and cloud-based systems has led to a rapid increase in demand for massive data movement at high speeds. Since hardware devices on the existing network have been released as firmware, it is time consuming and expensive to perform an upgrade when demand on the network increases. Internet service providers need devices that can satisfy various user demands and support new service business by enabling rapid upgrades. SDN solves the problem that cannot be solved with existing traditional hardware-centric network devices, namely to perform the role of a network controller that uses software to control network resources [1]. Among them, ONOS (Open Network Operating System) was developed in 2014 by ONLab as an open source project in which most vendors participate and donate to; thus, it is possible to develop various applications more easily than for other controllers. Although SDN is more convenient and efficient than traditional network management, several security vulnerabilities were found and countermeasures were developed [2][3]. However, many deficiencies in terms of security vulnerabilities continue to exist in ONOS, which is used as a controller. Firewall implementations against static network attacks have been developed as an ACL (Access Control List) application, but implementations of intrusion detection for dynamic continuous attacks are still under development. Although SK Company is pursuing research on the 5G Internet network using ONOS in Korea, it will take time until it becomes a commercial service.

In this research, we aim to design a system that provides automatic intrusion detection in an SDN-based environment. In addition, we adopt machine-learning algorithms to implement intrusion detection using the ONOS controller. In Section II, we introduce SDN-related technologies and an IDS (Intrusion Detection System). In Section III, we select the most accurate machine-learning algorithm by using a sample dataset to compare results from different work. In addition, we describe the architecture of our system and process scenario. In Section IV, we introduce our development environment, network topology, and implemented features. We analyze the test result in Section V and present the conclusion and describe future work in Section VI.

II. BACKGROUND AND RELATED WORK

A. Software Defined Network

In this section, we describe SDN-related technologies such as SDN structure, OpenFlow, and the ONOS controller, after which we introduce the IDS system.

In the 1980s, SDN was initially proposed as a method to control a network with software in a research field related to IN (Intelligent Network) technology. SDN was released as open source software in 2008 with the OpenFlow project at Stanford University. SDN provides a standardized interface by separating the control layer and the data layer as shown in Figure 1 [4][5]. Thus, it is possible to manage a network by programming the control layer and by using the data layer to develop various applications such as network monitoring and a firewall [6]. This kind of controlling work provides several APIs for implementing various network services on an SDN-based controller.

Between the control layer and infrastructure layer, the Southbound API provides a standardized protocol, which can control physical network devices using software. OpenFlow is the most widely used open source standardized protocol. It is a standard communication protocol between the control layer and data layer of an SDN and is developed by the ONF (Open Networking Foundation). OpenFlow provides a standard protocol for switching network control features from hardware to central controllable software. It also functions as an interface that exists between devices and the SDN controller and runs in both hardware and software domains [7]. OpenFlow is composed of a controller and OpenFlow Switch, which is in a secure channel, as shown in Figure 2. The controller sends, modifies, and discards packets to its destination through the protocol to the switch.

Manuscript published on 30 September 2019
* Correspondence Author
Sang Boem Lim*, Department of Smart ICT Convergence, Konkuk University, Seoul, Korea, sblim@konkuk.ac.kr

The path of the packet is determined by the parameter information at the times of packet transmission. The path is calculated by OpenFlow and then sent to the switch and stored in the Flow Table. Each time a switch receives a packet; it checks the Flow Table and sends the packet along the specified path.

In this construction, an SDN controller is required to control and manage various types of SDN including OpenFlow.

The SDN controller can be roughly divided into a centralized structure and a distributed structure including a hierarchical structure. The centralized structure refers to a single controller that centrally controls and manages all the network devices, thereby leading to the bottleneck phenomenon. This problem can be solved by using a distributed SDN controller. The use of distributed controllers is one of the more recent emerging technologies capable of mitigating bottlenecks in centralized SDN controllers.

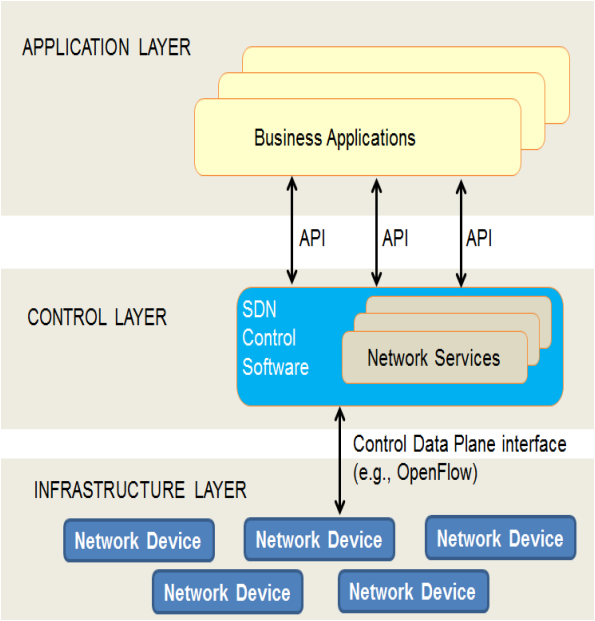

Figure 1. Structure of an SDN [4]

One of the most representative distributed SDN controllers is ONOS, which provides a Web GUI (Graphic User Interface) and plug-in to facilitate the development of ONOS-based applications. By providing a hierarchical structure on a per-component basis, developers are able to develop applications and provide features that can be used on the controller [8][9]. The purpose of the ONOS project is to address the needs of excessive network usage. Therefore, it is characterized by high performance and high-efficiency processing such as high throughput and a short waiting time. It also provides useful Northbound abstraction and APIs to make it easy to develop applications; it creates Southbound abstraction and interfaces to control not only OpenFlow-enabled devices but also traditional legacy devices. ONOS can also provide a stable environment compared to other SDN controllers such as NOX, Beacon, SNAC, and POX [10].

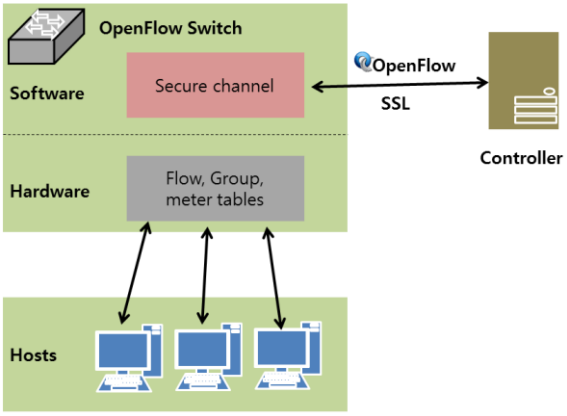

Figure 2. Construction of OpenFlow [7]

B. Intrusion Detection System

An IDS can detect attacks to server or network domains in real time. It is a network security solution that provides security by using a firewall that can prevent attacks when the firewall is blocked or the network administrator is not available. It monitors every attempt to access the network and use the system and immediately blocks these in real time if suspicious commands are given or movements are performed. The detection engine can withstand not only external intrusion, but also illegal internal or abusive action [11]. It can also minimize the damage when the firewall is attacked. Therefore, the IDS monitors the network and detects intrusion in real time. It provides statistical analysis with various kinds of information, observation, and ways to defend against new intrusion patterns.

In this paper, intrusion detection is implemented by using a machine-learning algorithm in an SDN-based environment, which is controlled by an ONOS controller. It is necessary to pre-analyze the most accurate machine-learning algorithms. Depending on this analysis, the automatic intrusion detection system, scenario, and architecture are designed.

III. DESIGN OF SYSTEM

In this section, we describe pre-processing for machine-learning algorithms for intrusion detection in an SDN-based environment. Pre-processing includes data collection, data feature extraction, and algorithm selection. According to this pre-processing step, we introduce the architecture for an automatic intrusion detection system and test-bed environment.

A. Pre-processing

Similar to the way in which a person gains knowledge through learning and making decisions, a computer also learns from data collection through machine learning and makes decisions. Machine learning requires three steps [12][13][14]. First, it is necessary to collect data for learning. Second, machine learning extracts meaningful information that provides the basis of feature extraction, which is for making decisions. Third, we have to choose the most accurate method or algorithm for learning. The following subsections describe these three procedures in detail.

Data Collection

A computer needs a sufficient number of datasets to use for learning purposes in machine-learning methods. A widely used dataset for evaluating the performance of an intrusion detection system is KDD'99 (Knowledge Discovery and Data Mining) [15][16][17][18]. This dataset contains network traffic information with normal and abnormal connections. Four different patterns of abnormal attacks are represented, i.e., DoS (Denial of Service), U2R (User to Root), R2L (Remote to Local), and probing attacks. In addition, a normal connection is classified as a normal attack [16][17]. For SDN-based intrusion detection, the KDD'99 dataset is used in the performance evaluation of the IDS in this study.

Data Feature Extraction

In an SDN environment, the network traffic flow is controlled by the flow unit. Information about networking flow can be obtained through the OpenFlow protocol. In order to extract the features for classifying attacks using network flow information, we analyze the characteristics of the methods.

We chose the widely-known DoS attack for analysis [19]. A DoS attack can deplete a network or system resource by allowing many packets to flow through the network and sending packets to a specific host, server, or service. Therefore, a DoS attack can be detected by analyzing the packet IP header, TCP header, ICMP, TCP, UDP protocol, and so on.

Algorithm Selection

The supervised method is well known as a type of intrusion detection machine-learning algorithm. This method includes the *BayesNet, NaïveBayes, J48, SMO, Multilayer Perceptron,* and *RBFNetwork* algorithms [20]. An algorithm with high accuracy was selected by carrying out a performance evaluation for each algorithm on the KDD'99 dataset. Table 1 contains details of the datasets used in the performance evaluation. Through pre-processing, we collect 16 types of features that include both training and testing datasets. Each of them contains abnormal and normal data.

Table 1. Sample Dataset

KDD'99	Abnormal	Normal
Training Dataset	396743	97278
Testing Dataset	250436	60593

The process of measuring the accuracy using various machine-learning algorithms includes the following steps. First, the training dataset is used as input into each algorithm for modeling. Then, the testing dataset is inputted for evaluation by each model using the algorithm. Table 2 contains the measured result by each algorithm. As shown the table, the highest performance score of 93.25 is obtained by the J48 algorithm. The second highest performance score is 91.99 and is obtained by the Multilayer Perceptron algorithm. BayesNet obtains 91.69, SMO obtains 91.55, NaïveBayes obtains 91.04, and RBFNetwork obtains 80.82, which is the lowest score. We therefore chose to use the most accurate algorithm, J48, to implement in our system.

Table 2. Algorithm Accuracy Measurement Results

Algorithm	Accuracy
BayesNet	91.69
NaiveBayes	91.04
J48	93.25
SMO	91.55
Multilayer Perceptron	91.99
RBFNetwork	80.82

B. Design of System Architecture

The system architecture is designed as shown in Fig. 3. Our aim is to implement an SDN-based automatic IDS using an ONOS controller as an application to detect and defend against network attacks. Our system consists of three layers: the infrastructure layer, control layer, and application layer.

Infrastructure Layer

The infrastructure layer includes the network devices and hosts that form the physical part of the SDN-based network environment. In this layer, when attackers launch attacks on victims through switches, the ONOS controller in the control layer controls the packet flows by managing the information in the header of the network packet, including the source and destination of the packet.

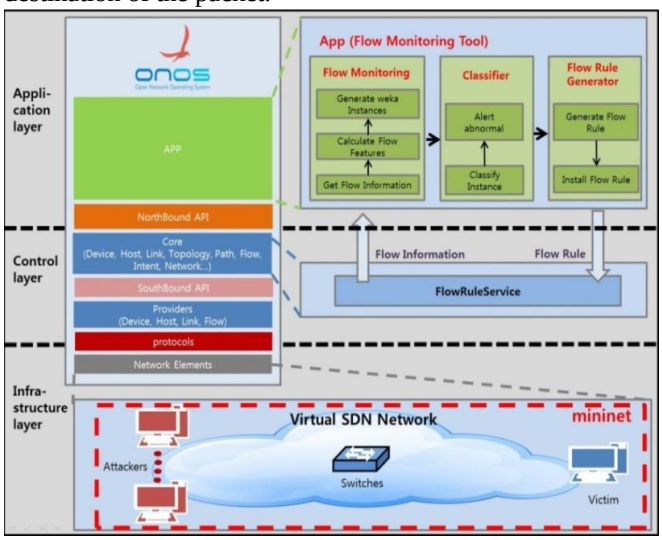

Figure 3. System Architecture

Control Layer

In the control layer, the controller provides various APIs to manage and control applications. Network intrusion detection requires flow information to be periodically collected. The flow rules, which are usually stored in the switch, can be obtained by using FlowRuleService, which is provided in the control layer.

Application Layer

The application layer provides a flow-monitoring tool that includes three functions, namely flow monitoring, a classifier, and a flow rule generator. When the monitoring tool in the application layer is activated, as shown in Fig. 4, it loads the model for attack determination using the machine-learning Weka tool.

Next, it uses the classifier every 10 seconds to classify instances using the model. If the result of the classification is an abnormal attack, the rule generator generates flow rules for dropping. If the result of the classification is normal, the application is activated to obtain flow rules every two seconds. Flow monitoring monitors instances from static traffic features that are generated by the flow rule.

The machine-learning algorithm uses the model to create a packet from outside the application and then send to the application. According to this model prepared with Weka, it classifies datasets that are collected by using the Flow Monitoring Tool. If the packet flow is that of an abnormal attack, it directly calls the Flow Rule Generator. The measured abnormal attack flow creates new rules that include the destination/source IP address, destination/source TCP/UDP port, protocol, duration, priority, and drop actions. The new rules are stored in the switch through FlowRuleService. New network packets that are matched with rules are blocked by the drop action when an intrusion attack is detected.

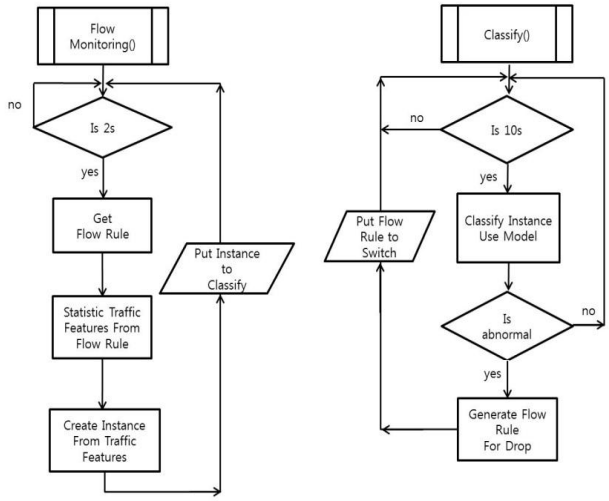

Figure 4. Diagram of Application Layer

C. Test-bed Environment

As shown in Table 3, we use ONOS version 1.7.0 as a controller to detect and defend against an intrusion. In order to construct a virtual network environment, we use the mininet simulation tool. By adopting machine learning, we use the Weka data-mining tool to provide a machine-learning algorithm and evaluate the performance. The hping3 tool is used to generate both attack packets and normal packets. We created the testing environment by creating a virtual machine with Ubuntu version 14.04 to configure, install, and test our system.

Table 3. Environment Implementation Tools

Tools	Version	Remarks
ONOS	1.7.0	Controller of SDN
Mininet	2.2.0	Network Environment Simulator
Weka	3.8.0	Data Mining Tool
Hping3	-	Packet Generator

IV. ANALYSIS

Our system evaluates its detection intrusion and defense function by creating attack packets using hping3. When the ONOS application is activated to run, it can automatically recognize attack packets and test whether blocking rules are generated. We used the packet monitoring tool Wireshark to monitor the network packets generated by hping3. This enabled us to verify whether attacks were blocked by generating rules with the ONOS controller.

Fig. 5 shows the flow status for the switch, which is created by the ONOS controller by default. The value in the red box shows the accepted flow rule. In order to detect attack packets, system creates new rules by FlowRuleGenerator to add to this table. Fig. 6 shows the flow status after new flow rules have been added. The red box indicates the new flow rules added by the FlowRuleGenerator. Depending on these newly added rules, attack packets are deleted, as shown in Fig. 7. We used Wireshark to monitor incoming attack packets, as shown in Fig.8. After detecting attack packets, the system automatically deletes these packets, as shown in Fig. 9.

Figure 5. Default Flow Status

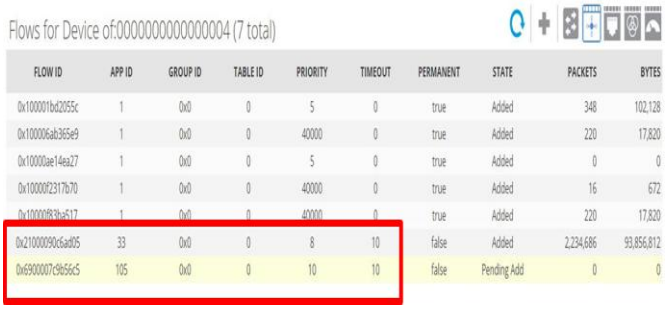

Figure 6. Added New Flow Rules

Figure 7. Deleted Flow Rules Status

V. CONCLUSION AND FUTURE WORK

In this study, we aimed to develop an SDN-based automatic intrusion detection system using an ONOS controller. We adopted the most accurate machine-learning algorithm to develop the ONOS application prototype. The most accurate algorithm was selected by using the KDD'99 dataset to create training and testing datasets for evaluating the performance. We used Weka to implement the most accurate algorithm, J48, in our system. We implemented the ONOS application that performs SDN-based intrusion detection and defense functions by composing three main functions, i.e., Flow Monitoring, Classify, and FlowRuleGenerator, in the application layer.

In the future, we aim to test additional machine-learning algorithms to solve more complex types of data and topologies. Although it was not proposed to develop security applications as a commonly used solution, our system can provide a simple standard interface to implement secure automatic intrusion detection when configuring an SDN-based environment using the ONOS controller.

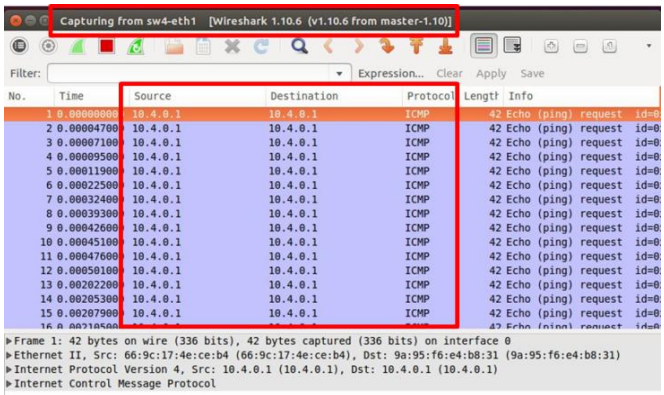

Figure 8. Incoming Attack Packets

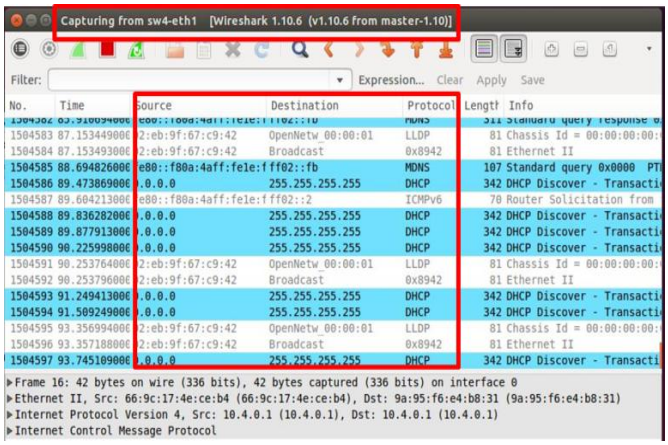

Figure 9. Result of Intrusion Detection

REFERENCES

1. ONF Market Education Committee, "Software-Defined Networking: the New Norm for Networks," ONF White Paper, Palo Alto, US: Open Networking Foundation, Apr. 2012
2. K. Poularakis, Q. Qin, L. Ma, S. Kompella, K. K. Leung, and L. Tassiulas, "Learning the Optimal Synchronization Rates in Distributed SDN Control Architectures" in 2019 IEEE International Conference on Communications (ICC) ICC 2019, pp. 1-7, 2019.
3. V. R. Dasari, T. S Humble, "OpenFlow arbitrated programmable network channels for managing quantum metadata," The Journal of Defense Modeling and Simulation, Vol. 16, no. 1, pp. 67-77, Jan. 2019.
4. T. Kim, J. Myung, S. Yoo, "Load Balancing of Distributed Datastore in OpenDaylight Controller Cluster," IEEE Transactions on Network and Service Management, Vol. 16, no. 1, pp. 72-83, Mar. 2019.
5. A. Vishnu Priya and N. Radhika, "Performance comparison of SDN OpenFlow controllers," International Journal of Computer Aided Engineering and Technology, Vol. 11, no. 4-5, pp. Mar. 2019.
6. N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, "Survey on SDN based network intrusion detection system using machine learning approaches", Peer-to-Peer Netw. Appl., Vol. 12, no. 2, pp. 493-501, Mar. 2019.
7. Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry L. Peterson, Jennifer Rexford, Scott Shenker, and Jonathan S Turner, "OpenFlow: enabling innovation in campus networks," ACM SIGCOMM Computer Communication Review, Vol. 38, no.2, pp. 69-74, Apr. 2008.
8. H. Maziku, S. Shetty, and D. M. Nicol, "Security risk assessment for SDN-enabled smart grids," Computer Communications, Vol. 133, pp. 1-11, Jan. 2019.
9. ONF, ONF. "Openflow switch specification version 1.3.0(wire protocol 0x04)," 2012.
10. ON.LAB, "Introducing ONOS – a SDN network operating system for Service Providers," ON.LAB White Paper, 2014
11. J. Singh and M. J. Nene. "A survey on machine learning techniques for intrusion detection systems," International Journal of Advanced Research in Computer and Communication Engineering, Vol 2 no.11, 4349-4355, Nov. 2013.
12. S. Dotcenko, A. Vladyko, and I. Letenko. "A fuzzy logic-based information security management for software-defined networks," in 16th. IEEE International Conference on Advanced Communication Technology (ICACT), 2014.
13. K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments," Computer Networks, Vol. 62, pp. 122-136, Apr. 2014.
14. R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in 2010 IEEE 35th Conference on Local Computer Networks (LCN), IEEE, 2010.
15. I. Obeidat, N. Hamadneh, M. Alkasassbeh, M. Almseidin, and M. AlZubi, "Intensive Pre-Processing of KDD Cup 99 for Network Intrusion Classification Using Machine Learning Techniques," International Association of Online Engineering, 2019.
16. K. Siddique, Z. Akhtar, F. A. Khan and Y. Kim, "KDD Cup 99 Data Sets: A Perspective on the Role of Data Sets in Network Intrusion Detection Research," Computer, Vol. 52, no. 2, pp. 41-51, Feb. 2019.
17. S. El-Sappagh, A. S. Mohammed, and T. A. AlSheshtawy, "Classification Procedures for Intrusion Detection based on KDD CUP 99 Data Set," International Journal of Network Security & Its Applications (IJNSA), Vol. 11, no.3, May 2019.
18. D. Jankowski and M. Amanowicz, "Intrusion detection in software defined networks with self-organized maps," Journal of Telecommunications and Information Technology, Vol. 4, pp. p3-9 Jan. 2015.
19. S. Denga, X. Gaob, Z. Lua, Z. Lia, and X. Gao, "DoS vulnerabilities and mitigation strategies in software-defined networks," Journal of Network and Computer Applications, Vo. 125, no. 1, pp. 209-219, Jan. 2019.
20. T. Pham, Q. Son, N. U. Nguyen, and X. H. Nguyen, "Generating artificial attack data for intrusion detection using machine learning," In Proceedings of the Fifth Symposium on Information and Communication Technology. ACM, pp. 286-291, Dec. 2014.

AUTHORS PROFILE

Sang Boem Lim, Prof. Sang Boem Lim received his PhD in Computer Science from Florida State University in 2003. Previously, he leaded Korea e-Science Project as technical Team Leader at Korea Institute of Science Technology Information (KISTI) Supercomputing Center. He currently is a professor for Konkuk University. His research interests include high-performance computing and cloud computing.