Detection of Control Layer DDoS Attack using Entropy metrics in SDN: An Empirical Investigation

Kshira Sagar Sahoo
Department of Computer Science
National Institute of Technology, Rourkela, India
Email: kshirasagar12@gmail.com,

Manikanta Vankayala
Department of Computer Science
National Institute of Technology, Rourkela, India
manikanta.v403@gmail.com

Bibhudatta Sahoo
Department of Computer Science
National Institute of Technology, Rourkela, India
bibhudatta.sahoo@gmail.com

Ratnakar Dash
Department of Computer Science
National Institute of Technology, Rourkela, India
ratnakar@nitrkl.ac.in

***Abstract*— The Software Defined Networks (SDN) and OpenFlow technologies become the emerging networking technology that supports the dynamic nature of the network functions through simplified network management. The main innovation behind SDN is the decoupling of forwarding plane and control plane. In control plane, the controller provides a pivotal point of control to distribute the policy information throughout the network through a standard protocol like OpenFlow. Despite numerous benefits, SDN security is still a matter of concern among the research communities. The Distributed Denial-of-Service (DDoS) attack have been posing a tremendous threat to the Internet since a long back. The variant of this attack is quickly becoming more and more complex. With the advancement in network technologies, on the one hand SDN become an important tool to defeat DDoS attacks, but on another hand, it becomes a victim of DDoS attacks due to the potential vulnerabilities exist across various SDN layer. Moreover, this article focuses on the DDoS threat to control plane which is the central point of SDN. The entropy-based DDoS detection method is a wildly used technique in the traditional network. For detection of DDoS attack in control layer of SDN, few works have employed entropy method. In this paper, taking the advantages of flow based nature of SDN, we proposed General Entropy (GE) based DDoS attack detection mechanism. The experimental results show that our detection mechanism can detect the attack quickly and achieve a high detection accuracy with a low false positive rate.**

Keywords—SDN; Controller; DDoS; General Entropy

I. Introduction

Safeguarding the security of the network is a rat race process between attackers and victims for many years. Both academic and industry experts have been working in this area since a decade ago. Advancement of the technology, open up new attack tools to launch various attacks, consequently, the defenders require sophisticated and up-to-date defense mechanism to countermeasure the attack. As contrasting to other attacks, Distributed Denial of Service (DDoS) attack, can cause a massive interruption in any kind of network infrastructure. The intention behind the DDoS attacks is many, including political advantage, financial advantage, criminal extortion, and personal grudge etc. On the top list e-commerce, blogging sites, finance sectors are the target of the DDoS attack. With the recent advancement of virtualization-based cloud computing, Software Defined Network (SDN) paradigm, OpenFlow protocol many organization and researchers are adopting the security solutions using these technologies [1], [16].

In SDN the entire control decisions are made by a separate entity called controller. This decoupling framework brings many benefits to the network management and provides an easy solution to improve the overall network efficiency [12], [13]. As the control plane separate from the data plane, the OpenFlow protocol designs a secure channel for the communication. Hence, it is believed that a flexible and scalable network can be designed to the ever-changing business requirements through SDN. On one hand, the programmability and centralized view of the entire network help SDN controller to easily detect the attack, whereas on the other hand the centralized control architecture is considered as more vulnerable. Thus, for instance, SDN paradigm itself is likely to target by DDoS threat [17], [19].

Basically, there are two ways to detect this attack. One is signature based and another is anomaly based DDoS detection. In the signature based system, for an efficient detection, the signature needs to update continuously [18]. The DDoS attack can be categorized into the low rate or high rate attack depending on the speed of the malicious traffic. In anomaly-based detection systems, attackers train the detection systems to detect the attack traffic [2]. Usually, the false positive rate using the anomaly-based detection method is usually higher than the signature-based detection technique. It is difficult to set the actual thresholds which help to balance the false positive rate as well as the false negative rate.

In anomaly detection metric the threshold is fixed, hence an abnormal deviation of some statistical features from benign traffic, can help to identify abnormal traffic. Therefore, the choice of statistical techniques is so vital in case of DDoS detection. The information theory based metrics such as entropy can identify the variations of the traffic behavior of such events [11]. The main contribution of the paper are:

- Investigate various security issues of SDN and then focus on DDoS threat to the control layer.
- Implemented entropy metric based on incoming packets coming to the controller for identifying the attack traffic.
- Analyse the GE metric and compare the result with Shannon metric. We utilize these information to identify low rate DDoS attack.
- We simulate the above scenario on Mininet emulator along with POX controller.

The rest of the paper is organized as follows. Section II describes the motivation behind this work and related work. Section III discussed the information metrics used in the work. The detection procedure explains in section IV. The experimental setup is clearly mentioned in Section V. The performance of the algorithms and results are well discussed in Section VI. Finally, Section VII ends with concluding remarks.

II. Motivation and Related Work

The OpenFlow protocol provides a secure communication channel between the controller and the underlying switches. The controller is the central element of SDN that takes all the routing decision for the incoming flows to the network. All incoming flows are managed by flow tables of switches. There is a search in the flow table for every new packet to the switch. For a successful match, the flow action will carry out. Otherwise, the packet will be sent to the controller for further instructions. In turn, the controller will either add a flow rule or drop the flows from the flow table. Spoofing IP address is a common practice in the DDoS attack. In SDN scenario, for spoofed address, there is a mismatch each time on the table. Therefore, for each unmatched flow, a *packet_in* will send to the controller [14]. If the arrival rate of *packet_in* is very high in case of DDoS attack, the controller resources will start to deplete soon. A high rate IP spoofed may overwhelm the controller and as a result, it disconnects from the data plane. In a centralized SDN controller architecture single point of failure will defunct the entire network [15]. Hence, it is necessary to identify the DDoS traffic from the benign traffic.

In the last few years, there is significant research carried out on SDN security. Most of the authors have used the traditional solutions to the SDN. A Self-Organizing Maps (SOM) based machine learning model was used for detecting DDoS attack traffic by Braga et al. [3], but ignores the control plane attacks. Shin et al. proposed AVANT-GUARD, for detection and prevention of SYN flooding attack [4]. Anomaly detection mechanism proposed by Giotis et al., leverage the properties of both OpenFLow and sFlow [5]. The sFlow is used for traffic sampling and OpenFlow protocol is used for mitigating the attack by modifying the priority values of the existing flows. The COFFEE framework utilizes the OpenFlow protocol to identify the zombie's activities and delete the flow entries [6]. An adaptive flow collection method has proposed in [7] and [8] for DDoS detection in SDN. A traffic measurement tool called OpenSketch, uses a hash table for measuring the traffic. Instead of flow sampling, this tool uses a three stage pipeline process to collect traffic and identify the malicious traffic from them. Most of the DDoS detection solutions for SDN, have used machine learning and knowledge-based techniques to identify the attack traffic. Very few authors utilize the statistical method for attack detection in SDN [9]. An entropy based solution was proposed by Mousavi *et. al.* [10], for early detection of DDoS attack in SDN. In their experiment, after an extensive experiment, the threshold value of entropy has been chosen. The threshold value can be adjusted with the dynamic nature of the incoming traffic. Although the true positive rate is much higher, but the false positive still exist in their work. Motivated by this, we have evaluated the above work and in addition to this we have used general entropy (GE) metric to lower the false positive rate.

III. Background of General EntropY

Entropy was introduced to measure the uncertainty of an event associated with a given probability distribution *X*. The formal definition of entropy in terms of a discrete variable *X*, with possible outcomes $x_1, x_2, \ldots, x_n$ can be defined as:

$$\begin{aligned} H(X) &= \sum_{i=1}^{n} p(x_i)\log_2 \frac{1}{p(x_i)} \\ &= -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i) \end{aligned} \tag{1}$$

In Equation 1, $p(x_i)$ is the probability of the i^{th} outcome of *X*. A generalized entropy (GE) can be defined as:

$$H_\alpha(X) = \frac{1}{1-\alpha}\log_2\left(\sum_{i=n}^{N} p_i(\alpha)\right) \tag{2}$$

By varying the α order of Equation 2, different types of entropy values can be obtained. When $\alpha = 0$, it indicates the maximum value of the generated information. It can be derived as: $H_0(X) = \log_2 n$. But, when $\alpha=1$, the GE can be expressed as: $H_1(X) = -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i)$, which is termed as Shannon Entropy *(E_{sh})*.

Lemma 1. $H(X) \geq 0$Proof: It's already known that: $p(x_i) = prob(X = x_i)$ is the probability of i^{th} outcome of *X.* $0 \leq p(x_i) \leq 1$, implies that $\log \frac{1}{p(x_i)} \geq 0$.

Lemma 2. $H_b(X) = \log_a H_a(X)$

Proof: $\log_b M = \log_b a \log aM$. This property of entropy allow us to change one base to another of the logarithm.

Minimum entropy information can be obtained when α value is ∞ i.e. H_∞. The probability density is high when *α* value is When $\alpha \geq 0$, the derivate of $H_\alpha \leq 0$. It implies that the GE value is a non-increasing function of α. So it can be written as:

$$H_{\alpha 1}(x) \geq H_{\alpha 2}(x) \tag{3}$$

The GE value depends on _ which increase the differences between two probability distributions while comparing with the Shannon entropy (E_{sh}). When we analyse the formulas of GE and (*Esh*), it can be concluded that the high probability event can provide more information to the GE than (E_{sh}) at $\alpha > 1$. Hence, according to the requirement, we can get different and better detection result by adjusting the value of α. The Equation 4 shows when $\alpha =\ \ 2$ the GE value can be:

$$H_2(x) = -\log_2 \sum_{i=1}^{n} p_i^{\ 2} \tag{4}$$

We formulate our SDN based DDoS attack detection on the above analysis.

IV. DDoS Detection Algorithm

Before discussing the detection algorithm, we formalize certain terms for further explanation. For a SDN flow, there are certain header fields exist. We define the input flow by 6 tuple such that:

$IFi = \{src_{ip}; dest_{ip}; src_{MAC}, dest_{MAC}, src_{MAC}, dest_{port}\}$.

It is a set of packets containing the similar properties moving through a same network channel in a given time period. The Algorithm 1 is meant for statistics collection for the unmatched incoming flows. When the number of *packet_in*s reached to 100, it creates a hash table for the incoming flows. For every incoming *packet_in* coming to the controller, the module (Algorithm 2) first unwraps it and get the flow information from the flow table. Then extract the $dest_{ip}$ and stored into a hash table. The hash table *H*, contains the $dest_{ip}$ and occurrence of it. This information helps for computing the entropy and GE. The order of *α* value helps to improve the detection rate. In the first experiment, we have collected the entropy values of attack traffic and normal traffic on the different value of α for 80 windows.

Algorithm 1 Flow_Statistics_Collections

```
Input: sampling interval (Δ)T,r=0
Output: Hash table H(dest_ip, occurrence)

    1.  For all packet_ins received by the controller
        C do
    2.  Create a hash table
    3.  End For
    4.  When (Δ)T over,
    5.  For all dest_ip
    6.      If dest_ip_i ∉ H(dest_ip_i,r) do
    7.          H(dest_ip_i, r) ←1
    8.      Else
    9.          H(dest_ip_i, r) ←r+1
    10.     End If
    11. End For
Output: Return H(dest_ip, occurrence)
```

Algorithm 2 HR_DDoS Attack_Detection

```
Input: Hash table H(dest_ip, occurrence)
Output: Detection of DDoS attack
    1.  Set the threshold (_) using Equation 7
    2.  For all IP ∈ Hashtable do
    3.    Find the probability distribution
    4.    Calculate entropy($E_{sh}$)
    5.       If $E_{sh} < \delta$ then
    6.          DDoS attack detected
    7.       Else
    8.          Normal Traffic
    9.      End If
    10. End For
Output: Return DDoS attack Alert
```

We assume that if the calculated entropy value is less than the preset threshold value for a 10 consecutive windows, then it has considered that there will be a possible of attack. Detection of 1000 packets within 10 entropy periods gives an early alert of attack to the SDN controller.

V. Experimental Setup

In this work, an OpenFlow (OF) controller has connected to the network consisting of OF enabled switches to create an SDN network. During a DDoS attack scenario, some important issues need to be considered. The GE of the controller traffic is evaluated under attack and non-attack scenario.

A. Controller

Before experiment choosing a controller is an important task. Among few available controllers, the POX controller is used in our experiment. As it is a fast and lightweight controller, most of the authors have used this in their experiment. This controller can work on all most all available platforms.

B. *Network Emulator:*

To emulate the data plane of testbed networks, we use Mininet emulator tool. The kernel namespace feature of Mininet helps to prototype a network scenario in a single PC. The individual process will have their own routing table, network interface etc. Mininet takes the help of this features and utilizes the process based virtualization concept to run network elements in the kernel. In addition to Mininet we have used Scapy tool for network traffic generation.

C. *Packet Generation Tool:*

In order to create DDoS attack scenario on Mininet we need to generate custom attack packets and send them to victims. To accomplish this task the packet generation tool Scapy has been used. With regard to the proposed method, the Scapy tool is considered as a powerful tool to generate real flooding attacks [20]. In our experiments this tool has been used to generate both TCP and UDP packets and spoof the source IP address of the packet. We have used the "randrange" function of Python to generate random IP addresses.

D. *Network Setup:*

We run our experiment on a PC with Intel Core i7-4770 processor, 3.4GHz clock speed with 4GB RAM. The operating system is Linux Ubuntu 14.04 LTS and Mininet V 2.2.26 which supports the OF version 1.3. Using MiniEdit, a simple user interface located in Mininet, a tree type network has been created. The network consists of 10 switches and 64 hosts. The Open Virtual Switches are acted like OF switches. The OVS is referred to as the OpenFlow enabled switch. The L3 learning module is used for SDN controller. In reactive mode, this module serves several things such as: learn the association between IP and MAC addresses, generating ARP requests in case of a destination IP is not known, install the flow rule in the flow table etc. The bandwidth of each link between OF switches is 800 Mbps and the link between hosts to switch is 100 Mbps.

VI. Performance Evaluation

In order to evaluate the proposed method, we have conducted various experiments on the above-stated SDN test bed. In case of DDoS attack scenario, all attack nodes attack to the victims in a distributed coordinated manner using a shared logic program. For threshold value we have used the method used by Mousavi *et al*. If the calculated entropy value persists for ten consecutive windows and less than the threshold then we conclude that an attack is in progress. To find the optimal threshold value, we conduct a set of experiments. For comparison purpose we generate different rates of incoming *packet_in*s to the controller. The packet rate R can be decided by the Equation 5.

$$R = \frac{P_{attack}}{P_{norm}} * 100 \tag{5}$$

Here P_{attack} and P_{norm} denotes the number of attack and normal traffic respectively. The threshold value can be calculated as follows. At first we calculate the possible minimum value of a normal traffic entropy. This can be achieved by difference between normal traffic mean entropy and confidence interval. Then, we calculate the possible maximum value of the attack traffic. It can be achieved by adding the attack traffic mean entropy and confidence interval.

VII. Simulation Result

Keeping the window size 80, we test the experiment several times. We conduct the experiment on four different attack rate. In the 50% and 80% cases, since the attack packets increases within the window, the drop in the value of the entropy will be a minimal gap. All the scenarios are given below.

Table I represent the threshold value selection in different attack scenario. The Fig. 1, shows the drop for 10% attack rate is very small compared to the other attack rate. The entropy value not clearly discriminate the attack traffic from the normal traffic in a lower rate attack scenario. In case of 50% and 80% attack rate, the entropy value drops substantially as depicted in Fig. 4 and Fig. 5. The difference is insignificant in case 10% attack rate. Although the true positive rate is almost 100%, the false positive rate still high. At the higher rate of DDoS attack, the false positive is near about 2%. Table II represents the obtained error rate during the simulation. From this, we conclude that it is not only to identify the attack traffic correctly, at the same time it should identify the benign traffic as well. Hence, the entropy value is not sufficient to discriminate the benign traffic correctly. To solve this we have employed GE.

In the beginning, we have used the destination IP address to calculate the entropy value within a time window. The individual probability value of the IP addresses is within the value of 0 and 1. For total entropy, we sum all the individual probability within the window. The generalized entropy value of different α order and the spacing between the attack and benign traffic has shown in Fig. 6.

.

TABLE I. Threshold value comparison w.r.t. different attack rate

Parameters	Normal Traffic	10% attack traffic	20% attack traffic	30% attack traffic	50% attack traffic	80% attack traffic
Mean	**0.8189**	**0.80705**	**0.7751**	**0.72994**	**0.6274**	**0.3327**
Standard Deviation	**0.0152**	**0.0193**	**0.0256**	**0.03007**	**0.0374**	**0.03711**
Confidence-Max	**0.8204**	**0.80988**	**0.77760348**	**0.733955**	**0.63103**	**0.33627**
Confidence-Min	**0.8174**	**0.80421**	**0.77260259**	**0.723955**	**0.623748**	**0.32908**
Confidence-Interval	**0.00301**	**0.00566**	**0.004957**	**0.0080157**	**0.0072816**	**0.007219**
Threshold		**0.8058**	**0.7894**	**0.75174762**	**0.6346**	**0.3439**

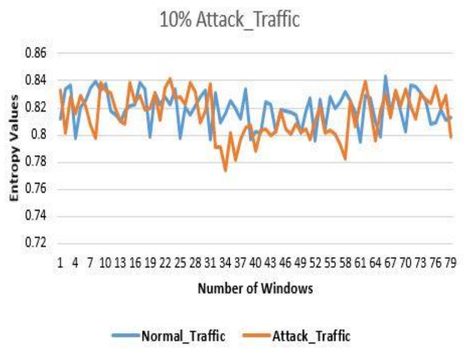

Fig. 1: 10% attack rate

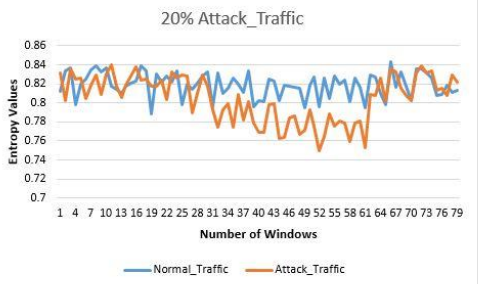

Fig 2: 20% Attack rate

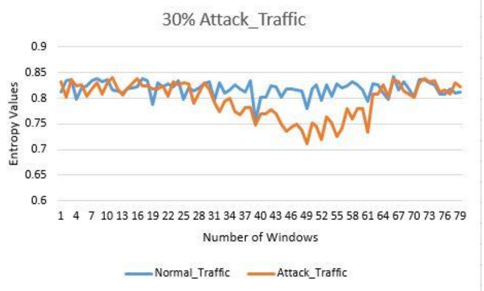

Fig 3: 30 % attack rate

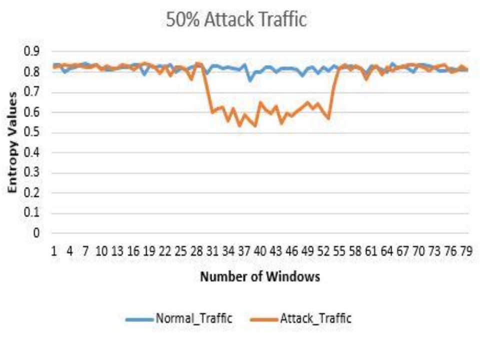

Fig 4: 50 % attack rate

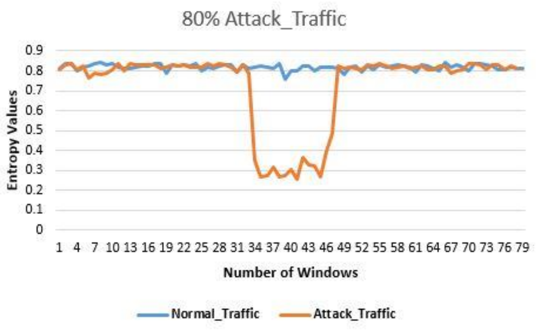

Fig 5: 80% attack rate

TABLE II. Different attack rate with error rate

Value	10% attack rate	20%	30%	50%	80%
True Positive Rate	93	94	96	96	98
False Positive Rate	7	5	3	3	2

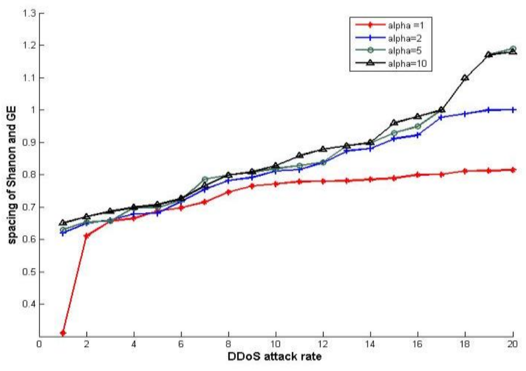

Fig 6: variation of spacing between GE and Shanon

Table III. Attack rate with error rate (α = 10)

Value	10% attack rate	20% attack rate
True Positive Rate	98	97
False Positive Rate	1	2

To evaluate the performance of GE metric, we test it in the following scenario. We observe the variation of the spacing by changing the attack rate. The goal of our work is to reduce the false positive in a low rate attack rate. Fig. 6 shows that the spacing of Shannon (E_{sh}) and GE are increasing in nature with respect to the increasing number of attack traffic. During the initial stage of the spacing between E_{sh} and GE are not so discriminative, because of the low rate attack. However, the spacing of the GE has stable value when α = 10. It can be observed that the spacing of E_{sh} cannot produce a stable value with the increasing number of attack traffic.

VIII. CONCLUSION

Low rate DDoS attack is a serious threat to the SDN control layer. It is very much important to identify the attack much before it happens. One of the way the controller layer can be attacked by increasing the number of *packet_in* control packets. When *packet_in* events increases, it becomes a bottleneck for the controller. In such situation, normal Shanon entropy is a less efficient method to detect the false alarm. Hence, we have employed general entropy (GE) metric to discriminate low rate DDoS attack and normal traffic. We have observed that this metric can able to identify attack traffic from legitimate traffic to a greater extent with a better false positive rate. In the future work, we can employ this technique for high rate DDoS attack and try to set the threshold more dynamic way in a real traffic scenario such that the detection can be done as early as possible.

References

[1] Hu, F., Hao, Q. and Bao, K. (2014). A Survey on Software-Defined Network and OpenFlow: From Concept to Implementation. *IEEE Communications Surveys & Tutorials*, 16(4), pp.2181-2206.

[2] Swain, Biswa Ranjan, and Bibhudatta Sahoo.(2009) "Mitigating DDoS attack and Saving Computational Time using a Probabilistic approach and HCF method." In *IEEE International Advance Computing Conference (IACC)*,pp. 1170-1172.

[3] Rodrigo Braga, Edjard Mota, and Alexandre Passito(2010). "Lightweight DDoS flooding attack detection using NOX/OpenFlow." *35th Conference on*. *Local Computer Networks (LCN)*, pp.408-415 ,*IEEE.*

[4] Shin, Seungwon, Vinod Yegneswaran, Phillip Porras, and Guofei Gu. "Avant-guard: Scalable and vigilant switch flow management in software-defined networks.(2013)" In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 413-424.

[5] Giotis, K., Argyropoulos, C., Androulidakis, G., Kalogeras, D. and Maglaris, V. (2014). Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments. *Computer Networks*, 62, pp.122-136.

[6] Schehlmann, Lisa, and Harald Baier.(2013) "COFFEE: a Concept based on OpenFlow to Filter and Erase Events of botnet activity at high-speed nodes." In *GI-Jahrestagung*, pp. 2225-2239.

[7] Zhang, Ying. (2013)"An adaptive flow counting method for anomaly detection in SDN." In Proceedings of the ninth ACM conference on Emerging networking experiments and technologies, pp. 25-30. ACM.

[8] Yu, Minlan, Lavanya Jose, and Rui Miao.(2013) "Software Defined Traffic Measurement with OpenSketch." In *NSDI*, vol. 13, pp. 29-42.

[9] Wang, Rui, Zhiping Jia, and Lei Ju. (2015)"An entropy-based distributed DDoS detection mechanism in software-defined networking." In *Trustcom/BigDataSE/ISPA*, vol. 1, pp. 310-317. *IEEE.*

[10] Mousavi, Seyed Mohammad, and Marc St-Hilaire. (2015) "Early detection of DDoS attacks against SDN controllers." In *International Conference on, Computing, Networking and Communications (ICNC),* pp. 77-81. IEEE.

[11] Xiang, Yang, Ke Li, and Wanlei Zhou. (2011) "Low-rate DDoS attacks detection and traceback by using new information metrics." *IEEE Transactions on Information Forensics and Security* 6,(2),pp. 426-437.

[12] Jarraya, Yosr, Taous Madi, and Mourad Debbabi.(2014) "A survey and a layered taxonomy of software-defined networking." *IEEE communications surveys & tutorials* 16(4), pp. 1955-1980.

[13] Farhady, Hamid, HyunYong Lee, and Akihiro Nakao.(2015)"Software-defined networking: A survey." *Computer Networks* 81, pp.79-95.

[14] Haleplidis, Evangelos, Kostas Pentikousis, Spyros Denazis, J. Hadi Salim, David Meyer, and Odysseas Koufopavlou (2015). *Software-defined networking (SDN): Layers and architecture terminology*. No. RFC 7426.

[15] Sahoo, Kshira Sagar, Sagarika Mohanty, Mayank Tiwary, Brojo Kishore Mishra, and Bibhudatta Sahoo.(2016)"A Comprehensive Tutorial on Software Defined Network: The Driving Force for the Future Internet Technology." In *Proceedings of the International Conference on Advances in Information Communication Technology & Computing*, pp. 114,ACM.

[16] Gyires, Tibor. "Software Defined Networking; OpenFlow."

[17] Akhunzada, Adnan, Ejaz Ahmed, Abdullah Gani, Muhammad Khurram Khan, Muhammad Imran, and Sghaier Guizani.(2015) "Securing software defined networks: taxonomy, requirements, and open issues." *IEEE Communications Magazine* 53(4), pp. 36-44.

[18] Sahoo, Abhaya Kumar, Kshira Sagar Sahoo, and Mayank Tiwary. (2014)"Signature based malware detection for unstructured data in Hadoop." In *2014 International Conference on Advances in Electronics, Computers and Communications (ICAECC)*, pp. 1-6, IEEE.

[19] Sahoo, Kshira Sagar, Bibhudatta Sahoo, and Abinas Panda.(2015) "A secured SDN framework for IoT." In *International Conference on Man and Machine Interfacing (MAMI)*, pp. 1-4. IEEE.

[20] Secdev.org. (2017). *Scapy*. [online] Available at: http://www.secdev.org/projects/scapy/ [Accessed 9 Dec. 2017].