Suspicious Traffic Sampling for Intrusion Detection in Software-Defined Networks

Taejin Ha, Sunghwan Kim, Namwon An, Jargalsaikhan Narantuya, Chiwook Jeong, JongWon Kim, and Hyuk Lim

Abstract

In order to defend a cloud computing system from security attackers, an intrusion detection system (IDS) is widely used to inspect suspicious traffic on the network. However, the processing capacity of an IDS is much smaller than the amount of traffic to be inspected in a large-scaled network system. In this paper, we propose a traffic sampling strategy for software-defined networking (SDN) that fully utilizes the inspection capability of malicious traffic, while maintaining the total aggregate volume of the sampled traffic below the inspection processing capacity of the IDS. We formulate an optimization problem to find an appropriate sampling rate for each switch, and sample the traffic flows in the network according to the optimal sampling rates using the SDN functionalities. The simulation and experiment results indicate that the proposed approach significantly enhances the inspection performance of malicious traffic in large-sized networks.

Index Terms

Intrusion detection, cloud technology, software-defined network, traffic sampling

I. Introduction

Cloud technologies have become some of the most promising next-generation technologies for the efficient and cost-effective management and operation of diverse industrial applications. A cloud system is a distributed platform that manages and coordinates a number of distributed devices and services, and provides a wide range of services and functions. Cloud services are

The authors are with the School of Information and Communications, Gwangju Institute of Science and Technology (GIST), Gwangju 61005, Republic of Korea. Email: hlim@gist.ac.kr

A part of this work has been demonstrated at the demo session in IEEE International Conference on Cloud Networking (CloudNet), October 2014 [1].

expected to be deployed to enhance the operation and management efficiency for a variety of Internet application and service systems. To strengthen the security in a cloud system, an intrusion detection system (IDS) is widely deployed. An IDS is one of the most important means for protecting a network from threats. It is used to monitor network behaviors and inspect data packets for detecting malicious activities. In a conventional network, the IDS is configured to operate in two modes, passive and in-line modes [2]. In the passive mode, the IDS is connected to the network as a subordinate of a network node, and it passively receives and inspects data packets captured from the node. On the other hand, in the in-line mode, the IDS is set up as a stand-alone node in the network. In both modes, the IDS is located at a certain location connected to a network link and conducts a traffic analysis on local flows going through the link.

However, because of an explosive expansion in the network scale and the increase in network traffic, it has become much harder to determine the inspection points where the IDS is placed and where the data packets are captured within the network. This is also because the IDS has a limitation of hardware resources in terms of CPU power, memory access speed, and storage capacity [3]. In particular, for a large-scale network security system, a number of IDSs are needed to be deployed to inspect all data packets owing to their inspection capability, which incurs high costs.

With the aim of achieving a scalable inspection of a large-scaled network with a limited capacity IDS, we use software defined networking (SDN) technology. SDN is an emerging network architecture that decouples the network control plane from the packet forwarding plane (data plane) [4]. It has a centralized controller, called an SDN controller, which is responsible for all network control decisions of the network-wide distributed forwarding elements [5]. The controller uses an OpenFlow (OF) protocol, which is a communication networking protocol, to access OF-enabled switches using OF APIs [5]. Because the scale of cloud systems is expanding, and communication network management between distributed server systems requires diverse functionalities, SDN is a promising solution for the networking architecture connecting between the server host systems. Under an SDN-based network system, it is possible to easily sample data traffic from multiple switches and forward them to one of the IDSs in the cloud system [4].

In this paper, we consider an IDS to prevent malicious data propagation in an SDN-based network. If the network traffic to be inspected is much larger than the IDS capacity, the IDS cannot inspect all packets in the network. Therefore, it is desirable to sample a certain amount of data traffic from the network switches and forward it to the IDS using the SDN functionalities.

We propose a sampling rate adjustment method that determines the appropriate sampling rates at the network switches for fully utilizing the inspection capability of malicious traffic while the total aggregate volume of the sampled traffic is kept below the maximum processing capacity of the IDS.

The remainder of this paper is organized as follows. In Section II, we provide an overview of previous related work. In Section III, we present the system model and derive a relationship between the sampling rate and the capture-failure rate of malicious traffic when the network traffic is sampled. Then, in Section IV, we propose a measurement-based sampling method along with the details regarding the algorithm used for obtaining the appropriate sampling rates. In Sections V and VI, a performance evaluation is presented, followed by concluding remarks in Section VII.

II. Related work

There has been a significant amount of work on intrusion detection technology and systems. Most research has focused on how to process a large amount of traffic samples efficiently. They proposed to use various traffic characteristics such as the flow size, flow statistics, and flow entropy changes [3], [6], [7]. In [3], Androulidakis *et al.* proposed a flow-size based sampling technique. The authors considered that network attacks usually use small flows as a traffic source. Based on the observation, those flows smaller in size than a certain threshold are sampled with a constant probability. In [6], Kawahara *et al.* introduced a flow-statistic based intrusion detection strategy. The authors noted that the number of flows increases dramatically if a network is under attack. They spatially partitioned sampled traffic into several groups according to a source-autonomous system. Their experimental results indicate that an analysis of the flow statistics for individual groups can enhance the ability of intrusion detection. In [8], Mai *et al.* showed how traffic sampling degrades the detection ability of non-volume dependent anomalies. The authors used three intrusion detection algorithms to detect non-volume based attacks from both the original and sampled trace data. The results indicate that traffic sampling can increase the number of false positives and degrade the detection ability. In [9], Kacha *et al.* proposed a new pattern matching technique for improving the performance of Snort IDS. To reduce a rate of false alarms, it combined misuse and anomaly detection techniques. This method provides faster packet inspection with less consumption of the IDS resource compared to the conventional one.

Recently, research efforts have been made to exploit the emerging SDN technology for network security by a number of research groups. In [2], Shin *et al.* introduced a framework, called Cloudwatcher, that provides monitoring service for cloud networks. Using SDN technology, Cloudwatcher changes the paths of the network flows to allow them to pass through more secure links, or their data packets to be inspected by an IDS. In [10], Khurshid *et al.* used a packet forwarding operation of an SDN in real time for eliminating network errors such as routing loops, black holes, and access control violations. Because these errors mostly result in the unavailability of a service, they are weak points and make a network vulnerable to network attacks. In [11], Jafarian *et al.* proposed a technique for changing the Internet protocol (IP) address to protect an end node from attackers. This technique frequently mutates the actual IP address of the end-host into a fake virtual IP address. In [7], Giotis *et al.* proposed an entropy-based detection algorithm that measures the randomness of specific data sets in an SDN environment. The authors classified network attacks by observing the entropy changes. For instance, if there is a significant decrease in the number of destination IPs and destination ports, the network is considered to be under a DDoS attack.

Most of the existing work has focused on the sampling and inspection of traffic packets observed from only a single point in either a traditional or SDN-based network. In contrast, we propose a distributed sampling scheme that samples suspicious data packets from multiple switches on an SDN-based network. In consideration of the IDS inspection capacity, the proposed scheme determines the appropriate sampling rate for each switch to fully utilize the detection ability of an IDS. We provide an analytical model for evaluating the capture-failure rate of malicious traffic in an SDN. The network simulation results indicate considerable gains in the detection of malicious traffic when the IDS capacity is much smaller than the total traffic to be inspected.

III. System Model

A. System Description

We consider an SDN-based network shown in Figure 1. The SDN-based network architecture is composed of an SDN controller and OF-enabled switches. OF is a communication networking protocol that enables the SDN controller to access the forwarding table of the switches. The SDN controller connects with the OF-enabled switches on the control plane, and by using OF, it can

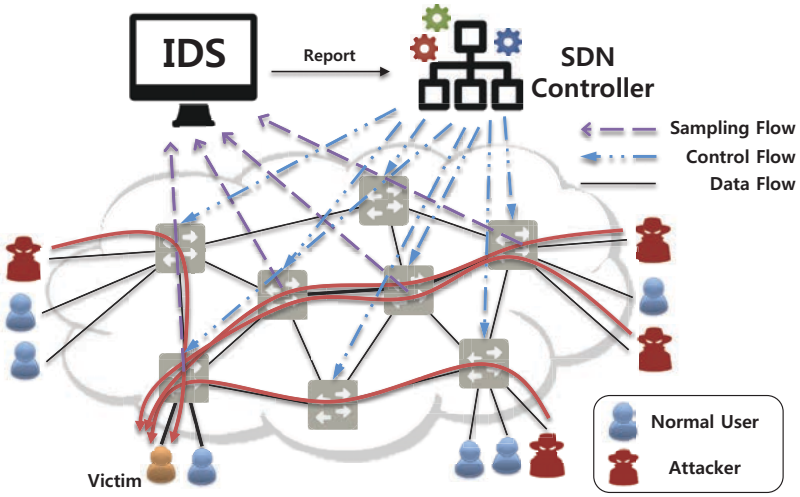


Fig. 1. Suspicious traffic inspection on SDN-based network with IDS.

gather the flow status at each switch and control its flow forwarding table. All data packets are exchanged through OF-enabled switches operated by the SDN controller.

An IDS is usually used to inspect all or a certain number of packets that the IDS can capture at its connected links. A signature based IDS compares captured packets against a database of signatures from known threats. For example, Snort IDS [12] is one of the popular open-source IDSs, and it can detect malicious probes and attacks such as server message block probes, stealth port scans, and malicious code injection by analyzing network traffic against a rule set. It can also detect protocol anomalies such as TCP SYN flood attacks using a variety of preprocessors.

Figure 1 shows the suspicious traffic inspection on SDN-based network with IDS. For an SDN-based IDS, it is possible to sample the data packets at any OF-enabled switches by using a mirroring method at each switch, which is fully configurable by the SDN controller. The IDS then inspects all of the data packets that are mirrored from the switches, and generates a security alarm if it detects a network attack or suspicious packet. The security alarm is fed back to the SDN controller. Based on the inspection results of the IDS and the current status of the switches, the SDN controller can reconfigure the network to defend against an attack and make the network more secure.

While the SDN technology enables traffic flows at the switches to be sampled and forwarded to the IDS, the IDS cannot inspect all sampled packets if the sampled traffic volume is greater than its processing capacity. Therefore, an algorithm for determining the sampling rates of the

switches needs be developed to ensure that the total amount of sampled traffic is kept below the maximum inspection capacity of the IDS while minimizing the rate of missing malicious traffic packets.

B. Network Model

We assume that there are f flows and n OF-enabled switches in the network. Let λ_i denote the malicious rate belonging to the ith flow. If a flow does not include any malicious packets, its malicious rate is 0. The sending rate (or data rate) of the flows is denoted by sending rate vector $\boldsymbol{s} = [s_1, \cdots, s_f]^T$, and that of malicious flows is $\boldsymbol{\lambda} = [\lambda_1, \cdots, \lambda_f]^T$. The unit for both the sending and malicious rate is packets per second (pps).

As mentioned before, the flow status at each switch can be available at the SDN controller. Based on this flow information, it is possible to generate a flow path information matrix, $\boldsymbol{A}$, as a routing table used in conventional IP networks. Here, $\boldsymbol{A}$ is an n-by-f binary matrix, and its element $a_{i,j}$ is 1 when the jth flow passes the ith switch. Otherwise, it is 0. By utilizing $\boldsymbol{A}$, the data rate of the switches, $\boldsymbol{b}$, whose element b_i denotes the data rate of the ith switch, is obtained as follows:

$$\boldsymbol{b} = \boldsymbol{A} \cdot \boldsymbol{s}. \tag{1}$$

Each switch samples the packets at a certain rate for a malicious traffic inspection. Let $\boldsymbol{x}$ denote the sampling rate vector, whose element x_i is the sampling rate at the ith switch.

$$\boldsymbol{x} = \{x_1, \cdots, x_n\}, \quad 0 \leq x_i \leq 1 \ \text{ for all } i\text{'s}. \tag{2}$$

For example, if x_i is 1, all packets passing the ith switch are sampled and forwarded to the IDS.

C. Sampling of Suspicious Traffic

One of the performance metrics that is widely used for intrusion detection is the false-negative rate. A false negative implies that the IDS has not detected a malicious attack that has taken place because the IDS fails to classify the malicious packet. In this paper, we consider another metric for detection failure of malicious attacks. If malicious packets belonging to a malicious attack are not captured, the malicious attack cannot be detected by the IDS. Thus, we focus on a *capture-failure rate* instead of a false-negative rate. The term of "capture-failure," from the perspective of an IDS, indicates the unawareness of an attack. Because we sample a certain amount of

traffic for inspection rather than all the packets belonging to a traffic flow, the performance of suspicious traffic inspection can be quantified based on the capture-failure rate.

Let $\boldsymbol{p}$ denote the capture-failure rate vector, where p_i is the capture-failure rate of the ith flow.

$$\boldsymbol{p} = \{p_1, \cdots, p_f\}, \quad 0 \leq p_i \leq 1 \quad \text{for all } i\text{'s}. \tag{3}$$

A flow passes through several switches while it is relayed toward its destination. If none of the malicious packets in a flow are sampled at any of the intermediate switches, the IDS cannot detect them. Therefore, the capture-failure rate of an ith flow that includes malicious packets is the product of the capture-failure rates at every intermediate switch.

$$p_i = \prod_{j \in \alpha(i)} p_{i,j}, \tag{4}$$

where $\alpha(i)$ is a set of switches that the ith flow passes, and $p_{i,j}$ is the capture-failure rate of the ith flow at the jth switch. It is important to note that the capture-failure rate in (4) is the probability that malicious packets fail to be captured at every intermediate switch. Therefore, the capture-failure rate should be minimized to have higher chance of detecting attacks.

First, assume that x_j, b_j, and, $x_j \cdot b_j$ are natural numbers. The process of sampling the data packets is then the same as picking $x_j \cdot b_j$ balls out of a total of b_j balls. Therefore, the capture-failure rate $p_{i,j}$ can be calculated as follows:

$$p_{i,j} = \begin{cases} 0 & \text{if } b_j - \lambda_i \leq x_j \cdot b_j, \\ \frac{\binom{b_j - \lambda_i}{x_j \cdot b_j}}{\binom{b_j}{x_j \cdot b_j}} = \frac{(b_j - \lambda_i)! \cdot (b_j - x_j \cdot b_j)!}{b_j! \cdot (b_j - \lambda_i - x_j \cdot b_j)!} & \text{otherwise.} \end{cases} \tag{5}$$

If the number of normal packets is less than that of the sampled packets, then $(b_j - \lambda_i - x_j \cdot b_j)$ becomes negative. In this case, because the sampled packets have at least one malicious packet, the capture-failure rate is set as 0. In addition, if the ith flow does not pass the jth switch, then the sampling at the jth switch does not affect the capture-failure rate of the ith flow, and thus the capture-failure rate of the ith flow at the jth switch is 1.

Next, to relax the above assumption for natural numbers, the gamma function is used, which has the following relationship with the factorial:

$$\Gamma(n) = (n-1)!, \tag{6}$$

where the gamma function is defined by $\Gamma(t) = \int_0^\infty x^{t-1} \cdot e^{-x} dx$. Using (6), the capture-failure rate in (5) is converted to

$$p_{i,j} = \begin{cases} 0 & \text{if } b_j - \lambda_i \leq x_j \cdot b_j, \\ \frac{\Gamma(b_j - \lambda_i + 1) \cdot \Gamma(b_j - x_j \cdot b_j + 1)}{\Gamma(b_j + 1) \cdot \Gamma(b_j - \lambda_i - x_j \cdot b_j + 1)} & \text{otherwise.} \end{cases} \tag{7}$$

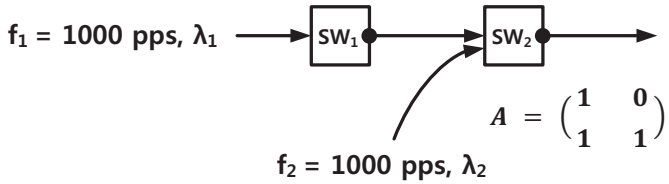
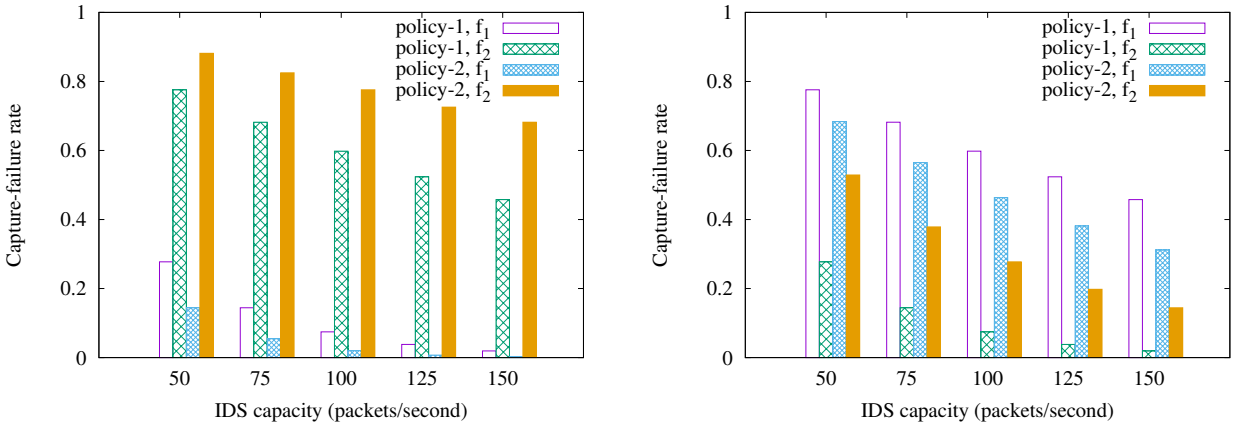


Fig. 2. Simple network topology and path information matrix.



(a) Case 1: $\lambda_1 = 50$ pps, $\lambda_2 = 10$ pps

(b) Case 2: $\lambda_1 = 10$ pps, $\lambda_2 = 50$ pps

Fig. 3. Capture-failure rates with respect to the IDS capacity.

Consider a simple network consisting of two switches (denoted by SW_1 and SW_2) and two flows (denoted by f_1 and f_2) as shown in Figure 2. The sending rates of both flows are 1,000 pps, and the flows have malicious rates of λ_1 and λ_2, respectively. Matrix $\boldsymbol{A}$ shows the routing path of each flow. In the network topology, f_1 passes both switches, and f_2 passes only SW_2. Let C denote the capacity of the IDS. Then, we can consider two possible sampling policies: 1) sampling at only SW_2 with a rate C and 2) sampling at both switches with a same rate $C/2$. Namely, for the policy 1, only SW_2 samples the traffic at the IDS capacity. On the other hand, for the policy 2, each switch samples the traffic evenly with one half of the IDS capacity.

Through MATLAB simulations, we measure the capture-failure rate while the IDS capacity is varied from 50 to 150 pps. Figure 3(a) shows the capture-failure rate when $\lambda_1 = 50$ pps and $\lambda_2 = 10$ pps. In Figure 3(a), it is seen that the capture-failure rates decrease as the IDS capacity increases and that f_2 experiences the higher capture-failure rate within the entire range of the IDS capacity. Between two policies, the sampling policy 1 must be the better sampling strategy because f_2 has lower capture-failure rate under the policy 1. Figure 3(b) shows the

capture-failure rate when $\lambda_1 = 10$ pps and $\lambda_2 = 50$ pps. The sampling policy 2 achieves the better performance because f_1 has the higher capture-failure rate than f_2 within the entire range and the capture-failure rate of f_1 is smaller under the sampling policy 2.

These simulation results imply that the sampling rates should be carefully determined for a better inspection according to the network status, such as the IDS capacity, the rate of malicious flows, and flow paths.

IV. Proposed Algorithm

In this section, we describe our proposed sampling algorithm for intrusion detection in SDN-enabled network. In a conventional network, an IDS inspects the traffic flows at only a few specific switches or links owing to the physical limitation of network device access. However, in an SDN-based network, it is possible to sample the traffic at any location by mirroring duplicated traffic to the IDS using an SDN controller. Although an IDS used in SDN can inspect packets at any point within the network, it is impossible to analyze all packets because of the limited IDS capacity. The traffic volume in a network is generally much larger than the capacity of an IDS. Therefore, instead of analyzing all of the packets, we sample packets at the switches at a certain sampling rate, which is determined through an optimization process that minimizes the capture-failure rate of the malicious flow with the largest capture-failure rate.

A. Sampling Rate Decision

Because the sampling approach does not sample all the packets in the network, a certain possibility that malicious packets will not be sampled exists. Therefore, the performance of the intrusion detection depends highly on how much malicious traffic is sampled and forwarded to the IDS. In other words, the objective of the sampling method is to minimize the capture-failure rates of malicious traffic. Under the assumption that every malicious packet forwarded to an IDS is properly classified, an optimization problem minimizing the capture-failure rate can be formulated as follows:

$$\begin{aligned} M(\boldsymbol{x}) &= \min_{\mathbf{x}}\{\max_{i} p_i\} \\ &= \min_{\mathbf{x}}\{\max_{i}(\prod_{j} p_{i,j})\} \\ &= \min_{\mathbf{x}}\{\max_{i}(\prod_{j} \frac{\Gamma(b_j - \lambda_i + 1)\cdot\Gamma(b_j - x_j\cdot b_j + 1)}{\Gamma(b_j+1)\cdot\Gamma(b_j - \lambda_i - x_j\cdot b_j + 1)})\}. \end{aligned} \tag{8}$$

A different objective function, such as a summation of the capture-failure rates over all of the flows, rather than the use of the largest capture-failure rate of all flows, may also be considered. However, the solution to this objective function may reduce only the summation of capture-failure rates by scarifying the detection performance for a few flows with a large capture-failure rate. In practice, however, this could result in a very crucial and dangerous situation in terms of security. Alternatively, a weighted summation of the capture-failure rates using the coefficients related to malicious traffic volumes can be used. Further research on different types of objective functions and an evaluation of their performance for specific security applications remains as our future work.

In addition to the objective function in (8), some constraints need to be taken into consideration. First, owing to the limited IDS capacity, the total amount of sampled traffic cannot exceed the inspection capacity of the IDS. Let C denote the capacity of the IDS. The constraint on the relationship between C and the sampling rate vector $\boldsymbol{x}$ is then given by

$$\sum_{i}^{n} x_i \cdot b_i \leq C. \tag{9}$$

The inspection capacity of an IDS can be defined simply as the maximum amount of traffic that the IDS can properly process without a significant degradation in the inspection performance. Second, the sampling rate at each switch is a positive value between 0 and 1.

$$0 \leq x_i \leq 1 \quad \text{for} \quad i = 1, 2, \cdots, n. \tag{10}$$

The optimization problem is to minimize the object function in (8) with the constraints of (9) and (10). This can be readily solved using optimization algorithms such as interior-point optimization or sequential quadratic programming.

B. Update Strategy

In the optimization of (8), the objective function is given as a function of $\boldsymbol{b}$, $\boldsymbol{\lambda}$, and $\boldsymbol{x}$, where $\boldsymbol{b}$ is the data rate vector of the switches, $\boldsymbol{\lambda}$ is the malicious traffic rate of the flows, and $\boldsymbol{x}$ is the sampling rate at the switches. Whereas the data rates of the switches $\boldsymbol{b}$ can be easily gathered using an SDN controller, the malicious rate $\boldsymbol{\lambda}$ continuously changes over time and is an unknown parameter to be estimated. Once the traffic sampling is conducted at a certain rate of $\boldsymbol{\lambda}$, the IDS inspects the sampled traffic and provides security reports containing statistics on the security attacks along with the source and destination IP addresses and types of attacks. These reports

enable the malicious traffic rate to be estimated, and the details are explained later. Based on the estimate of $\boldsymbol{\lambda}$, the optimization provides the next appropriate sampling rates at the switches. During these iterations, the estimated malicious rates converge to the actual values through a re-adjustment of the sampling rates.

Algorithm 1 Proposed sampling rate decision strategy

Require: n, f, C, $\boldsymbol{s}$, $\boldsymbol{A}$, η, k

1: $\boldsymbol{b} \leftarrow \boldsymbol{s} \cdot \boldsymbol{A}$
2: Set $\lambda_i \leftarrow \eta$ for $\forall i$ and $\boldsymbol{\lambda} \leftarrow \{\lambda_1, \lambda_2, \cdots, \lambda_f\}$
3: Find the solution $\boldsymbol{x}$ for the optimization with $M(\boldsymbol{x})$
4: **loop**
5: Configure OF-flow tables to sample traffic according to $\boldsymbol{x}$
6: Obtain the malicious traffic amount $\boldsymbol{r}$ from IDS inspection results
7: Estimate instantaneous malicious rate $\hat{\lambda}$
8: $\tilde{\lambda} \leftarrow (1-\omega)\tilde{\lambda} + \omega\hat{\lambda}_i$ for a small ω
9: Find a new sampling rate $\boldsymbol{x}$ by $M(\boldsymbol{x})$ with $\tilde{\lambda}$ in (8) subject to the constraints in (9) and (10)
10: **end loop**

Algorithm 1 shows the pseudocode of the proposed traffic sampling scheme. The initial malicious traffic rate of all the flows are set to a constant η on line 2. Then, the initial sampling rate $\boldsymbol{x}$ can be obtained by solving (8) on line 3. Once the sampling is performed and the sampled traffic is forwarded to an IDS, then the IDS inspects the traffic and classifies the malicious traffic. Let $\boldsymbol{r}$ denote the measured malicious traffic amount vector, and its element r_i represents the amount of detected malicious packets of the ith flow, which are obtained by the IDS inspection report on line 6.

After that, by using $\boldsymbol{r}$, $\boldsymbol{s}$, $\boldsymbol{x}$, and $\boldsymbol{b}$, the instantaneous malicious traffic rate of each flow $\hat{\lambda}_i$ can be estimated. First, we calculate how many data packets are forwarded to the IDS for each flow. Although the number of detected malicious packets of the ith flow is smaller than in the other flows, it does not necessarily mean the malicious rate of the ith flow is lower than that of the other flows because the total number of sampled packets is not the same. Let $\boldsymbol{y}$ denote the aggregated sampled traffic amount vector, and its element y_i is the total number of traffic

packets of the ith flow sampled at every switch. Then, y_i can be calculated as follows:

$$y_i = \sum_{j \in \alpha(i)} s_i \cdot x_j, \quad i = 1, \cdots, f. \tag{11}$$

Then, the instantaneous malicious rate is obtained as follows:

$$\hat{\lambda}_i = \frac{r_i}{y_i} \cdot s_i. \tag{12}$$

Note that $\left(\frac{r_i}{y_i}\right)$ is the ratio of the number of malicious packets identified at the IDS to the number of the all the sampling packets belonging to the ith flow.

Because the estimated instantaneous malicious traffic rate $\hat{\lambda}_i$ may fluctuate over time, especially when y_i is small, the average malicious traffic rate is obtained by a simple moving average on line 8. As shown in Alg. 1, the average malicious traffic rate vector $\tilde{\lambda}$ is used in (8) instead of the instantaneous value of (12). By repeating the operations in the loop from lines 5 through 9 in Alg. 1 over time, the estimated value of the malicious vector will catch up with the actual value.

C. Traffic Sampling Mechanism

For a sampling rate $\boldsymbol{x}$ obtained by Algorithm 1, each switch samples traffic at the given rate and forwards the sampled traffic to IDS. The packet sampling mechanism has a significant impact on malicious traffic detection performance of IDS. One may consider packet-driven and time-driven mechanisms. Under packet-driven mechanism, the switch captures one packet in every $1/x$ packets if the sampling rate is x. On the other hand, under time-driven mechanism, it captures all the packets during a portion of x at each sampling interval. Attacks on stateless protocols can be easily identified using either packet-driven or time-driven mechanism. However, the time-driven mechanisms has advantage of detecting stateful attacks because it captures all the packets for a certain time duration. For example, if the sampling rate x is 0.1 and the sampling interval 1 second, then the switch captures all the packets for the first 100 ms in every 1 second. It is more likely that the time-driven mechanism can provide a higher degree of visibility on complete traffic flows when the amount of traffic samples is limited. In Section VI, we will discuss in detail how we implemented the sampling mechanism on SDN-enabled switches.

V. Simulation Results

To evaluate the performance of the proposed sampling strategy, we conducted extensive simulations using an ns-2 network simulator [13] and compared its performance against that

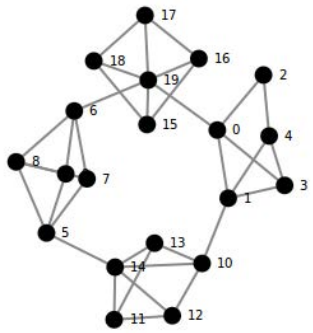


Fig. 4. Scenario 1: A simple network topology with 20 switches.

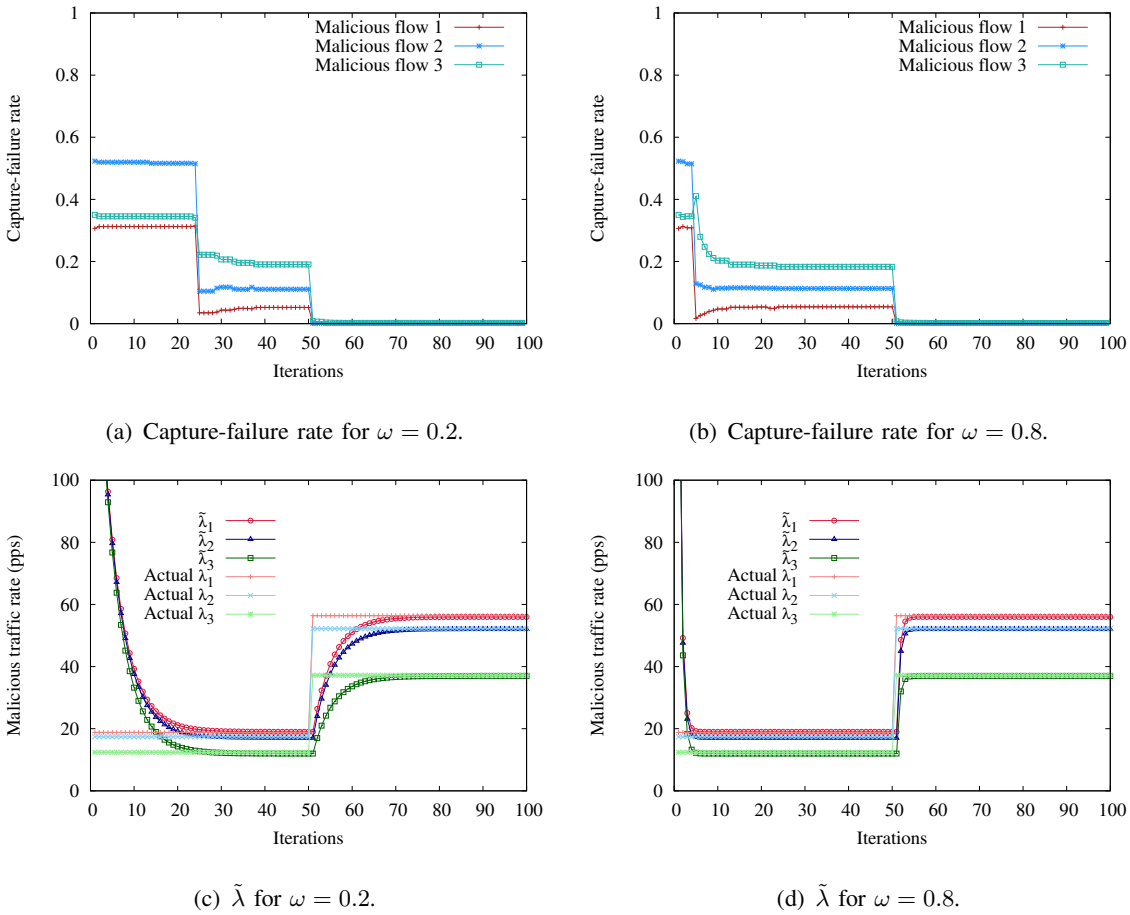
of a naive sampling method. Naive sampling equally samples data traffic at every switch under the condition that the aggregated amount of sampled traffic is equal to the IDS capacity. The maximum IDS capacity is set at 3 Gb/s, and thus the total rate of sampled data traffic is kept below 3 Gb/s. We evaluated the performance of the proposed method for three different topologies. The optimization of $M(\boldsymbol{x})$ in Algorithm 1 is solved using the `fmincon` function of MATLAB.

A. Scenario 1: Simple Network (20 switches)

We conducted a set of simulations for a simple network, as shown in Figure 4. There are 20 switches in the network topology, as indicated by the black spots in the figure. The bandwidth of each link is set to 1 Gb/s. In this case, we set up 100 data flows through the network, and each source node is configured to generate data packets at a constant bitrate (CBR) of 12 – 256 Mb/s. The packets are transferred over UDP. Additionally, we set three malicious traffic flows, including both normal and malicious packets.

Figure 5 shows the detection performance of the proposed algorithm for all malicious traffic as the number of iterations increases from 1 to 100. Each iteration corresponds to 2 seconds, but it is adjustable depending on the network size and IDS performance. The traffic flows including malicious packets change their malicious traffic ratio λ from 1% to 3% at the 50th iteration (i.e., t = 100 s). The proposed sampling scheme updates its solution to the sampling rate using the estimation of malicious traffic rates at each iteration. The IDS capacity is set to 0.5 Gb/s.

In Figures 5(a) and (b), the graphs clearly show that the maximum capture-failure rate gradually decreases as the number of iterations increases. At the first iteration, the maximum of the capture-failure rates of the malicious traffic is greater than 0.5, the main reason for which is



(a) Capture-failure rate for $\omega = 0.2$.

(b) Capture-failure rate for $\omega = 0.8$.

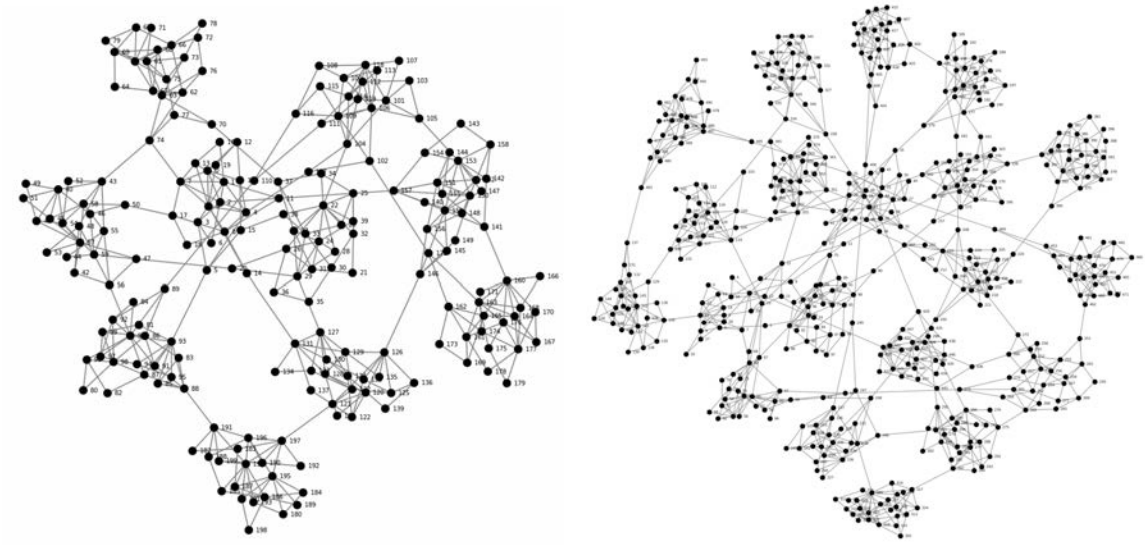
(c) $\tilde{\lambda}$ for $\omega = 0.2$.

(d) $\tilde{\lambda}$ for $\omega = 0.8$.

Fig. 5. Detection performance for the topology shown in Figure 4 when the malicious traffic ratio λ changes from 1 to 3% at the 50th iteration (1 iteration = 2 seconds).

the inaccuracy of the initial malicious rates. Because information regarding a malicious traffic rate is unavailable, the intrusion detection performance during the first tens or so iterations depends highly on the initial values of the malicious traffic rates. As the inspection progresses, the maximum value gradually decreases based on the updating strategy of the proposed algorithm.

Comparing Figures 5(a) and (b), the capture-failure rate decreases more rapidly in Figure 5(b) because the moving average factor ω used in Algorithm 1 for the simulation in Figure 5(b) is greater than that for Figure 5(a). When the malicious traffic ratio λ abruptly changes at the 50th iteration, the proposed algorithm successfully adjusts the sampling rates of switches and keeps the capture-failure rate near zero. Interestingly, some points exist where the capture-failure rate of a certain flow increases in comparison with that of the previous iteration (e.g., traffic flow 1 in Figure 5(b) at the 5 – 10th iteration). However, this is due to the sampling policy attempting



(a) Network topology with 200 switches.

(b) Network topology with 500 switches.

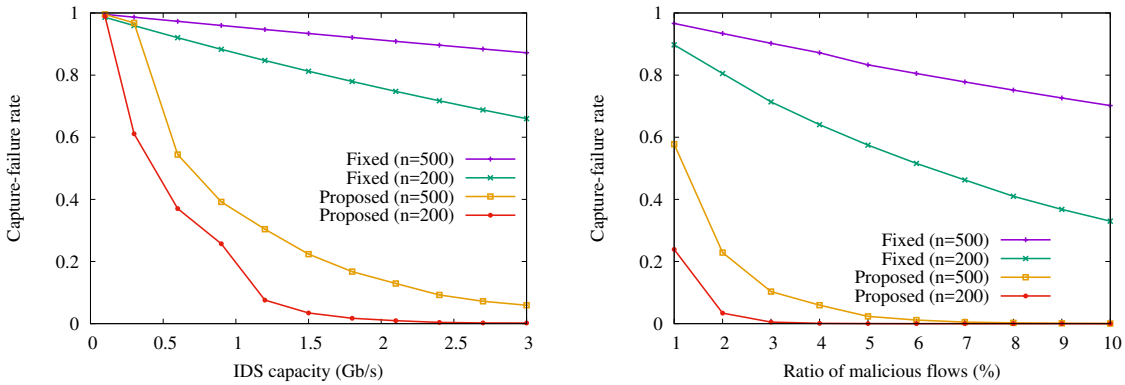
Fig. 6. Scenario 2: Network topologies for larger-scaled scenario.

to reduce the maximum value among the capture-failure rates, which is continuously decreased as the iteration increases.

In Figure 5(c) and (d), we observed that the estimated malicious traffic rates reach closer to the actual values through a re-adjustment of the sampling rates as the iteration increases. As discussed above, if ω is greater, the estimate of λ approaches the actual value more rapidly. However, if ω is too large in real network environments, the estimation is more susceptive to noises in instantaneous values, and it may fluctuate severely with instability. Therefore, ω should be carefully determined according the characteristics of network environment.

B. Scenario 2: Larger-Scaled Network (200 and 500 switches)

We conducted our performance evaluation for larger sized networks. Figure 6(a) and (b) show the topology with 200 and 500 switches, respectively. Using these topologies, it was possible to conduct a more realistic and complex simulation closer to a real network scenario. Each link has a capacity of 1 Gb/s. For each topology, 1,000 and 1,500 flows are generated between randomly chosen source and destination nodes, respectively. Each source node is configured to generate data packets at a CBR of 12 – 256 Mb/s. These packets are to be delivered over UDP. Three flows are selected as malicious traffic, which includes both normal and malicious packets, of which 1 to 10% are malicious packets. Under this scenario, the naive sampling approach was



(a) Ratio of malicious traffic: 2.5 percent

(b) IDS capacity: 2 Gb/s

Fig. 7. Capture-failure rates for the topologies shown in Fig. 6.

implemented for a comparison. The naive scheme samples the traffic equally at a rate of (C/n) for all n switches.

Figure 7(a) shows the capture-failure rate with respect to the IDS capacity for $\boldsymbol{\lambda}$ = 0.025. The IDS capacity varies from 0.1 to 3 Gb/s. For both algorithms, the change in the capture-failure rate shows a downward trend as the IDS capacity increases. This result is reasonable in that a larger IDS capacity means that an IDS can conduct more inspections. For all cases, the proposed sampling strategy significantly outperforms the naive sampling approach. In particular, for a naive approach, more than 85% of the malicious flows are undetected in an IDS for the network topology with 500 switches. This means that naive sampling is significantly ineffective in large-scaled networks because the IDS capacity is quite limited in comparison with the volume of network traffic.

Figure 7(b) also shows that our proposed scheme outperforms the naive approach for a fixed IDS capacity of 2 Gb/s and a varying ratio of malicious packets of 1 to 10%. The graph shows that the capture-failure rate depends on the ratio of malicious traffic. In fact, the IDS can find malicious flows more easily when the malicious traffic ratio is increasing because the increase in the ratio of malicious traffic leads to an increase in the amount of malicious packets, whereas the amount of total traffic in the network is fixed in this simulation. Therefore, the probability of sampling malicious packets is also increasing, resulting in lower capture-failure rates, as shown in Figure 7(b).

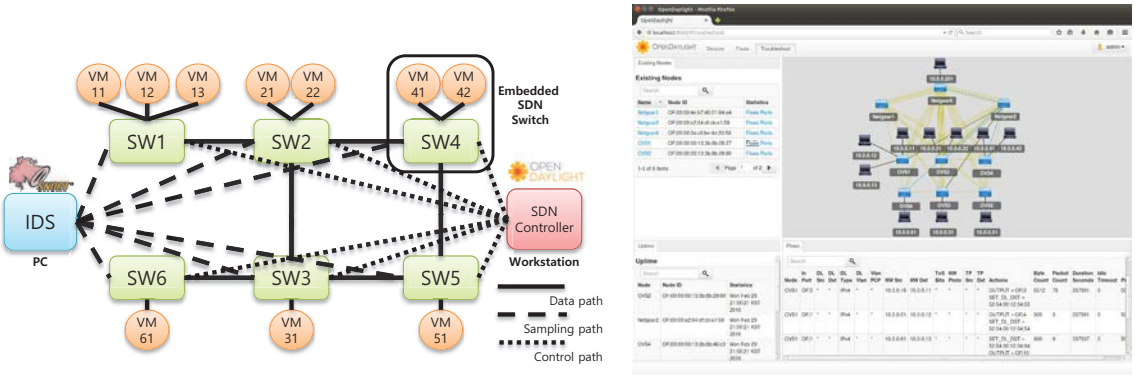
In our simulations, the computation time for solving the optimization in Algorithm 1 varies depending on the topology size and the number of flows. For the simple topology in Figure 4, it

took about 4 ms, and for the large-scaled topologies, it took about 3.6 s and 8.79 s, respectively, on HP Z620 workstation equipped with Intel Xeon E5-1650v2 CPU. Because the computational complexity significantly increases with respect to the network scale, it is highly required to develop an efficient method to solve the optimization. As part of our future work, it would be possible to incrementally solve the optimization under the assumption that the traffic flows change more slowly than the sampling frequency.

VI. Experiment Results

For empirical performance evaluation, we constructed an SDN testbed and implemented the proposed algorithm on the testbed. The testbed consists of a HP Z420 workstation, which is equipped with Intel Xeon E5-1620 and runs Ubuntu 14.04.2 LTS, as an SDN controller, 6 Odroid-XU3 embedded boards as SDN switches, and a Dell Optiplex Desktop PC for IDS. The details of the testbed are as follows:

- **SDN controller:** There exist a variety of SDN controller packages such as FloodLight, ONOS, and OpenDayLight (ODL) [14]. Among them, we chose ODL as the SDN controller for our testbed and installed it on the workstation. ODL is a Java-written open-source project software. It supports the OpenFlow protocol and provides the northbound API (REST API) and southbound API (OSGi) for programming interface.
- **SDN switches and traffic sources:** We constructed an SDN with SDN-enabled switches. Odroid-XU3 embedded boards are used as an SDN switch by installing Open vSwitch (OVS) software [15]. Because each board is equipped with a single Ethernet port, a USB hub and USB-to-Ethernet adapters are added to have four more Ethernet ports. One is connected to the SDN controller, another is used for mirroring the sampled traffic, and the others are for constructing a data network topology. In addition, the embedded board is virtualized by Kernel-based Virtual Machine (KVM) [16] hypervisor, and it hosts a couple of virtual machines (VMs). The VMs are used to generate normal and malicious traffic flows using a network bandwidth measurement tool, iPerf [17].
- **IDS:** Snort [12] and Suricata [18] are ones of the most popular open-source IDSs. In our testbed, Snort was used to detect malicious traffic generated by the iPerf on the VMs. The attackers are assumed to perform a port-scan attack using a port number 10,000. The rule set for Snort is configured to trigger an alarm if a packet is destined to an unallowed port of the destination node.



(a) Network topology. (b) GUI of OpenDayLight.

Fig. 8. SDN-based testbed configuration.

As mentioned in Sec. IV-C, traffic sampling mechanism can be implemented as either packet-driven or time-driven mechanism. In our testbed, we exploited a port mirroring mechanism of OVS software. We created a virtual queue and attached it to one port of OVS switch. Then, the traffic to be sampled is mirrored to the virtual queue, and is forwarded to the IDS through the port. By adjusting a quality of service (QoS) policy of the queue (i.e., max-rate), the sampling rate of the switch is controlled in our implementation.

Figure 8 shows the SDN testbed with the SDN controller, 6 SDN-enabled switches, 10 VMs, and a single IDS. The number of flows generated by VMs are five, and their flow rates are 20 Mb/s. One of five flows is configured to include 5% malicious packets from t = 10 s. The IDS capacity is set to 10 Mb/s, and the SDN controller blocks malicious flows if it receives more than 400 alarms from the IDS. For the proposed algorithm, the moving average parameter ω is set to 0.2, and the sampling policy is updated in every 2 seconds.

Figure 9 shows the experiment results. At t = 10 s, the attack begins at a rate of 1 Mb/s. At t = 17 s, the IDS detects malicious packets and generates the first alarm. The reason for this time elapse is two-folds. First, it takes time for the switches to capture packets at a certain rate and forward the packets to the IDS. Note that the switches in our testbed are a software-based switch running OVS software. Second, the IDS consumes a certain amount of time to analyze the captured packet and update its log file. After the first alarm, the proposed algorithm calculates a new sampling rate for each switch, and updates the sampling rate at every switch. These two processes take about 3–5 seconds. In Figure 9, it is seen that the number of alarms for the proposed algorithm rapidly increases at t = 22 s. At t = 47 s, the accumulated number of alarms

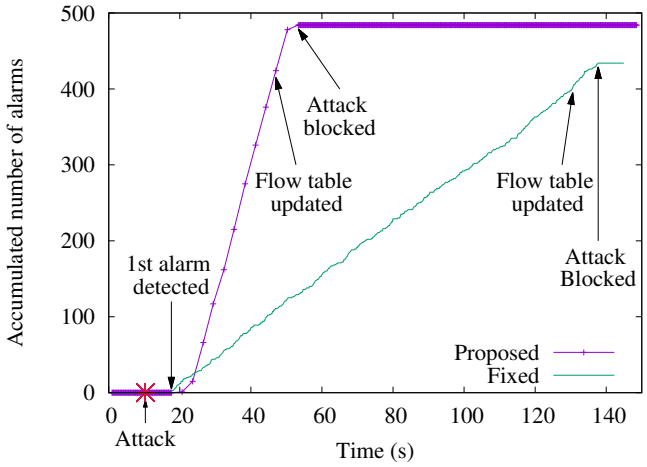


Fig. 9. Experimental results for intrusion detection and flow blocking.

exceeds 400, and the SDN controller sends an updated flow table to the switch. At t = 53 s, the flow is eventually blocked. For the proposed algorithm, it takes 43 seconds from the beginning of attack to the flow blocking. On the contrary, it takes 127 seconds for the naive sampling. Note that the time elapse for the proposed algorithm highly depends on the performance of the IDS and OVS switches, and it could be further reduced in enterprise-scale environments.

VII. Conclusion

In this paper, we proposed a traffic sampling rate decision strategy for efficiently exploiting limited IDS resources in the detection of malicious traffic. The proposed method increases the intrusion detection performance considerably by estimating the appropriate sampling rate for each switch and focusing more on suspicious traffic. With the help of SDN technology, our method measures the throughput of each switch, the current distribution of malicious traffic, and the flow path information. The proposed approach evaluates the rate of missing malicious traffic for selecting the most appropriate sampling rates, while the sampled traffic volume is kept below the capacity that can be handled by an IDS. The simulation and experiment results indicate that the proposed sampling scheme significantly outperforms a naive scheme that samples the traffic from all switches at an equal rate.

ACKNOWLEDGMENT

This work was supported in part by the NRF funded by MSIP of the Korea government (2014R1A2A2A01006002), and by the ICT R&D program of MSIP and IITP of the Korea government (14-824-09-013, Resilient CPS Research).

REFERENCES

[1] C. Jeong, T. Ha, J. Narantuya, H. Lim, and J. Kim, "Scalable network intrusion detection on virtual SDN environment," in *IEEE International Conference on Cloud Networking (CloudNet 2014)*, October 2014, pp. 264–265.

[2] S. Shin and G. Gu, "Cloudwatcher: Network security monitoring using OpenFlow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?)," in *IEEE International Conference on Network Protocols (ICNP)*, October 2012, pp. 1–6.

[3] G. Androulidakis and S. Papavassiliou, "Improving network anomaly detection via selective flow-based sampling," *Institution of Engineering and Technology (IET)*, vol. 2, no. 3, pp. 399–409, April 2008.

[4] "SDN specification," http://www.opennetworking.org/sdn-resources/sdn-library/whitepapers/.

[5] S. Agarwal, M. Kodialam, and T. Lakshman, "Traffic engineering in software defined networks," in *IEEE International Conference on Computer Communications (INFOCOM)*, April 2013, pp. 2211–2219.

[6] R. Kawahara, T. Mori, N. Kamiyama, S. Harada, and S. Asano, "A study on detecting network anomalies using sampled flow statistics," in *IEEE/IPSJ International Symposium on Applications and the Internet (SAINT)*, January 2007, pp. 81–81.

[7] 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: The International Journal of Computer and Telecommunications Networking*, vol. 62, pp. 122–136, April 2014.

[8] J. Mai, A. Sridharan, C.-N. Chuah, H. Zang, and T. Ye, "Impact of packet sampling on portscan detection," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 12, pp. 2285–2298, December 2006.

[9] C. C. Kacha, K. A. Shevade, and K. S. Raghuwanshi, "Improved Snort intrusion detection system using modified pattern matching technique," *International Journal of Emerging Technology and Advanced Engineerings (IJETAE)*, vol. 3, no. 7, pp. 81–88, July 2013.

[10] A. Khurshid, W. Zhou, M. Caesar, and P. B. Godfrey, "Veriflow: Verifying network-wide invariants in real time," in *ACM SIGCOMM Workshop on Hot topics in software defined networks (HotSDN)*, August 2012, pp. 49–54.

[11] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "OpenFlow random host mutation: Transparent moving target defense using software defined networking," in *ACM SIGCOMM Workshop on Hot topics in software defined networks (HotSDN)*, August 2012, pp. 127–132.

[12] "Snort," http://www.snort.org/.

[13] "The network simulator ns-2," http://www.isi.edu/nsnam/ns/.

[14] "OpenDayLight (ODL)," http://www.opendaylight.org/.

[15] "Open vSwitch (OVS)," http://openvswitch.org/.

[16] "Kernel-based virtual machine (KVM)," http://www.linux-kvm.org/.

[17] "iperf," https://iperf.fr/.

[18] "Suricata," http://suricata-ids.org/.