Suspicious Flow Forwarding for Multiple Intrusion Detection Systems on Software-Defined Networks

Taejin Ha, Seunghyun Yoon, Aris Cahyadi Risdianto, JongWon Kim, and Hyuk Lim

***Abstract*— In recent years, there have been an increasing number of attacks on networks, such as the distributed denial-of-service (DDoS) attack. However, the traditional network is not sufficiently flexible to control the huge amount of traffic that now passes through an intrusion detection system (IDS). With software-defined networking (SDN), which separates control planes and data planes for programmability, elasticity, and simplicity, it becomes possible to force traffic to pass through an IDS by simply re-routing or mirroring traffic to an IDS. This article focuses on how to distribute traffic to multiple IDSs in order to increase the detection of network attacks and balance IDS loads. A clustering-based flow grouping scheme that distributes flows according to routing information and flow data rate is proposed. Through experiments with a virtualized testbed, we show that the proposed scheme detects network attacks more quickly and achieves a better balance of traffic loads on the IDSs.**

I. Introduction

IN the last few decades, as information and communication technology (ICT) has been developed rapidly, daily activities in today's society have been increasingly accomplished using the Internet. As a result, the amount of network traffic has increased and the scale of network infrastructure has expanded enormously. As ICT becomes an essential part of modern life, many types of attacks against networks have been used, including the denial of service (DoS), man-in-the-middle attacks, sniffer attacks, and malware. In order to provide security against these network-based attacks, intrusion detection systems (IDSs) are widely used. An IDS is a device that protects networks from various types of malicious attacks. This system monitors network traffic and inspects data packets to determine if the packets contain malicious attacks. If an IDS finds suspicious activities, it generates an alarm and reports the possibility of an ongoing network attack. An IDS is connected to a certain host or link. This means that an IDS can inspect traffic only from the connected source. Therefore, the intrusion detection performance in a conventional network depends on where the IDS is located. However, because the scale of the network and the amount of its traffic have increased exponentially, the number of IDSs deployed in a network should be also increased proportionally.

In order to manage multiple IDSs efficiently, software-defined networking (SDN) technology can be used. SDN is an emerging network architecture that separates the network control plane from the data plane. It has a centralized controller, called the SDN controller, which is responsible for all the network control decisions of the network-wide distributed forwarding elements. OpenFlow is a communication protocol standard between the SDN control plane and the data plane of forwarding switches in the SDN. With SDN technology, it is possible to easily check the network status and to forward certain flows to a specific node.

In this article, the performance of an SDN-based network with multiple IDSs is evaluated in terms of its ability to inspect the transfer of malicious data. With SDN technology, suspicious flows can be forwarded to specific IDS. If the flows from the same attack are forwarded to the same IDS, it is intuitively expected to achieve better inspection of the attack. Based on this, a flow grouping scheme that determines which flows should be forwarded to which IDSs is proposed to achieve the best intrusion detection performance.

II. Related Work

A. DDoS Attack Detection

In general, there are three ways for an IDS to detect a distributed denial-of-service (DDoS) flooding attack: source-based, destination-based, and network-based detection [1]. The source-based method deploys an IDS close to the source. This can minimize resource waste associated with detecting and responding to attacks. However, it is difficult to detect the attack when the attacking sources are distributed. In contrast, the destination-based method places an IDS close to the victims. Though it is much easier to detect the attack, it is an inefficient use of the network resources. Network-based methods deploy an IDS in the intermediate networks. This represents a compromise between the other two methods.

There exists research on DDoS attack detection that uses the characteristics of the attack. Androulidakis *et al*. [2] proposed an intrusion detection technique that samples packets according to the size of the flows. Given that smaller flows are usually the source of network attacks, the main targets of inspection are flows whose size is smaller than a certain threshold. Kawahara *et al*. [3] proposed a flow statistic based strategy. Many network attacks cause an increasing number of flows when they attack. However, the sampling approach often fails to find this increment. This problem was solved by partitioning sampled traffic into groups using a source autonomous system. Giotis *et al*. [4] used the change in entropy to detect a network attack. For instance, a significant entropy drop in the destination internet protocol (IP) address and destination port indicates the possibility that the network is under a DDoS attack.

The authors are with the Gwangju Institute of Science and Technology (GIST), Republic of Korea.

B. SDN-based Intrusion Detection

Because of the inherent scalability and programmability of a network, SDN can be used for security enhancement. NetFuse [5] is a mechanism for cloud and data center environments to protect themselves against traffic overloads from network attacks, operator errors, or routing misconfigurations. It uses both passive listening and adaptive active queries to monitor network status effectively. CloudWatcher [6] uses an SDN to build a framework to efficiently monitor services in large and dynamic cloud networks. The framework enables the network administrators to protect their network easily by writing a simple policy script. Fayaz *et al.* [7] proposed the Bohatei DDoS defense system, which was built on an SDN and NFV (network function virtualization) platform. This system is scalable because its resource management algorithm controls the network in order to avoid control and data plane bottlenecks in the SDN based DDoS defense system. In addition, it exploits NFV capability to flexibly place the defense virtual machine (VM) resources in the locations where they are needed on the network. Mehdi *et al.* [8] considered how to exploit the SDN technology in order to effectively detect and mitigate the traffic anomaly in home and SOHO (small office / home office) networking environments. In their work, they implemented four traffic anomaly detection algorithms in an SDN testbed running a NOX controller and showed that efficient anomaly detection can be achieved without disturbing user data traffic. Yoon *et al.* [9] investigated how the SDN technology can enhance network security. They implemented various network security functions such as firewalls, IPS (intrusion prevention system), and IDS using available SDN features and then evaluated the feasibility and performances of the security functions in real SDN testbeds with OpenFlow-enabled switches. Chu *et al.* [10] proposed the DDoS defender implemented on the OpenFlow controller. Once the amount of traffic exceeds a predetermined threshold, a DDoS attack is identified and the rules for the controller change in order to drop the incoming packets. Braga *et al.* [11] proposed a DDoS flooding detection algorithm based on traffic patterns. This method classifies network traffic by using self-organizing maps. When flows are classified as malicious, the controller drops the flow. FlowRanger [12] is a solution that interrupts DDoS attacks using a priority-based scheme. The trust management module inspects incoming routing requests and determines whether they are malicious. If the requests are not trustworthy, then they are given a lower priority than regular requests. Therefore, the DDoS attack does not work smoothly. Wang *et al.* [13] proposed a graphic model based on an attack detection method that deals with a dataset shift problem. It saves known traffic patterns as a relational graph. If new traffic is generated, the system can determine whether the traffic is malicious by comparing the graphs.

However, most schemes considered networks with only a single IDS, and they do not consider how a huge volume of suspicious flow traffic can be handled for enhancing intrusion detection performance with multiple IDSs. Some approaches with multiple IDSs such as [7] mainly focused on load-balancing among the IDSs.

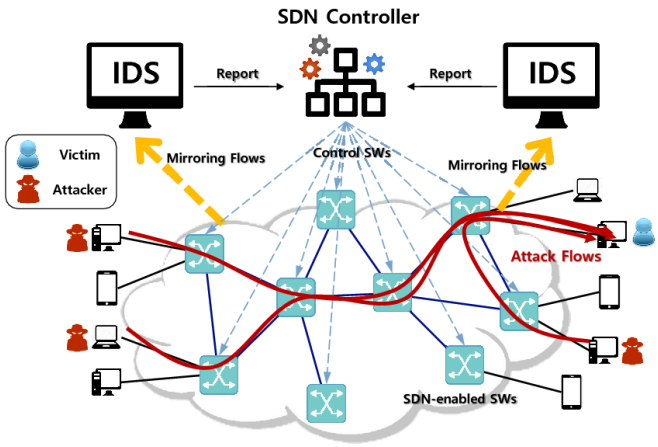


Fig. 1. Suspicious traffic inspection architecture with multiple IDSs on an SDN-based network.

III. Proposed Traffic Forwarding Algorithm

A. Motivation

Consider a network with SDN-enabled switches and multiple IDSs as shown in Figure 1. In SDN-based network intrusion methods, unlike in conventional networks that can only inspect traffic directly passing through an IDS, flows are forwarded to one of the IDSs using SDN technology. This makes the analysis of malicious incoming packets more flexible. There are new challenges associated with forwarding flows properly and balancing loads between IDSs in SDN-based intrusion detection methods.

When detecting DDoS attacks, many intrusion detection algorithms use a simple rule to check that the number of packets going to a specific destination does not exceed a certain threshold within a specific unit of time. If malicious traffic flows belonging to the same attack are assigned to the same group and forwarded to the same IDS, it is expected to achieve a higher probability of detecting the attack since it is easier to exceed the threshold of the pre-defined rule. Unfortunately, it is impossible to judge whether the flow is malicious or not before inspection. Instead, flow path information can be exploited to distinguish the flows. Most network-based attacks generate a huge number of flows when the attacks occur. Usually the attacks target only a few victims. That is, a large number of attacking senders generate malicious flows and send them to a small number of victims. These malicious flows destined for the victim gradually converge on the network and eventually meet at the routers close to the victim. As a result, the paths of the malicious flows have a common portion of paths. Even in the case where there are no common routers on the paths, the flows originating from or targeting the same host would have a topological proximity to each other. This observation motivates the system to make groups of flows according to their similarity and to forward them to the same IDS. This grouping does not guarantee that all the flows belonging to the same attack are forwarded to the same IDS; however, these flows are more likely to be forwarded together to the same IDS under the grouping algorithm. Another benefit of flow-grouping is the load-balancing of the IDSs. In conventional networks, because

the IDS is fixed to a certain link or node, if network traffic is not uniformly generated, then the IDSs are utilized unequally. With an SDN, flows can be forwarded to any IDS in the network, resulting in balanced IDS loads.

The motivations for the proposed algorithm can be summarized as follows:

- Fast Intrusion Detection: There are a number of IDSs in the present day networks and each IDS, in general, detects intrusions by matching packet patterns to known attack patterns. In order to notice the existence of attacks, the number of detected malicious packets within a certain time satisfies a pre-defined set of rules. Thus, better inspection performance is obtained if malicious flows belonging to the same attack are forwarded to the same IDS. The proposed algorithm finds the flows that have a close relationship in terms of routers on the path and groups them.
- IDS Load-balancing: Conventional networks place IDSs with specific nodes or links. As a result, IDSs can inspect the traffic of only those nodes or links. Since the utilizations of IDSs are dependent on their location, IDSs are not utilized in a balanced way. The proposed algorithm performs load-balancing when it groups the flows and assigns each group of flows to the IDSs.

B. System model

Consider an SDN-based network composed of k IDSs and n SDN-enabled switches. The inspection capacities of IDSs are denoted as the IDS capacity vector $C = [c_1, \cdots, c_k]^T$. Suppose there are f flows in the network and their data rates are represented with the vector $s = [s_1, \cdots, s_f]^T$. In the SDN-based network, the flow status of each SDN-enabled switch can be checked using the APIs of the SDN controller. The status includes flow information such as the MAC address of a source node and a destination node, the volume of packets and bytes, flow ID, priority, and so on. Since OpenFlow does not provide the flow rate information, the rate vector s should be indirectly estimated by periodically checking the per-flow counters such as 'byte count' and 'packet count' values, which are accessible by the standard SDN northbound interfaces (e.g., REST APIs).

Let $\mathbf{A}$ denote the flow routing matrix, which is an n-by-f binary matrix. For example, if flow j passes through switch i, then element a_{ij} in $\mathbf{A}$ is 1.

$$\mathbf{A} = \begin{pmatrix} a_{11} & \cdots & a_{1f} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nf} \end{pmatrix}, \ a_{ij} \in \{0,1\}. \quad (1)$$

The proposed algorithm categorizes flows into several groups based on the similarity between flow paths. Because the number of switches that each flow passes through is quite small in the context of the whole network, it seems that the flow paths do not have any correlation if they do not belong to the same attack. To compute the similarity, the proposed algorithm uses principal component analysis (PCA). PCA converts a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. PCA projects the observations into a new, uncorrelated space of smaller dimension. Using PCA, the dimension of flow routing

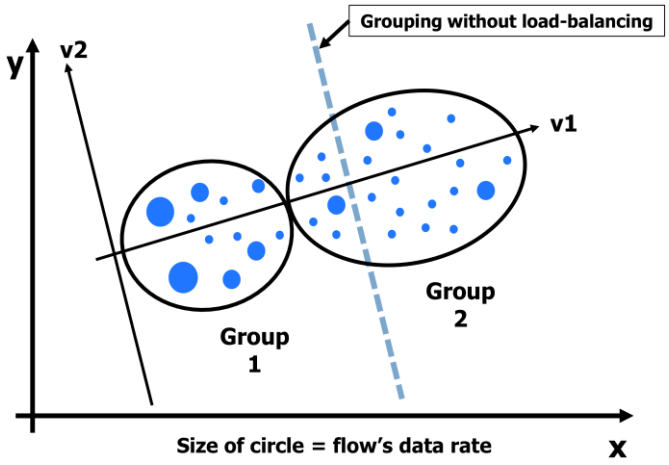


Fig. 2. Suspicious flow grouping using PCA.

matrix $\boldsymbol{A}$ is reduced to an m-by-f matrix (denoted by $\mathbf{A}'$), meaning that m PCs are used to make the new space. PCs used for dimension reduction are represented as $p = \{pc_1, pc_2, \ldots, pc_m\}^T$. While each column vector in $\mathbf{A}$ corresponds to the coordinate of the flow in the original n-dimensional space, the column vector in $\mathbf{A}'$ corresponds to the coordinate in the m-dimensional space. Then, the similarity between two flows is obtained by the Euclidean distance between the coordinates of the flows in the reduced and uncorrelated space. Figure 2 shows an example of the 2-dimensional space of $\mathbf{A}$. Two principal components $v1$ and $v2$ by PCA are depicted in Figure 2. If $m = 1$, the coordinates are projected into a single point on $v1$ axis. Then, the coordinate points can be clustered into two groups depending on the distance between the points in 1-dimensional space.

C. Flow grouping and load-balancing

The flows are clustered into g groups, and g is simply set to the number of IDSs, (i.e. $g = k$). Let $\boldsymbol{G} = [G_1, \cdots, G_k]^T$ and $\boldsymbol{B} = [b_1, \cdots, b_k]^T$ denote the set of groups and the set of data rates in the groups, respectively. The sum of data rates of flows in i-th group is given by b_i, which is obtained by $b_i = \sum_{j \in G_i} s_j$. The proposed algorithm uses the gravity based clustering [14] in order to group flows. Gravity based clustering exploits the density of points. First, the area of the region $\mathbf{A}'$ is measured. Because there are k IDSs, each group has an initial radius as follows:

$$r = \sqrt{\frac{\text{Area of region } \mathbf{A}'}{k \cdot \pi}}. \quad (2)$$

After calculating the group radius r, cluster header nodes are selected. For each node, the number of nodes that exist within r is counted. Then, the node with the largest number of neighbor nodes is chosen as the first cluster header. In order to choose the second cluster header, the counting process is repeated, excluding the first cluster header nodes and its neighbor nodes. Through this procedure, the properly distributed cluster header nodes are obtained, and each node belongs to the cluster with the closest cluster header.

Until now, each flow is assigned to one of k groups, and the data packets of each group's flows are forwarded to the same IDS. However, since this grouping does not consider the data

rate for each group, the traffic loads for each IDS are unbalanced. In this case, there is a possibility that the loads will exceed the capacity of the IDS. In order to prevent the overflow of packets in IDS, the flow grouping should take the total data rate of the groups into consideration. The idea of balancing the data rates is to adjust the group radius according to the group data rate. If a certain group has a larger group data rate, the group data rate can be reduced by reducing the group area. Accordingly, in order to increase the group data rate, the group area size can be enlarged. Therefore, the radius of each group is set to be inversely proportional to the group data rate. That is, the new group radius of the *i*-th group can be obtained as follows:

$$r_i' = r_i \cdot \frac{\sum_{j=1}^{k} b_j}{k \cdot b_i}. \quad (3)$$

This balancing algorithm is repeatedly performed until the ratio between the maximum and minimum group data rates is smaller than a certain threshold δ_g.

IV. Experimental Result

In order to verify the proposed algorithm, an SDN-enabled testbed was used for experiments. The testbed runs on a single workstation (specifications: Intel Xeon W3565 CPU and 48 GB RAM) and is composed of five different key parts: virtualization, tapping and mirroring, centralized control, traffic and attack generation, and intrusion detection.

A. Virtualization

In order to run the experiments, which involve a high number of nodes and functions (i.e., switches and hosts), a virtualization technique is deployed in the testbed. Linux Container (LXC) and Open vSwitch (OVS) are often used to virtualize a host machine and its network connectivity. LXC is a userspace interface for the Linux kernel containment features that creates an environment similar to a standard Linux installation without the need for a separate kernel. It can be used to deploy functions or services such as servers, clients, load balancers, software IDSs, and other network services including an SDN controller. OVS is a software-based multilayer network switch that virtualizes the network and provides multiple protocols and standards that are often used in networks. The network topology of the testbed can be created by connecting OVS switches via OVS patching and is managed centrally by using the SDN Controller. This virtualization-based testbed is more similar to a real physical network environment than conventional simulators or emulators because each container operates software and applications independently, and is connected to the OVS using its own virtualized network interfaces.

B. Tapping and Mirroring

One important operation for intrusion detection is how to capture suspicious network traffic from the network. One famous method is the 'tapping' mechanism, which exploits network TAPs (terminal access points) that can passively capture traffic on a network. In this experiment, virtual OVS ports are created for passive traffic tapping. With a simple TAP aggregation application, the packets can also be filtered, and the filtered packets can then be redirected to any port. The filtering process matches the packets with specific packet identifiers or parameters (i.e., IP addresses, protocol types, TCP/UDP ports, and VLAN ID) and then redirects those flows to specific ports. This mechanism is known as the flow tapping mechanism, and the captured traffic can easily be managed from the centralized SDN controllers.

In another mechanism known as 'port mirroring', packets are replicated based on their incoming or outgoing ports and can be sent to other ports. The SDN controllers can replicate traffic flows by performing additional 'OUTPUT' action to the flows to be mirrored. Then, the replicated packets can be sent directly to the packet processing tools (i.e., IDS, traffic analyzer, and network manager) or sent to another connected switch by updating the flow table of the switches. The original traffic and replicated traffic can be distinguished by assigning different VLAN tags on the network.

C. Centralized Control

As mentioned earlier, OVS-based network topology requires centralized control for network programmability. The SDN controller manages all requests and policies for switch interconnection and then translates them into forwarding rules (flow entries) in the switches. The controller periodically communicates with switches to collect the state of the flows or other statistical measurements. In this testbed, two SDN controllers are used to distinguish between real data traffic and replicated traffic. Each type of traffic is handled by a different controller. Open Network Operating System (ONOS) is used to deliver data traffic flows from senders to receivers. OpenDaylight is used to deliver tapped or replicated packets to the IDSs for traffic inspection.

D. Traffic and Attack Generation

An important aspect of this experiment is the generation of user traffic and attacker traffic with a realistic pattern (i.e. traffic paths, traffic rates, and burstiness) in the testbed environment. In addition, the generated traffic should be identified and controlled. For user traffic, the iperf program is used to simulate user traffic by specifying the sender and receiver with a specific traffic rate. Attack traffic is generated from an attacker node in several identified locations by using the Scapy [15] application, which is a python-based packet manipulation tool. In this experiment, TCP SYN DoS attacks using HTTP port (80) are performed. Each attacker VM runs multiple Scapy scripts to generate TCP request packets destined for the victim. The location of the traffic sender/receiver and attacker are randomly selected (it could originate from anywhere in the network and attach to any traffic switch), but should be identified in order to analyze the path as required in the flow-grouping scheme.

E. Intrusion Detection

To inspect network traffic, there are many software-based IDSs, such as Snort and Suricata. In the testbed, Snort IDS is used for intrusion detection. Snort is an open-source IDS software that inspects network traffic by matching it with

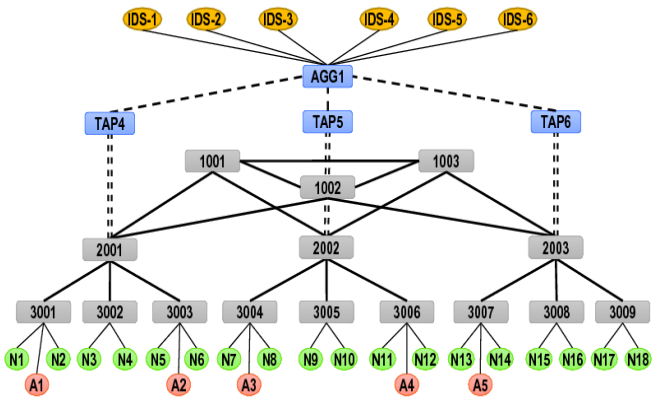


Fig. 3. Network configuration of the SDN-enabled testbed.

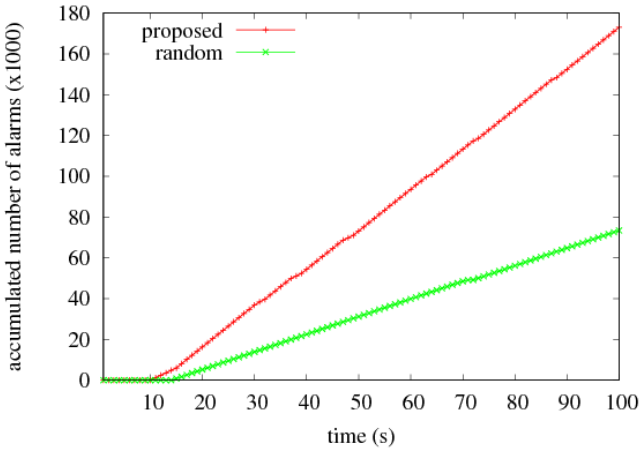
pre-defined rule sets. Snort can be easily installed and run in a low-cost virtualized environment (i.e., VMs and containers). In order to increase performance, based on the total user traffic, multiple IDSs can be deployed and used to handle some parts of the traffic analysis. The flow tapping mechanism helps traffic distribution between IDSs by connecting all IDSs with the same tapping aggregation switch that filters the traffic packets either randomly or with the flow-grouping scheme, and the filtered packets are sent into specific ports connected to specific IDS numbers. Snort IDS performs intrusion detection by counting the number of attack packets in a specific time duration before generating alarms or alerts. The more attack packets detected by the same IDS, the more likely that alarms or alerts will be generated.

Figure 3 shows the network topology of the SDN testbed with 19 SDN-enabled switches (15 switches for network topology and 4 switches for tapping and mirroring), and 29 containers (23 containers for traffic generation and 6 containers for IDSs). The number of flows generated is 30, and their flow rates vary from 1 Mb/s to 10 Mb/s. The number of attacking flows is 5 with a rate of 0.5 Mb/s. Note that DDoS attack uses a low rate with a number of small-sized packets. The IDS detects this attack when more than 800 packets are analyzed during 1 second.

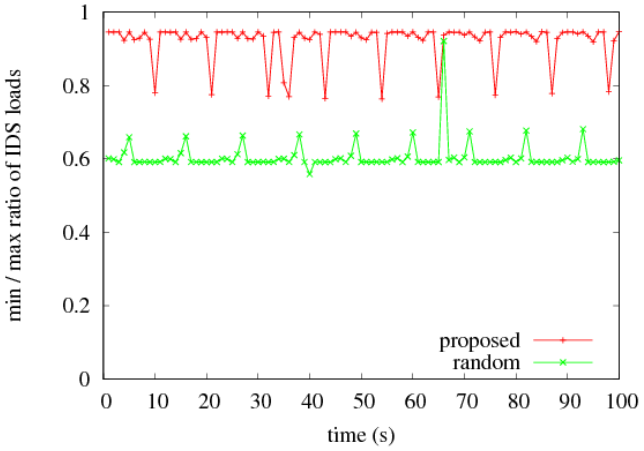
Figure 4(a) shows the accumulated number of alarms reported by the Snort IDS for the proposed algorithm and a random IDS selection algorithm. The first alarm of the proposed algorithm is earlier than that of the random method, and the number of alarms for the proposed algorithm is greater than that for the random algorithm. This result indicates that the proposed algorithm can achieve better intrusion detection performance relative to the random method. Figure 4(b) shows the min/max ratio of IDS loads. The ratio of the proposed algorithm is closer to 1, which implies that the loads are balanced well among the IDSs.

V. Conclusion

In this article, we considered the detection of malicious attacks on SDNs with multiple IDSs. With multiple IDSs, the intrusion detection performance highly depends on how the suspicious traffic flows are distributed among the multiple



(a) Attack detection performance



(b) Load-balancing performance

Fig. 4. Experiment results on SDN-enabled testbed.

IDSs. The proposed algorithm distributes the flows to multiple IDSs according their routing paths. If two flows are close to each other in terms of the routing path, they are forwarded to the same IDS. The proposed algorithm uses a gravity clustering algorithm to group the flows, and the cluster size is inversely proportional to the sum of the data rates in each group for load-balancing purposes. The experimental results indicated that the proposed algorithm outperforms a strategy that randomly distributes flows to the IDSs.

VI. References

[1] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 2046–2069, March 2013.

[2] 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.

[3] 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.

[4] 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, April 2014.

[5] Y. Wang, Y. Zhang, V. Singh, C. Lumezanu, and G. Jiang, "NetFuse: Short-circuiting traffic surges in the cloud," in *IEEE International Conference on Communications (ICC)*, June 2013, pp. 3514–3518.

[6] 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.

[7] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey, "Bohatei: Flexible and elastic DDoS defense," in *24th USENIX Security Symposium (USENIX Security 15)*, August 2015, pp. 817-832.

[8] S. A. Mehdi, J. Khalid, and S. A. Khayam, "Revisiting traffic anomaly detection using software defined networking," in *International Workshop on Recent Advances in Intrusion Detection, Springer Berlin Heidelberg*, September 2011, pp. 161-180.

[9] C. Yoon, T. Park, S. Lee, H. Kang, S. Shin, and Z. Zhang, "Enabling security functions with SDN: A feasibility study," *Computer Networks*, July 2015, pp. 19-35.

[10] Y.H. Chu, M.C. Tseng, Y.T. Chen, Y.C. Chou, and Y.R. Chen, "A novel design for future on-demand service and security," in *IEEE International Conference on Communication Technology (ICCT)*, November 2010, pp. 385–388.

[11] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *IEEE Conference on Local Computer Networks (LCN)*, October 2010, pp. 408–415.

[12] L. Wei and C. Fung, "FlowRanger: A request prioritizing algorithm for controller DoS attacks in software defined networks," *in IEEE International Conference on Communications (ICC)*, June 2015, pp. 6876–6881.

[13] B. Wang, Y. Zheng, W. Lou, and Y. T. Hou, "DDoS attack protection in the era of cloud computing and software-defined networking," *Computer Networks*, vol. 81, pp. 308–319, 2015.

[14] M. Indulska and M. E. Orlowska, "Gravity based spatial clustering," in *ACM international symposium on advances in Geographic Information Systems (GIS)*, November 2002, pp. 125–130.

[15] "Scapy," http://www.secdev.org/projects/scapy/.