A Novel DDoS Attacks Detection Scheme for SDN Environments

Di Wu*, Jie Li*, Sajal K. Das†, Jinsong Wu‡, and Yusheng Ji§

*Department of Computer Science
University of Tsukuba, Japan
Email:wudi@osdp.cs.tsukuba.ac.jp and lijie@cs.tsukuba.ac.jp

†Department of Computer Science
Missouri University of Science and Technology, USA
Email: sdas@mst.edu

‡Department of Electrical Engineering
University of Chile, Santiago, Chile
Email: wujs@ieee.org

§Information Systems Architecture Research Division
National Institute of Informatics, Japan
Email: kei@nii.ac.jp

***Abstract*—Software-Defined networking (SDN), as a new paradigm, fixes the shortage that traditional network does not support the dynamic, scalable computing and storage needs of more computing environments. SDN, however, also faces security problems such as vulnerable to DDoS attacks. DDoS attacks are well-known and powerful attacks. DDoS detection and DDoS traffic separation for SDN environments are still an open research issue. DDoS attacks in SDN environments will not only bring damage to target server, but also takes exact impact on SDN system. In this paper, we identify a new type DDoS attack, specifically aiming SDN environment, which is harder to be detected. We propose a novel real-time DDoS detection scheme for SDN environment, by using Principal Component Analysis (PCA) scheme to analyze the network status on traffic packets data. We separate the network into different parts, to reduce the total calculation burden. We compare our scheme with sample entropy, showed our scheme achieves better detecting ability for DDoS attacks.**

I. Introduction

Software-Defined networking (SDN) separates a network's control logic from switches, simplifies network management. It allows network administrators to manage network services through abstraction of lower-level functionality. SDN is meant to address the fact that the static architecture of traditional network doesn't support dynamic, scalable computing and storage needs of more computing environments such as data centers. This is done by decoupling or disassociating the system that makes decisions about where traffic is sent (The Control Plane) from the underly systems that forward traffic to the selected destination (The Data Plane). SDN [1] is an emerging architecture that is dynamic, manageable, cost-effective, and adaptable, which it ideal for the high-bandwidth, dynamic nature of today's applications. These specific capabilities make SDN deployable in many network environments, from home and enterprise network to data centers in cloud networks.

OpenFlow [2] is a protocol that standardizes how SDN switches communicate with an SDN controller. In OpenFlow white paper [3], it introduced a way for researchers to run experimental protocols in the networks. DDoS attacks are widely used to run out the target's network bandwidth or process resources. DDoS attacks are not only effective in traditional networks, but also available in SDN environments. Many related works have been conducted to solve DDoS attacks on SDN environments. Kazemian, et al. [4] and Khurshid, et al. [5] check the SDN environment in real time, Shin and Porras, et al. [6] [7] focused on the flow table problem, Wang, et al. [8], Garg, et al. [9] studied the payload of SDN, Hong, et al. [10] solved the topology poisoned problem. and Dong, et al. [11] focusing on low-traffic flows DDoS on SDN.

Previous studies of quick DDoS detection [12], [13] and [11], show some treatment against DDoS attacks on SDN, but there are shortages in these work. For example, Mousavi, et al. [13] used the entropy to describe the traffics, which may cause a false alarm when traffic feature getting larger. Dong's method [11] only studies the flows with low traffic, which is powerful to solve the traffic pattern, which cannot handle other kinds of DDoS attacks.

Principal Component Analysis (PCA) is a traditional scheme which reduce data size, usually been used for picture analysis. We use it on packet traffic data collected in SDN environments, to analysis whether a DDoS attack is inside this network. Along with dealing DDoS attack, we also consider a new type DDoS attack which specifically points at SDN environments. This new DDoS attack brings more impact on switches and controllers and is difficult to be detected by traditional DDoS detecting schemes.

Our detecting scheme is based on separation of the space of traffic measurements into normal and anomalous subspaces, by means of Principal Component Analysis. We collect simulation data of the whole network traffic packets in our environment, and test the whole network scheme, partition scheme, and sample entropy in different situations. Our results show that all three schemes could handle traditional DDoS attack, but the new type DDoS attack aiming weak points of SDN, limited storage in switches and limited calculating ability in controller, can not be detected by sample entropy.

In this paper, we identify a new type DDoS attack targeting the SDN environment specifically. This DDoS attack is basically the same as conventional DDoS, using zombie computers to send packets. However, instead of sending to a common target server, the DDoS attack send packets to random targets. This behavior change makes this new type DDoS attack harder to be detected, and bring more impact on SDN environments.

The contributions of this paper are as follows.

- We adapt a scheme using principal component analysis to diagnose anomalies in SDN network.
- We partition the network to get a lighter burden.
- We identify a new type DDoS attack specifically aiming SDN environment.
- We compare the result between using PCA and sample entropy under different settings including the new type DDoS attacks.

The organization of the rest of this paper is as follows. Section 2 describes the problem that DDoS attack occurs on SDN environment will make extra damages on SDN environment, and a novel DDoS attack aiming SDN only, who amplify those damages on SDN environment. Section 3 describes how to use Principal Component Analysis to analysis network. Section 4 describes the experiment and result. Section 5 concludes the paper.

II. PRELIMINARIES

A. *PCA on traditional networks*

We followed Lakhina's research [14]–[16], which introduced a method detecting anomalies for traditional networks, using principal component analysis to analysis data traffic of the network.

Principal Component Analysis (PCA) is a coordinate transformation method that maps the measured data onto a new set of axes. These axes are called the principal axes or components, each principal component has the property that it points in the direction of maximum variation or energy remaining in the data, given the energy already accounted for in the preceding components. So the i-th principal component captures the total energy of the original data to the maximal residual energy beside former $i-1$.

First of all, we assume the network administrator are able to collect traffic data through the network. And for SDN environment, it could be easily done by using flow table to collect packet data.

OD Pair. OD-pair presents a pair of node describe the origin node and the destination node of one packet.

OD Flow p. An OD flow consists of all traffic for this OD-pair. If the network has k entrance, there will be k^2 PoP-pairs maximum, and hence k^2 OD pairs. For short, we set the number of OD flows as p.

Number of successive time intervals of interest t. We collect successive network's traffic for total $w \times t$ seconds, and separate the time period into t pieces. Therefore each time period last for w seconds. And we could adjust the number of time period t to t_1, meanwhile adjust the length of time of each time period to w_1, so that $t \times w = t_1 \times w_1$. Therefore we could get a fit t, that $t > p$.

Matrix X. Let X be the $t \times p$ measurement matrix, which denotes the time series of all the OD flows. Each column i denotes the time-series of $i-th$ OD flow and each row j represents an instance of all the OD flows at time period j.

For matrix X^TX, it could calculates

$$X^TXv_i = \lambda_i v_i, \tag{1}$$

where $\{v_i, i = 1, .., p\}$ are the eigenvector, $\{\lambda_i, i = 1, .., p\}$ are the eigenvalues corresponding to each v_i.

Finding the first r non-negligible principal component, could approximate the original matrix.

Detecting anomalies relies on the separation of x (Matrix X's i-th row, a vector of all flows at i-th interval) into normal and anomalous components, refer as the modeled and residual parts of x. And decompose the set of measurements at a given point in time x:

$$x = \hat{x} + \tilde{x}, \tag{2}$$

that $\hat{x}$ corresponds to modeled and $\tilde{x}$ to residual traffic. To accomplish this, need to arrange the set of principal components corresponding to the normal subspace $(v_1, v_2, ..., v_r)$ as columns of a matrix P with size $m \times r$ where r denotes the number of normal axes (chosen in II-B). And write $\hat{x}$ and $\tilde{x}$ as:

$$\hat{x} = PP^Tx = Cx, \text{ and } \tilde{x} = (I - PP^T)x = \tilde{C}x, \tag{3}$$

where the matrix C represents the linear operator that performs projection onto normal subspace, and $\tilde{C}$ projects onto the anomaly subspace. The occurrence of a volume abnormal will tend to result in a large change to $\tilde{x}$. A useful statistic for detecting abnormal changes in $\tilde{x}$ is the squared prediction error (SPE):

$$SPE \equiv \|\tilde{x}\|^2 \equiv \|\tilde{C}x\|^2, \tag{4}$$

and consider network traffic to be normal if $SPE \leq \delta_\alpha^2$ where δ_α^2 denotes the threshold for the SPE at the $1-\alpha$ confidence level. A statistical test for the residual vector known as the *Q-statistic* was developed by Jackson and Mudholkar and is given in [17] as:

$$\delta_\alpha^2 = \phi_1[\frac{c_\alpha\sqrt{2\phi_2h_0^2}}{\phi_1} + 1 + \frac{\phi_2h_0(h_0-1)}{\phi_1^2}]^{\frac{1}{h_0}}, \tag{5}$$

where

$$h_0 = 1 - \frac{2\phi_1\phi_3}{3\phi_2^2} \text{ , and } \phi_i = \sum_{j=r+1}^{m} \lambda_j^i \text{ ,} i = 1, 2, 3, \quad (6)$$

and where λ_j is the variance captured by projecting the data on the j-th principal component ($\|Xv_j\|^2$), and c_α is the $1-\alpha$ percentile in a standard normal distribution.

For matrix X of size $t \times p$, calculating the principal components is equivalent to solving the symmetric eigenvalue problem for the matrix X^TX, which is a measure of the covariance between flows.

Take the rows of X as points in Euclidean space, so that we have a dataset of t points in $\mathbb{R}^p$.

Each principal component v_i is the i-th eigenvector computed from the spectral decomposition of X^TX:

$$X^TXv_i = \lambda_i v_i, \quad (7)$$

where λ_i is the eigenvalue corresponding to v_i. And because X^TX is symmetric positive definite, its eigenvectors are orthogonal and the corresponding eigenvalues are nonnegative real. By convention, the eigenvectors have unit norm and the eigenvalues are arranged from large to small, so that $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_p$.

It can be shown that the eigenvector corresponding to the maximum energy of the residual by using the Rayleigh Quotient of X^TX. We can write the k-th principal component v_k as:

$$v_k = arg \max_{\|v\|=1} \|X - \sum_{i=1}^{k-1}(Xv_iv_i^T)v\|. \quad (8)$$

Thus, computing the set of all principal components, $\{v_i\}_{i=1}^p$ is equivalent to computing the eigenvectors of X^TX.

The principal component space can be used to examine the transformed data. The contribution of principal axis i as a function of time is given by Xv_i, and can be normalized to unit length by dividing by $\sigma_i = \sqrt{\lambda_i}$. Thus, we have each principal axis i,

$$u_i = \frac{Xv_i}{\sigma_i}, \quad i = 1, ..., p. \quad (9)$$

The u_i are vectors of size t and orthogonal by construction. The equation above shows that all the OD pairs, when weighed by v_i, produce one dimension of the transformed data. u_i captures the i-th strongest temporal trend common the all OD pairs, and the set of $\{u_i\}_{i=1}^p$ captures the time-varying trends common to the OD pairs, refer to them as the *eigenflow* of X.

The set of principal components $\{v_i\}_{i=1}^p$ can be arranged in order as columns of a principal matrix V, which has size $p \times p$. Likewise, we can form the $t \times p$ matrix U in which column i is u_i, that V, U and σ_i can be arranged to write each OD flow X_i as:

$$\frac{X_i}{\sigma_i} = U(V^T)_i \quad i = 1, ..., p. \quad (10)$$

The elements of $\{\sigma_i\}_{i=1}^p$ are called the singular values, and $\|Xv_i\| = v_i^TX^TXv_i = \lambda_i v_i^Tv_i = \lambda_i$.

B. Chosen Subspace

Specifically, finding that only r singular values are non-negligible, implies that X effectively resides on an r-dimensional subspace of $\mathbb{R}^p$. In this case, we approximate the original matrix as:

$$X' \approx \sum_{i=1}^{r} \sigma_i u_i v_i^T \quad (11)$$

III. System Statement and Problem Statement

A. SDN Matching Process

Packet matching process in SDN has limited storage spaces and process resources. These resources could be easily run out when DDoS attacks occur in SDN. Details of matching process in SDN is as following.

According to OpenFlow switch specification [2], through the pipeline processing is able to lookup through different flow table in a switch, we could simply consider there is one flow table in a switch.

SDN switch depends on flow tables to instruction traffic packets. The switch extracts the match fields from packets, depend on the packet type, and typically include various packet header fields, such as Ethernet source address or IPv4 destination address, to lookup a match in flow table. The packet will be matched against flow table. Only the flow entry with highest priority matched can be selected, and this matched table flow must be selected. The counters associated with the selected flow entry be updated and the instruction set included in the selected flow entry be applied.

When the controller receives the message, it searches its flow table for a match, if there get a match, the controller will instruct the switch to install a rule on the flow table, so that the switch knows how to handle the packet. And if there is no match, the controller sends PACKET-OUT to all connected switches, request for the target host, if one switch get a match, it will return a message to controller, and the controller will record the rule on its own flow table, and instruct the original switch. And if there is no match, it will need to wait till timeout.

When DDoS occurs, a packet burst will occurs in the network. But beside the amount of packets, there are two patterns that only DDoS contains, one is lot of packets from different source to one destination, and two is start in short time. Unlike a hot topic, which need time to separate, usually has exponential growth, DDoS attack usually grows like a spark, unless the attacker pretend this identification. And another different between hot topic with DDoS is that one people usually view this topic once, but DDoS need each bot continually access the target to increase its effectiveness and strength. According to the matching problem mentioned above, there will be two extra side effect on SDN environments.

1. Impact on switches. Suppose the DDoS attack occurs among all this network, and bots of the "botnet" controlled by the attacker separate in each switch. There is no matching flow table rule exists, the space of switch to store uninstructed packets will exhaust, and have to drop old or new packet

TABLE I: Main components of a flow entry in a flow table [2].

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie

(depending on the policy) when a new uninstructed packet comes in. Another problem is that if the controller manage the flow table poorly, that each separate packet with different source and destination, usually called OD (origin-destination) pair, need an individual flow table space, the flow table will run out of space quickly.
2. Impact on controller. When DDos attack happens, huge amount of uninstructed packets through different switches waiting for the controller's instruction, this will use up the controller's process ability quickly, and cause the latency of instruction and cause time out, leading to packet lost, or the controller down totally and the network is unable to work.
With higher specification device of switch and controller, that has larger space and faster process spend, allows SDN environment to handle more packets, but this can't fix the problem. And there are also other method to amplify these side effect.
1. Low-traffic flow. In [11] introduced that no matter how heavy the traffic of a new flow is, only the first few packets of the flow will be encapsulated in the packet-in messages and sent to the controller. Thus, the attackers will prefer to use low-traffic flows to gain more impact to trigger attack on controller.
2. Heavy-traffic flow. On the contrary, we could use heavy-traffic that each packet filled with meaningless data to achieve maximum size to consume the space of switches.

B. The New Type DDoS Attack in SDN

While we doing the research, we found that we could launch a new type DDoS to amplify effective on SDN environment. This new type DDoS attack different with traditional DDoS attack, that the destination of packet is randomly choose. This attack is not aiming on one fixed target server, but the SDN network system. So that there will be no server detecting been attacked, therefore no server will alarm been attack, therefore harder to be detected and reported.
(1). Amplify on Buffer. Buffer store all packets waiting for controller's instruction. But once this controller was instructed, until the controller give another instruction to remove this instruction, the switch has table flow recording how to deliver the packets. It means that only the first packet of a new flow, and if this flow occurs before, the first packet may even don't need to be buffered and wait.
With new type DDoS, because of the randomness of destination IP, there is only small chance this flow has been instructed and record in the switch. So that almost every packet at any time need to be buffered, and wait for controller's instruction, while the traditional DDoS attack usually has same target, that once instructed no packet would need to be buffered.
(2). Amplify on Flow Table. As we mentioned above, once the controller instructs using a table flow with fixed target IP, following packets will all match this table flow.
But with new type DDoS, the following packets have different origin and destination, so that need a new table flow for every packet. And because there are no pattern between the packets, it's very hard to classification table flows, which means it will cost lot of flow table space. Further more, when each packet consume a table flow, the table could be easily filled with attack flow.
With randomly OD pairs, it is difficult for controller to classify the flow table, cost more flow table space, running out flow table space. And since the destination is randomly, instead of the original attack only affect the switch in the beginning, there is no matching flow table all the time, there will be uninstructed packets all the time, and exhaust the space storing uninstructed packets. On controller side, with randomly destination, which could be high possibility not exist, with no reply from other switches, the controller has to wait time out, it will be much easier to exhaust the controller's resource.
Therefore, automatically real time DDoS attack detection is one urgent problem in SDN environment.

IV. Proposed Scheme

In a typical SDN, all the extra work (data collect, matrix calculate, result compare) need to be done at the controller side in each time interval. Consider of the bandwidth between controller and switches, and controller's computational capabilities, this could put the controller into a risky position. And inspired by [18], which pointed out that OpenFlow assumes a logically centralized controller, which ideally can be physically distributed. Also Onix [19] is network-wide control platform running on one or more servers in the network handling switches by partition and aggregate, we are interested in partition the network using several sub-controller, to get separated data, and report it to the controller.
Assume we separate one network into s subnets, we can get a set of OD flow $\{p_1, p_2, ..., p_s\}$. For each OD flow p_i of the i-th subnet, it contains the OD pairs that origin or destination is in this subnet (or both). Therefore we get a set of Matrix $\{X'_1, X'_2, ..., X'_s\}$. And each X' have same time interval amounts as X.

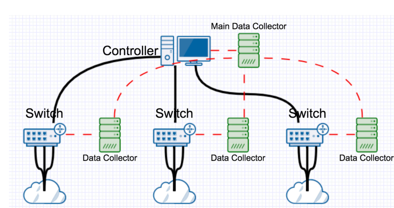


Fig. 1: System Model

As in figure 1, our system basically need extra computer(s) to collect packet flows through every switches. It would be easier if every switch has a computer connected closely, only collecting this switch's data. These computers could be used to do the calculation of this switch, therefore we could get a quick result. A simpler system could be only one main data collector unit in the middle of the network (smallest connecting lag with all switches).

And we noticed that original scheme and our scheme have a same problem that such scheme need to recalculate the SPE threshold δ_{α}^{2} using full period data. This will also cause a heavy overload. Along with the data become larger and larger, the calculate would become longer and longer, finally it could not be done in one time interval. For example, if we continue calculate the data of time interval of one second, and last for one week, than we will get a matrix that the number of row is 604,800, about 600Kbs, then $X^T X$ is about 360Gbs, and each item is stored as type of Double of size 8 bits, so it will need 360Gbs to store matrix $X^T X$, not to mention PCA calculate need more space.
Therefore, we consider to use the threshold and normal subspace calculated in the former period, and directly use in next intervals. This is out of the consider that the network's normal subspace would not change rapidly in real world, so that we can use it straightly. In the opposite side, we can forecast that this scheme it difficult to handle network change.

V. Experiment and Result

In order to test the performance of our scheme, we do the following experiments to see how PCA, partition PCA and sample entropy perform under different situations.

A. Experiment

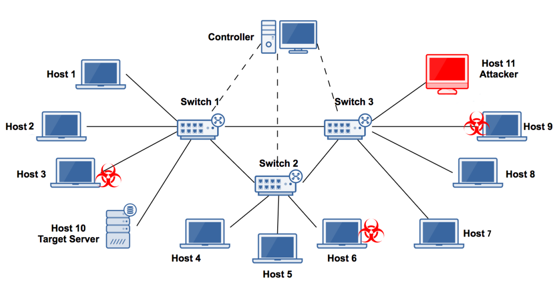


Fig. 2: The Topology In Our Experiment

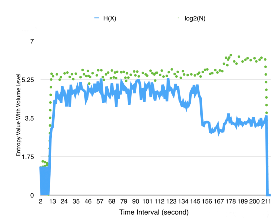
1) Experiment Setting: We set up a test environment by using Mininet creating a small scale of network of ring topology of three switches and 11 nodes (which could be a terminal or another network) directly connected to switch. There are 11 host in this network, so that there could be 11 original node, and 11 destination node, although the original node can not be the same with destination node. For each OD-pair (o, d) we could represent that:

$$(o, d), o = 1, 2, ..., 11, d = 1, 2, ..., 11, \text{and } o \neq d. \tag{12}$$

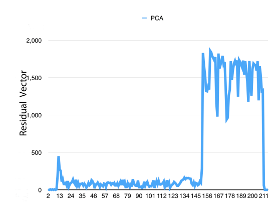
We choose such kind of network because of it is one kind of the most popular topologies, and switches could connect with each other directly.
And we generating dummy traffic by using Scapy, simulate normal traffic since time 0 and launch a DDoS attack at 180 second, and collect all data in 200 seconds (DDoS attack last for 20 seconds). Since these are 11 nodes, the OD flows number is 121 maximum, to get enough number of time interval, the time interval is set to "1 second", so that the number of time interval will be 200, and the $1-\alpha$ confidence level is set to 99%.
The topology is shown in Fig.2, each switch connected to another switches, and linked to 3 to 4 nodes. "Node 10" is set to be the victim server, and "node 3,6,9" are three zombie computers who will execute DDoS attack.

B. Comparison Between PCA and Sample Entropy

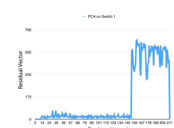
1) Sample Entropy: A general way for DDoS detection in SDN is conducted by collecting the flow statistics or traffic feature from the switches, and calculate the entropy measure randomness in the packets that are coming to a network. The higher randomness the higher is the entropy and vice versa. By setting a threshold, if the entropy passes it or below it, depending on the scheme, an attack is detected.
One metric that captures the degree of dispersal or concen-



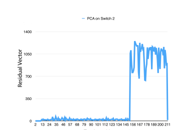
(a) Sample Entropy



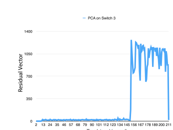
(b) PCA



(c) PCA on Switch 1



(d) PCA on Switch 2



(e) PCA on Switch 3

Fig. 3: Comparison Between Sample Entropy, PCA, and partition PCA with normal traffic doubled since time interval 173

tration is sample entropy. Assuming in one observation, total number of traffic is S, in which exists N OD-pairs (Origin-Destination Pairs), and n_i stands for the traffic amount of OD-pair i. Therefore OD-pair i will occur n_i times in this observation. So that $S = \sum_{i=1}^{N} n_i$. And sample entropy of this network is defined as following:

$$H(X) = -\sum_{i=1}^{N} \frac{n_i}{S} log_2 \frac{n_i}{S}. \tag{13}$$

The result $H(X)$ lies in the range $(0, log_2 N)$. It will takes the value 0 when the distribution is maximally concentrated, and takes the value $log_2 N$ when the distribution is maximally dispersed.

2) Comparison: We continually the experiment setting, and change some parameters to make a comparison between PCA and sample entropy.

We want to verify following questions: (i) whether the amount of normal traffic would impact the result, (ii) will these scheme detect when DDoS attack stop, and (iii) introduce a mutated DDoS attack aiming SDN, and to check whether these scheme are able to detect this attack.

Evaluation of DDoS

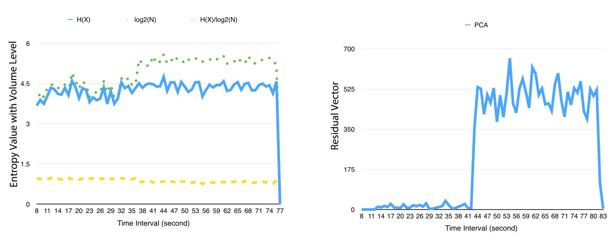
The result of sample entropy is not only affected by the distribution but also effected by the amount of OD-pairs appeared.

In this experiment, the time interval is set to each one second, the normal network traffic start at time interval 11, start the DDoS attacks at time interval 152, and double the amount of normal traffic at time interval 172. We want to see how the change of normal traffic would effect the result.

We analyze the data through sample entropy, conventional PCA scheme, and our partition PCA scheme on each switches, and get a result from Fig.3 that sample entropy, PCA and partition PCA all captured the DDoS attack. And the value of sample entropy slightly increased when the amount of normal traffic doubled. Meanwhile, the result calculated by PCA scheme and partition PCA scheme clearly separated with normal time interval, and didn't affected by the doubled normal traffic.

Evaluation of New Type DDoS attack

Sample entropy detect DDoS attack by that the destination IP is fixed, so that the entropy will decrease when DDoS attacks happened. But we can simply adjust the DDoS attack, that to set the destination IP randomly, so that sample entropy may not able to detect such attacks. So in this experiment, we start normal traffic from time interval 8, and launch mutated DDoS attack from time interval 43. And the result is shown in Fig.4. We can see that sample entropy can not handle this mutated attack, when PCA value level still change significantly.



(a) Sample Entropy (b) PCA

Fig. 4: Comparison Between Sample Entropy, PCA react when DDoS stop at time interval 173

VI. Conclusion

In this paper, we introduce a novel DDoS scheme using principal component analysis, to detect DDoS attack on SDN environment. Then, we put it into test, made a comparison with samply entropy, a popular used scheme. We show that this scheme have clearer result than another. Meanwhile, we identify a novel DDoS attack aiming on SDN environment, which could cause more damage on SDN, and used the two detection method on this novel DDoS attack, and found this novel attack is hard to be detected by sample entropy, and still be captured by PCA.

References

[1] "Software-defined networking (sdn) definition." https://www.opennetworking.org/sdn-resources/sdn-definition. Accessed: 2016-09-18.

[2] "Openflow switch specification 1.3.1." https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.1.pdf. Accessed: 2017-2-09.

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[4] P. Kazemian, M. Chang, H. Zeng, G. Varghese, N. McKeown, and S. Whyte, "Real time network policy checking using header space analysis," in *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pp. 99–111, 2013.

[5] A. Khurshid, X. Zou, W. Zhou, M. Caesar, and P. B. Godfrey, "Veriflow: verifying network-wide invariants in real time," in *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pp. 15–27, 2013.

[6] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu, "A security enforcement kernel for openflow networks," in *Proceedings of the first workshop on Hot topics in software defined networks*, pp. 121–126, ACM, 2012.

[7] S. Shirali-Shahreza and Y. Ganjali, "Rewiflow: Restricted wildcard openflow rules," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 5, pp. 29–35, 2015.

[8] M. Wang, H. Zhou, J. Chen, and B. Tong, "An approach for protecting the openflow switch from the saturation attack," 2016.

[9] G. Garg and R. Garg, "Detecting anomalies efficiently in sdn using adaptive mechanism," in *2015 Fifth International Conference on Advanced Computing & Communication Technologies*, pp. 367–370, IEEE, 2015.

[10] S. Hong, L. Xu, H. Wang, and G. Gu, "Poisoning network visibility in software-defined networks: New attacks and countermeasures.," in *NDSS*, 2015.

[11] P. Dong, X. Du, H. Zhang, and T. Xu, "A detection method for a novel ddos attack against sdn controllers by vast new low-traffic flows," in *Communications (ICC), 2016 IEEE International Conference on*, pp. 1–6, IEEE, 2016.

[12] R. Braga, E. Mota, and A. Passito, "Lightweight ddos flooding attack detection using nox/openflow," in *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*, pp. 408–415, IEEE, 2010.

[13] S. M. Mousavi and M. St-Hilaire, "Early detection of ddos attacks against sdn controllers," in *Computing, Networking and Communications (ICNC), 2015 International Conference on*, pp. 77–81, IEEE, 2015.

[14] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," in *ACM SIGCOMM Computer Communication Review*, vol. 35, pp. 217–228, ACM, 2005.

[15] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing network-wide traffic anomalies," in *ACM SIGCOMM Computer Communication Review*, vol. 34, pp. 219–230, ACM, 2004.

[16] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. D. Kolaczyk, and N. Taft, "Structural analysis of network traffic flows," in *ACM SIGMETRICS Performance evaluation review*, vol. 32, pp. 61–72, ACM, 2004.

[17] J. E. Jackson and G. S. Mudholkar, "Control procedures for residuals associated with principal component analysis," *Technometrics*, vol. 21, no. 3, pp. 341–349, 1979.

[18] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for openflow," in *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, pp. 3–3, 2010.

[19] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, *et al.*, "Onix: A distributed control platform for large-scale production networks.," in *OSDI*, vol. 10, pp. 1–6, 2010.