Improved Automated Graph and FCM Based DDoS Attack Detection Mechanism in Software Defined Networks

Xin Li[1], Zhijie Fan[2,3], Ya Xiao[2], Qian Xu[2], Wenye Zhu[2]

[1] Ministry of Public Security Key Laboratory of Safety Precaution Technology and Risk Assessment, People's Public Security University of China, China
[2] College of Electronics and Information Engineering, Tongji University, China
[3] Department of Information Security Technology, The Third Research Institute of The Ministry of Public Security, China
ndlixin@sina.com, aaronzfan@126.com, xiaoya_0922@hotmail.com, 1062942783@qq.com, 2013zhuwenye@tongji.edu.cn

Abstract

The DDoS attack is an unneglected cyber security threats in Software Defined Networks, specially it can have a fatal impact on SDNs. In this paper, we propose an improved automated graph based DDoS attack detection mechanism based on Fuzzy Cognitive Map (FCM) on SDNs. With the network patterns as nodes and similarity as link weights, our model based on feature-pattern graph and FCM is capable of detecting the DDoS attacks using graph based our method, and it can also scalable to insert new nodes to the graph model by graph update according to the unknown attack threats that are tried to find automatically. Our experimental results shown that the feasibility of our proposed method is a more accurate way to detect the DDoS attack by comparing with other similar methods.

Keywords: Software defined network, DDoS attack detection, Fuzzy cognitive maps, Graph model

1 Introduction

The Software Defined Network (SDN) has a especial network structure, it contains data plane, control plane and application plane. The data plane consists of OpenFlow switches and other network equipment like switch, which receives the commands from the controller and treats them as the programmed rules. The controller in control plane can command the forwarding actions of switches by programming thus simplifies the network management. The application plane provides APIs for the network management applications. With this open, centralized and programmable architecture, SDN brings the developers great convenience to experiment and deploy new idea.

However, the especial network structure in SDN also can expose some vulnerabilities lead to be attacked [1]. Attackers can launch denial of service attacks between the communication of controller and switch to make the network paralyzed. Once an SDN switch is compromised by an attacker, the flow table can be easily modified. Malicious applications and unauthorized application access may also appear on application plane. Among the well-known vulnerabilities of SDN, distributed denial of service (DDoS) attacks can have a devastating impact on the whole network, thus require more attention when building the network security mechanism on SDN.

There are several DDoS attack detecting mechanism in SDNs. First, the traditional statistical and signature based techniques need to be updated and adapted to the DDoS attacks in SDN environment. Since the size of Ternary Content-Addressable Memory (TCAM) that stores the flow rules is limited, how to collaboratively utilize and reduce the usage of TCAM is a challenge when the mechanism is deployed in SDN. Secondly, as the traffic in SDN are dynamic and the attack approaches are complex nowadays, it is difficult to use attack detection technology with only the traffic header information to discover different types of attacks. Third, there are increasing number of unknown attacks which SDN cannot prevent; thus, it is challenging to automatically detect the unknown attacks.

Considering the above problems, we propose an improved based DDoS attack detection mechanism based on automated graph model based on feature-pattern graph and fuzzy cognitive maps on SDNs according to our previous research. The proposed model in this paper is scalable to update and can be extended to other attack scenarios. We also use the DDoS attack dataset in traditional network and simulated attack traffics in SDN environment as signatures to build a feature-pattern and FCM graph (FPFG) for known attacks. The non-flow data in

*Corresponding Author: Zhijie Fan; E-mail: aaronzfan@126.com
DOI: 10.3966/160792642019122007010

traditional network is transferred into flow types which fit the SDN environment. To cope with the problem of limited TCAM, we directly monitor and collect the packets and parse them into flow table format instead of picking the flow table data from TCAM. Matric learning and Mahalanobis distance are used to model the similarity of different nodes in FPFG. With each node representing a network pattern and the weighted edges indicating the feature based similarity of nodes, the FPFG model can demonstrate the relationships of nodes properly. A graph based attack detection method is also proposed which utilizes not only the traffic header based features, but also the relation context of network patterns. The FPFG model is also scalable to update for the unknown attacks which are not in the predefined graph.

The rest of the paper is organized as follows. In Section 2, we introduce the background knowledge on SDN, discuss the security threats, and related research on DDoS attack detection in SDN. Section 3 proposes the detailed structure and procedures of our FPFG based DDoS attack detection mechanism. Section 4 presents our experiment and results to prove the effectiveness of the proposed mechanism. Finally, we conclude this paper and discuss the future work.

2 Related Work

2.1 Security Threats in SDN

In SDNs, network switches in data plane act as the forwarding devices and the controller in control plane implements the functionality and control logic. Numerous open-source SDN controllers, such as Floodlight, OpenDaylight, POX, Ryu [2], can be deployed in SDNs.

SDN has attracted considerable attention from the attackers because of the open framework. We use the taxonomy we utilized in [3] to illustrate the security threats in SDNs. And used FCM to analyze network security situation in [4]. Among all the threats on different plane of SDN, the DDoS attack is the most threatening one since it can be launched on various plane and can bring devastating impact on the whole SDN environment. Therefore, we mainly concentrate on the security mechanism on DDoS attack in this paper.

2.3 DDoS Attack Detection in SDN

DDoS attacks are aiming at reducing the target services by distributed multiple sources, have been critical threats in cloud security. The distinctive features of SDN offer new chances to detect and mitigate DDoS attacks. We explain it from four aspects according to our previous research [5] as follow.

When DDoS attack occurs, the randomness is reduced, leading to a lower value of entropy. Entropy-based methods calculate entropy and then detect the DDoS attack by filtering the entropy smaller than a threshold. Giotis et al. [6] defined four flow-related traffic features to calculate entropy: the source IP address (srcIP), the destination IP address (dstIP), the source port (srcPort) and the destination port (dstPort). When a DDoS attack occurs, the entropy on dstIP and dstPort have significant decrease. Based on this phenomenon, entropy-based calculations are periodically conducted on NOX controller to implement the anomaly detection algorithm. Wang et al. [7] proposed an entropy-based lightweight DDoS flooding attack detection model running in the OF edge switch. The information entropy of an edge switch is calculated, where each probability is approximately estimated by the frequency of each IP address. When the difference value between average entropy and real-time entropy is bigger than a threshold value, the DDoS alert is triggered. The entropy-based methods have a relatively low computation load and few limitations; however, when the values of features are continuous, relevant information about the feature distribution can be lost, which will lead entropy-based methods prone to errors in some scenarios [8].

Machine learning treats the attack detection as a classification problem, by learning the feature patterns to classify a new coming flow into attack or benign. [9] introduced a lightweight method for DDoS attack detection based on traffic flow features. A 6-tuple of features is proposed as the most likely six features to influence the decision of whether there is a DDoS attack in SDN. Self Organizing Map (SOM) is used to classify the network traffic. The six features include *average of packets per flow, average of bytes per flow, average of duration per flow, percentage of pair-flows, growth of single-flows* and *growth of different port*. [10] developed a deep learning based multi-vector DDoS detection system. They used stacked auto encoder(SAE) to classify the flows into types such as TCP, UDP, ICMP DDoS attack or benign with the 68 extracted features. Xu and Liu [11] proposed DDoS attack defense methods on victim detection and post-detection, leveraging SDN's flow monitoring capability. For each detection mission, four features are selected and SOM is used to classify the data by the similarity of statistical features. Although machine learning based methods are widely used with great success, their performance typically depends on the training datasets and selection of features.

Detecting traffic method makes the assumption that there is significant difference between the patterns of attack traffic and benign traffic. Yu et al. [12] used the traffic similarity to detect attacks from botnets, and they hypothesized that attack traffic from a botnet has high similarity which is approximately equal to 1, and the benign patterns come from different users thus have low similarity close to 0. Wang et al. [13] built a relational graph with known traffic patterns and their

labels, in which the maximum spanning tree is used to select features and maximum a posterior query is applied to detect attack. This method is effective, inexpensive and has low overhead. AlEroud and Alsmadi [14] proposed a Markov-based graph model with DoS attack pattern as nodes and their relationships as edges, in which the traffic features are used to describe the nodes. The traffic feature based similarity is used to find the nearest nodes as the traffic class. However, both [13] and [14] considered only the features that can be extracted directly from packet or flow content, which may not be sufficient in most circumstances.

It also used IDS/IPS detecting method, it relies on popular open-source network intrusion detection system (IDS) or intrusion prevention system (IPS), like Snort [15]. The mirroring-based network intrusion detection agent in NICE [16] utilized Snort to sniff the mirroring port on each virtual bridge in Open vSwitch and send alerts when suspicious or anomalous traffics are detected. A scenario attack graph and an alert correlation graph are constructed to analyze the vulnerabilities, alerts and traffics to select a proper countermeasure. To reduce the false alerts raised by Snort, NICE provided an approach to match if the alert is related to a vulnerability. Method in [17] also employed Snort to label the suspicious or benign type of flows. Aziz and Okamura [18] leveraged Suricata to detect the SMTP flood attack traffic and proposed FlowIDS as a supplement to collect the anomaly traffic omitted by Suricata. This method can help with the labeling of traffic, but relying highly on the third-party tools may cause the error propagation problem.

In this work, we improved our previous method by combining the entropy and traffic pattern analysis based on automated feature-pattern graph and fuzzy cognitive maps.

3 Proposed Model

3.1 The Structure of DDoS Attack Detection Method in SDN

Figure 1 shows the procedures of the proposed automated feature-pattern graph based DDoS attack detection mechanism for SDNs. Since AlEroud and Alsmadi [14] have proved the feasibility of using the existing attack signature to identify attacks on SDNs, the proposed model jointly uses the attack signatures extracted from traditional datasets and the data from SDN environment to build the attack signatures in SDN.

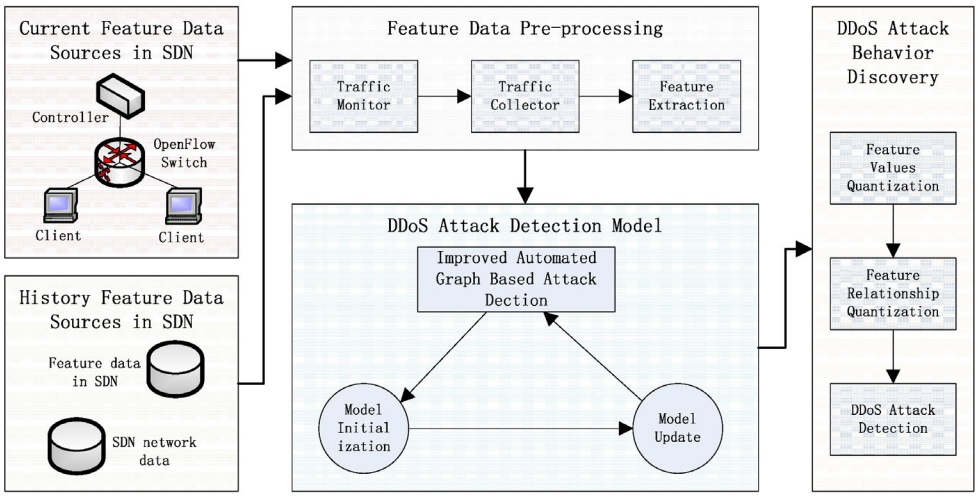


Figure 1. Procedures of the proposed model

Our method model is built with the attack signatures and also used to identify whether an incoming flow is malicious. Patterns are the network status such as *TCP DDoS attack* or *benign*, and features are those who can properly describe the status of network patterns, like *fraction of TCP flows*. Lines numbered as 1 in Figure 1 are the procedures of graph model generation. We directly monitor and collect the traffic packets in SDN environment and parse them into flow format. The packet data in traditional network is also transferred into flow type by flow creator. If the data is not classified, we need to label the data as different types in order to build the graph model. Lines with number 2 are processes of attack detection. The real-time traffic packets are monitored, collected and parsed into flow tables and then fed into the graph based attack detection model. If the new coming flow is malicious, a countermeasure is selected to prevent the SDN from further attack. When the graph based attack detection cannot detect a traffic, but the traffic tends to be malicious, the old graph needs to be updated and used for attack detection of subsequent flows (lines numbered as 3).

3.2 DDoS Attack Features Extractor in SDN

The OpenFlow protocol is accepted by Open

Networking Foundation (ONF) to be a standardized protocol in SDN southbound and northbound interfaces. It aims to define how an SDN controller communicates with the SDN switches. Flow table proposed in OpenFlow protocol is the foundation of querying and forwarding a package. The structure of flow table in OpenFlow protocol consists of six parts: match fields, priority, counters, instructions, timeouts and cookies. When a data packet arrives, the packet header will be compared with the match fields. if there is a match, the switch will update the counters and execute the actions, otherwise, the packet will be sent to the controller through the secure channel.

3.2.1 Attack Features Description and Definition

In our work, suppose that the flow entry information are sampled every T time period in OpenFlow switch, and the total number of samples is N, so the flow entry set is denoted as:

$Flow\_Entry\_Set=\{Protocl_i,\ ScrIP_i,\ DstIP_i,\ ScrPort_i,\ DstPort_i,\ Count_i,\},$

where $i = 1, 2, \ldots, N$.

We choose and focus on 8 typical attack features extractor to be the basis for detecting DDoS in software defined networks.

(1) The Growth Rate of Source IP Addresses (GRSIA)

Source IP addresses spoofing is an important behavior characteristic while the SDN is under DDoS attacked. Attackers always use different fake source IP addresses to launch attacks to prevent being tracked. When the DDoS attack occurs in SDNs, the growth rate of source IP address will change more frequently and be a higher level. The growth rate of source IP addresses can be obtained according to the following formula.

$$GRSIA = \frac{The\ Number\ of\ Source\ IP\ Addresses}{Time\ Interval\ T} \quad \textbf{(1)}$$

(2) The Growth Rate of Source Ports (GRSP)

When the SDN is under normal condition, the number of source ports is a relatively fixed value. When the attack occurs, the source ports will change more frequently than the normal time because the packages are generated by one attacker from random port. The growth rate of source ports will be a higher level while the SDN is under attacked. The growth rate of source ports can be obtained according to the following formula.

$$GRSP = \frac{The\ Number\ of\ Source\ Ports}{Time\ Interval\ T} \quad \textbf{(2)}$$

(3) The Growth Rate of Packets in per Flow (GRPF)

When the SDN is under DDoS attacked, the attacker will launch a large number packets to the specific host. The growth rate of packets in per flow will change more frequently than normal time and will be a higher level while the DDoS is launched by attacker. The growth rate of packets in per flow can be obtained according to the following formula.

$$GRPF = \frac{\sum_{i=1}^{N} PackNum_i}{Time\ Interval\ T'}, \quad \textbf{(3)}$$

where $PackNum_i$ is the i-th number of packets in time period T.

(4) The Average Number of Packets in per Flow (ANPF)

The Average number of packets in per flow is similar to the growth rate of packets in per flow, it is also an important feature to detect the DDoS attack behavior. When the DDoS attack occurs, most flow items have a small number of packets or bytes in order to increase the efficiency of attack. We use the middle value of the sorted packets as an average value in case some flows have a large number of packets to increase the average value.

$$mid(ANPF) = \begin{cases} F(\frac{n+1}{2})\ if\ n\ is\ odd \\ \frac{F(\frac{n}{2}) + F(\frac{n+1}{2})}{2}\ otherwise \end{cases}, \quad \textbf{(4)}$$

where $F(i)$ is the sequence of packets of each flow in ascending order, and n is the number of flows

(5) The Growth Rate of Flow Entries (GRFE)

When the attack occurs, the number of packages will grow rapidly, if we program the OpenFlow switch and record every packet flow in the flow table, we will see the rapid growth of the number of flows. The growth rate of flow entries can be obtained according to the following formula.

$$GRFE = \frac{The\ Number\ of\ Flow\ Entries}{Time\ Interval\ T} \quad \textbf{(5)}$$

(6) The Average of Number of Flow Entries (ANFE)

When the SDN is under DDoS attacked, source IP addresses spoofing attack is always launched to the specific host, the average of number of flow entries will change more frequently than the normal time and will be a lower level. The average of number of flow entries can be obtained according to the following formula.

$$GRFE = \frac{The\ Number\ of\ Flow\ Entries}{Time\ Total\ Number\ of\ Flow\ Entries\ N} \quad \textbf{(6)}$$

(7) The Percentage of Pair-Flow (PPF)

Pair-flow is the case when two flows, flow1 and flow2, satisfy the following three conditions, the flows do not fit these three conditions are called single-flows.

(a) The source IP of flow1 is the same as the destination IP of flow2.

(b) The destination IP of flow1 is the same as the source IP of flow2.

(c) Flow1 and flow2 have the same communication protocol.

The attacker will use a fake IP address, which leads to an increase of single-flows. The lower the percentage of pair-flows, higher is the possibility that SDN is attacked. The number of flows can be gained from flow stats in OpenFlow switch. The percentage of pair-flow can be obtained according to the following formula.

$$PPF = \frac{2 \times The\ Number\ of\ Pair\_Flow}{Time\ Total\ Number\ of\ Flow\ Entries\ N} \quad (7)$$

(8) The Rate of Flow Table Matching Accurately (RFTMA)

Under the normal conditions, the OpenFlow switch has a relatively fixed rate of flow table matching accurately. The packets arrived at OpenFlow switch are matched with original flow table in switch. When the attack occurs, a large number of new packets will be generated in switch, the rate of flow table matching accurately will be a lower level. The Rate of Flow Table Matching Accurately can be obtained according to the following formula.

$$RFTMA = \frac{The\ Number\ of\ Matched\ Flow\ Tables}{Time\ Total\ Number\ of\ Flow\ Tables} \quad (8)$$

3.2.2 Attack Features Values Quantization

To create the attack signatures in SDN, i.e., create flows for the packets which are not in flow format, we first need to install the flow rules based on the packet data from SDN or traditional network, and then label the flow based data into different patterns. Since TCP, UDP and ICMP are three most widely used protocols in network communication and DDoS attacks, we typically consider these three protocols in our research. A flow in TCP/UDP/ICMP is a group of packets having the same header fields. For TCP flows, the header fields contain protocol, source and destination MAC address, source and destination IP address, duration time, source and destination ports, flag, and packet size. For UDP flows, the header fields are similar to TCP flows except that UDP flows have no flag. The ICMP flows do not have source and destination ports but ICMP type and code.

Algorithm 1 shows the extracting the DDoS attack features in SDN. *Packets* is a list of packets need to be processed, $Header_{fields}$ is list of header fields which is predefined manually for each protocol. f_entry is a hashmap with the flow id as key and list of packets' indices as value, where the flow id is an auto increasing number from 0. f_entry_{width} is also a hashmap, where key is the flow id and value is the list of values of header fields. For each packet in *packets* list, we extract the header fields, and calculate the 8 DDoS attack features values according to our method in last section. Then we check if the fields already in the existing flow list, if so, update the corresponding list in *flow*; if not, create a new flow id and add the extracted fields in the f_entry_{width}. After traversing all the packets, the packets are classified by the flows and stored in *flow*, the fields of each flow are stored in f_entry_{width}. The patterns of flows depend on the pattern of corresponding packets; for example, if majority of packets in a flow are TCP flooding attack, then the flow is labeled as a TCP flooding attack.

Algorithm 1. Extracting DDoS Attack Features in SDN

```
1.  Input:
        Interval T; Number of Packets N;
        packets; header_fields;
2.  Initialize f[] and f_entry_width[] as two empty sets
3.  for i<-1 to N and t<-1 to T do
4.    for each packet in packets do
5.      extract header fields and store in list_t
6.      if list_t in f_entry_width.values then
7.        obtain f_id from f_entry_width
8.        calculate the 8 features values according
          to our formulas
9.        insert index of packet into f_entry[f_id]
10.     else
11.       insert lis_tt into f_entry_width
12.       f_entry[new f_id] = [index of packet]
13.     end if
14.   end for
15. end for
```

3.3 DDoS Attack Features Relationship Extractor in SDN

There are a lot of researches about different kinds of cyber attack detection in SDN, but there are relatively few researches about DDoS attack features relationship in SDN. Furthermore, after extracting the features in the target SDN network environment, most of detection methods independently calculate the values of different features at different time, and then complete the detection according to the different values of features. However, most of traditional detection methods always lack analysis of interrelations and restrictive correlations among different features, that usually exist in reality. In this article, we use Fuzzy Cognitive Maps (FCM) to quantitatively describe the interrelations and restrictive correlations between different DDoS attack features in SDN.

Fuzzy Cognitive Maps (FCM) is firstly proposed by

professor Kosko [19], who combines cognitive map with fuzzy set theory. FCM model consists of nodes, directed arcs and weights of directed edges. It is a weighted directed graph that can describe causal relationship. The node in FCM is called concept node that can describe the abstract things, concrete things, activities, system properties and system statuses according to actual demand. The weighted directed edges in FCM structure are used to describe the causal relationship between any two concept nodes. Directed edges can be viewed as single layer neural network with feedbacks and the object-oriented concept. The knowledge is inside of concept nodes and weighted directed edges. FCM model uses weighted directed relationships to simulate fuzzy reasoning, in which the interrelationships are used to stimulate dynamic behavior of system.

Definition of FCM: all concept nodes $c_1, c_2, \ldots, c_i, \ldots, c_n$ exist in FCM, the value's range of weighted directed edges is [-1, 1], e_{ij} is weight value of edge $<C_i, C_j>$, the matrix $E=f(e_{ij})$ is called an adjacent matrix or incidence matrix of FCM.

FCM can stimulate the operation states of system. The evolution process of FCM model includes forward and backward evolution. The forward evolution is mainly used for decision support and prediction, and backward evolution mainly is used for reason trace. In this article, we used the forward evolution to predict the cyber security situation. After building the FCM model and obtaining the initial status values of all concept nodes, the status values of all concept nodes at any time can be calculated according to forward evolution by the following formula.

$$A_i(t+1) = f(A_i(t) + \sum_{i=1, j\neq 1}^{n} A_j(t) \times w_i) \quad \textbf{(9)}$$

Suppose that $C=\{ c_1, c_2, \cdots, c_i, \cdots, c_n \}$ is a set of all concept nodes, $n=|C|$, and C_i is the value of the i-th concept node, recorded as A_i after mapping to range [0, 1], and it means the status value of concept node. $A_i(t)$ means the status value of i-th concept node at time t, and $A_i(t+1)$ means the status value of i-th concept node at time $t+1$. w_{ji} is the incidence matrix of concept nodes, also named as adjacent matrix. f is a function, the two or three valued step function and S-curve function are commonly used in practice.

Building an FCM model consists of several steps: selecting the concept nodes, connecting the causal relationship between any two concept nodes, and determining the impact degree of causal relationship. In this work, we choose all the indexes of third layer as the concept nodes in FCM, the causal relationship and adjacent matrix can be decided respectively by typical machine learning technology. In addition, we use the forward evolution procedure to predict cyber security situation and introduce threat intelligence data in similar network system to modify situation prediction.

3.4 Improved DDoS Attack Detection Method

After the extraction of DDoS attack features in SDN, we used our proposed attack detection module in [5] which is built on neighborhood classification in feature-pattern graph model. Furthermore, in this work, we used Fuzzy Cognitive Maps (FCM) to describe the relationships among the different DDoS attack features in SDNs. The module is divided into two sub modules, i.e., graph creation and detection engine.

3.4.1 Attack Detection Summary

Each node in our proposed graph model represents a type of DDoS attack or benign, i.e., network patterns. The DDoS attack patterns can be divided into three types on the basis of the protocol type, i.e., ICMP, UDP and TCP. Patterns can also be classified by experiences or according to the IDS alerts. The links among nodes are weighted, where the weight represents the similarity between two network patterns.

The link weight between two nodes in FPFG represents the feature based similarity. The Pearson Correlation is a widely used method in similarity measurement in intrusion detection system [14-20]; however, the Pearson Correlation may not be suitable for all the datasets. We use Mahalanobis distance and distance matric learning [21] to learn a proper distance formula given a set of DDoS attack signatures in SDN.

In reality, the actual attack pattern may not follow the pattern distribution as in the signature dataset, which means that the *FPFG* needs to be updated with new patterns. *FPFG* can be updated manually, which means that the administrator can manually insert new nodes in *FPFG*. Another type is to update automatically, which can be divided into local and global update.

When a new node V_{new} is discovered, local update is to insert the new node V_{new} into existing graph. First, we calculate the similarity between V_{new} and other existing nodes, and then find the γ -nearest nodes whose distances are significantly smaller than others. If γ is larger than 1, then we check whether the γ -nearest nodes have links among themselves, and new weighted links will be generated between V_{new} and the nodes who connect with each other. Global update is to relearn the matric matrix $\boldsymbol{M}$ and regenerate the entire graph. This can consume much more time and computing resources than the local one, but can obtain better performance on the following attack detection.

To detect an unknown DDoS attack pattern, i.e., a new node, both the similarity and entropy based methods are used. If a new pattern of DDoS attack occurs, the randomness will decrease, traffic will have high similarity and low entropy. When a group of traffics cannot be classified into existing attack patterns, and the Mahalanobis distance based similarities among

new traffics are higher than a threshold value and entropy of new traffic is lower than a threshold, then a new type of DDoS pattern need to be added into the *FPFG*.

3.4.2 Attack Detection Details

We used our proposed DDoS attack detection module in [5], that is used to detect the pattern, i.e., DDoS type or benign, of an incoming OpenFlow based flow according to the weighted *FPFG* at run time. In the process of DDoS attack

Algorithm 2 shows the steps of detecting the pattern of a new incoming flow. It needs two inputs as showed in Figure 2, one is the weighted feature-pattern graph which is built in the previous part, containing the nodes, weighted links and the feature vector for each node, another input is the $1\times k$ dimensional feature vector of the new incoming flow, denoted as $\boldsymbol{F}_{new} = [f_1^{(new)}\ f_2^{(new)} \cdots f_k^{(new)}]$. The distance between $\boldsymbol{F}_{new}$ and each node or distance between $\boldsymbol{F}_{new}$ and each feature are stored in hashmap *Dist* . Keys of *Dist* are the indices of nodes, and values of *Dist* are distance values or lists of distances. First, we turn the $1\times k$ dimensional feature vector $\boldsymbol{F}_i$ into $1\times k$ dimensional vector $\boldsymbol{F}_i'$ by calculating the entropy of each feature. By calculating the Mahalanobis distance between $\boldsymbol{F}_{new}$ and different nodes to find the minimum distance. If the minimum is significantly smaller than other distances, which means this method can classify the new incoming flow into one attack pattern, the attack pattern node who has the minimum distance is the predicted pattern of $\boldsymbol{F}_{new}$.

Algorithm 2. DDoS Attack Detection Algorithm

1. **Input:** *FPFG*: $\{V, E, F\}$; $\boldsymbol{F}_{new} = [f_1\ f_2 \cdots f_k]$
2. Initialize $Dist = \{\phi\}$
3. **for** each $\boldsymbol{F}_i$ in $\boldsymbol{F}$ **do**
4. $\quad \boldsymbol{F}_i' = get\_entropy\ (\boldsymbol{F}_i)$
5. **end for**
6. $\boldsymbol{F}_{new}' = \{F_1', F_2', \cdots, F_n'\}$
7. **for** each $\boldsymbol{F}_i'$ in $\boldsymbol{F}'$ **do**
8. $\quad dist(i, new) = dist_{mah}^2(\boldsymbol{F}_i', \boldsymbol{F}_{new})$
9. $\quad Dist[i] = dist(i, new)$
10. **end for**
11. $i = \min(Dist).index$
12. **if** $Dist[i]$ is significantly smaller than other items in *Dist* **then**
13. $\quad$ **Return** node V_i as the attack pattern
14. **else**
15. $\quad nodes_{nearest} = [list\ of\ \beta - nearest\ nodes\ with\ \boldsymbol{F}_{new}]$
16. $\quad Dist = \{key : value\}$
17. $\quad$ **for** each node V_j in $Nodes_{nearest}$ **do**
18. $\quad\quad$ **for** each row r in $\boldsymbol{F}_j$ **do**
19. $\quad\quad\quad$ calculate $dist(f, new)$ according to formula (9)
20. $\quad\quad$ **end for**
21. $\quad Dist[j] = [list\ of\ top - \alpha\ shortest\ dist\ (f, new)]$
22. $\quad$ **end for**
23. $\quad$ **Return** the smallest $sum\ (Dist[j])$, V_j is the attack pattern
24. **end if**

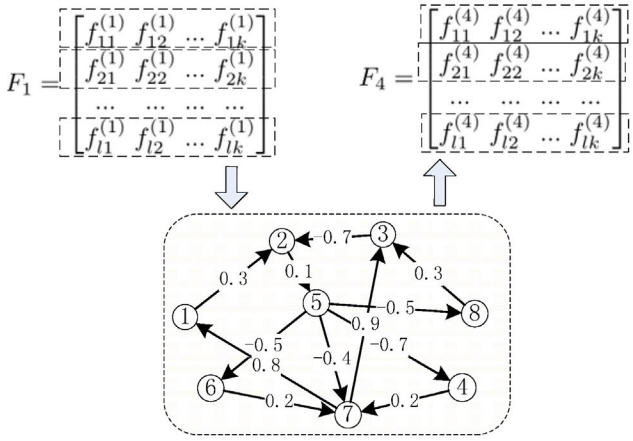


Figure 2. Improved calculating the distances between Fnew and each feature

After detecting the pattern of flows, the countermeasures should be taken. Countermeasures can be divided into forward, modify, or drop by the reacting method. For example, the benign flows will be forwarded to the target port; flows who have the same target port at the same time will be modified and redirected to other ports; flows that have no predefined countermeasures will be dropped. The countermeasures we defined include: flow redirection, flow isolation, deep packet inspection, creating filtering rules, MAC address change, IP address change, block port, quarantine, network reconfiguration, network topology change

4 Experiment

4.1 Instruction of Experimental Environment

We use the previous same experiment method to verify our method in this work. The test scenarios are performed by simulation to prove the feasibility and effectiveness of the proposed method. Figure 3 shows the network topology which is created to simulate the SDN environment. Mininet [22] is used to create a realistic virtual SDN and run example topologies based

on Ryu controller and OpenFlow switch. Ryu [23] is a component-based SDN framework written in Python which supports OpenFlow. The well-defined APIs can be used for network prototyping, management and monitoring. Open vSwitch is a widely used SDN OpenFlow switch which supports the OpenFlow protocol. The network1 in Figure 3 acts as a botnet, and the hosts simulate the DDoS attack traffics by sending packets to OpenFlow switch and Ryu controller with a packet manipulation program Scapy [24]. Network2 simulates normal traffics. We use Wireshark connection monitoring system to capture the network traffic between hosts and the controller.

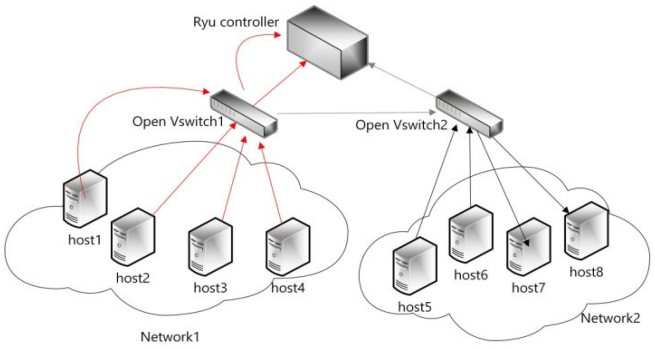


Figure 3. Network topology for test senario

4.2 Dataset and Implementation

The datasets we use for evaluation contain two parts. One is the Intrusion detection evaluation dataset (ISCXIDS2012) [25], named dataset1, which contains the network traffic collected during the time periods with DDoS attack using an IRC botnet. The traffic flows of DDoS attack are labeled as attack or normal, which are separated into two sets for training and testing, respectively, in the experiment. Data in dataset1 is organized in network packet format, so we need to transfer the packet-based data into flow-based data according to the procedure in Section 3.1. Another part of dataset, i.e., dataset2, is created by our test scenario shown in Figure 3. We generate DDoS attack traffic and normal traffic at different time periods so that the traffic will be automatically classified into attack or benign. The statistics of datasets used for training and testing is shown in Table 1. The implementation procedures are described as follows.

Table 1. Statistics of dataset for training and testing

Type	Dataset1		Dataset2	
	Train	Test	Train	test
Attack	638440	370961	61230	7389
Benign	266568	134880	134720	8444

The attack usually lasts for a time period, so we need to analyze the features of traffics at different time periods. The creation of flow table with flow-based data is necessary for feature extraction. A flow table is defined as a set of flows that are generated during a same time period, and the time interval is set to 10 seconds in this experiment. According to the generation time of packets in each flow, flows whose packet timestamps located in one time slot are categorized into the same flow table.

We define different features for DDoS attack patterns on TCP, UDP and ICMP, respectively. Each flow table is turned into a 42-dimensional feature vector after feature extraction. TCP based DDoS attack contains 21 features: fraction of TCP flows, fraction of symmetric TCP flows, number of distinct source IP, entropy of source IP, median of bytes per flow, median of packets per flow, number of distinct window size, entropy of window size, number of distinct TTL values, entropy of TTL values, number of distinct source ports, entropy of source port, number of distinct destination ports, entropy of destination ports, fraction of destination ports less than 1024, fraction of flows with SYN flag, fraction of flows with ACK flag, fraction of flows with URG flag, fraction of flows with FIN flag, fraction of flows with RST flag, fraction of flows with PSH flag. UDP based DDoS attack contains 13 features: fraction of UDP flows, fraction of symmetric UDP flows, number of distinct source IP, entropy of source IP, median of bytes per flow, median of packets per flow, number of distinct source ports, entropy of source port, number of distinct destination ports, entropy of destination port, fraction of destination ports less than 1024, number of distinct TTL values, entropy of TTL values. Based on the features extracted, we use the method proposed to calculate the distance matrix $\boldsymbol{M}$. By computing the Mahalanobis distance between each pair of nodes, a weighted graph is generated.

We evaluate the attack detection performance by launching a binary class prediction, i.e., attack or normal, and a multi-class prediction, i.e., normal or attack based on TCP, UDP, ICMP or combination of three. The binary classification used the dataset with binary labels to learn the metric matrix, while multiple classification used the dataset with multiple labels. Two methods are used in prediction to evaluate the prediction performance, as described. One is to calculate the similarity between new coming flow feature and each node to find the smallest distance, abbreviated as *SFN*. Another is to compute the similarity between new flow feature and each feature in β -nearest nodes, abbreviated as *SFF*; this method is more time consuming but has higher accuracy than the previous one.

4.3 Experiment Result

4.3.1 Attack Detection

We measure the effectiveness of the proposed attack detection approach in terms of Precision (*P*), Recall (*R*), and F-score(*F*) [5]. Both *P* and *R* are calculated by the value of True Positive (*TP*), False Positive (*FP*), True

Negative (*TN*) and False Negative (*FN*), where *P* is the fraction of predicted pattern which matches the corresponding flow, *R* is the fraction of target flow's pattern values which are correctly predicted, and *F* is the harmonic mean of *P* and *R*.

Binary class prediction and multi-class prediction are launched in the attack detection procedure. Binary classification is to predict the attack or normal pattern of a new incoming flow. The multi-class prediction represents 5 classes here, i.e., normal pattern with the majority of TCP flows, normal pattern with the majority of UDP flows, normal pattern with both TCP and UDP flows, TCP flooding DDoS attack, and combination of both TCP and UDP flooding DDoS attack.

We compare our proposed method (*FPFG*) with three other methods on distance calculation, which are KLD [26], Pearson [14-20] and Cosine [27]. The precision, recall and F-score of attack detection in Table 2. The results show that our proposed method generally outperforms the compared methods. Which indicates that by utilizing combined feature-pattern graph, fuzzy cognitive maps with matric learning and similarity on feature vectors, our model can better predict the pattern of a new flow thus can better detect the DDoS attacks.

Table 2. Average precision, recall and F-score of multi-class prediction

	Average Precision		Average Recall		F-score	
	SFN	SFF	SFN	SFF	SFN	SFF
FPFG	0.155	0.657	0.107	0.428	0.101	0.472
KLD	0.401	0.323	0.021	0.131	0.118	0.170
Pearson	0.300	0.569	0.304	0.252	0.351	0.393
Cosine	0.462	0.581	0.176	0.266	0.317	0.426

4.3.2 Graph Update

The other part of the experiment is to measure the effectiveness of the updated graph model. We add a new node by our update method and measure the performance.

The graph we generated before update is shown in Figure 4 with black nodes and edges.

The node V_5 in Figure 4 is a new node added to the graph. First step, we generate a test dataset of DDoS attack with UDP traffic, which is not in the existing feature-pattern graph; so, we added a new node, i.e. V_5, in the graph. The local update is to measure the traffic feature similarity of new pattern and the existing patterns to find the γ -nearest nodes and establish links. During the update, we found only the distance between V_5 and V_4 are significantly smaller than other distances, so the first step update only inserts one link. Second step is to relearn the matrix $\boldsymbol{M}$ and recalculate the Mahalanobis distance between nodes. The second step update can find the latent relationships and similarities between new node and existing nodes more accurately.

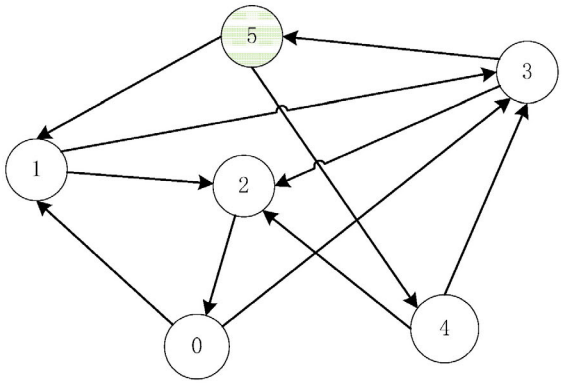


Figure 4. Graph generated by local and global update

The precision, recall and F-score of the *SFF* method after various updates are shown in Table 3. The three nodes in each line represent no update, local update and global update, respectively. We notice a minor decline in these three values after local update; this is because that the local update cannot precisely find the links between new node and existing nodes, which leads to the wrong prediction of new pattern and decline of precision and recall values. However, after the global update, the values of precision and recall increase and go back to the value before update.

Table 3. Performance of detection after graph update

	Precision	Recall	F-Score
Before update	0.652	0.454	0.350
After update	0.635	0.426	0.325

5 Conclusion

In this work, we propose an improved automated graph based DDoS attack detection mechanism based on Fuzzy Cognitive Map (FCM) in SDNs in this paper. With the network patterns as nodes and similarity as link weights, our model based on feature-pattern graph and FCM is capable of detecting the DDoS attacks using graph based our method, and it can also scalable to insert new nodes to the graph model by graph update according to the unknown attack threats that are tried to find automatically. We create the attack signatures adapted to SDN environment by flow creation with the dataset from traditional network and SDN traffic. A feature-pattern graph is modeled based on the attack signatures. To better depict the relationships among network patterns, we introduce the link weight learning and transfer it to fuzzy cognitive maps (FCM). The graph model is scalable and we can update the graph in our method.

In the part of experiment, we use the previous same

experiment method to verify our method in this work. Experiments are conducted to evaluate the performance of proposed mechanism. A simulated SDN environment is established to monitor and collect SDN traffic. The results shown that the feasibility of our proposed method is a more accurate way to detect the DDoS attack by comparing with other similar methods on precision, recall and F-score.

Acknowledgements

This work is supported in part by The Beijing Natural Science Foundation Program 4184099 and the National High Technology Research and Development Program of China under Grant 2015AA016009.

References

[1] S. Scott-Hayward, G. O'Callaghan, S. Sezer, Sdn Security: A Survey, *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, Trento, Italy, 2014, pp. 1-7.

[2] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, S. Uhlig, Software-defined Networking: A Comprehensive Survey, *Proceedings of the IEEE*, Vol. 103, No. 1, pp. 14-76, January, 2015.

[3] Z. Fan, Y. Xiao, A. Nayak, C. X. Tan, An Improved Network Security Situation Assessment Approach in Software Defined Networks, *Peer-to-Peer Networking and Applications*, Vol. 12, No. 2, pp. 295-309, March, 2019.

[4] Z. Fan, Z. Tan, C. Tan, X. Li, An Improved Integrated Prediction Method of Cyber Security Situation Based on Spatial-Time Analysis, *Journal of Internet Technology*, Vol. 19, No. 6, pp. 1789-1800, November, 2018.

[5] Y. Xiao, Z. Fan, A. Nayak, C. Tan, Discovery Method for Distributed Denial of Service Attack Behavior in SDNs Using a Feature-pattern Graph Model, *Frontiers of Information Technology & Electronic Engineering*, Vol. 20, pp. 1195-1208, September, 2019. https://doi.org/10.1631/FITEE.1800436

[6] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, 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.

[7] R. Wang, Z. Jia, L. Ju, An Entropy-based Distributed Ddos Detection Mechanism in Software-defined Networking, *2015 IEEE Trustcom/BigDataSE/ISPA*, Helsinki, Finland, 2015, pp. 310-317.

[8] P. Fiadino, A. D'Alconzo, M. Schiavone, P. Casas, Challenging Entropy-based Anomaly Detection and Diagnosis in Cellular Networks, *ACM SIGCOMM Computer Communication Review*, London, United Kingdom, 2015, pp. 87-88.

[9] R. Braga, E. Mota, A. Passito, Lightweight Ddos Flooding Attack Detection Using Nox/Openflow, *2010 IEEE 35th Conference on Local Computer Networks (LCN)*, Denver, CO, USA, 2011, pp. 408-415.

[10] Q. Niyaz, W. Sun, A. Y. Javaid, A Deep Learning Based Ddos Detection System in Software-defined Networking (sdn), *EAI Endorsed Transactions on Security and Safety*, Vol. 4, No. 12, Article e2, December, 2017.

[11] Y. Xu, Y. Liu, Ddos Attack Detection under Sdn Context, *The 35th Annual IEEE International Conference on Computer Communications*, San Francisco, CA, USA, 2016, pp. 1-9.

[12] S. Yu, S. Guo, I. Stojmenovic, Can We Beat Legitimate Cyber Behavior Mimicking Attacks from Botnets?, *2012 Proceedings IEEE INFOCOM*, Orlando, FL, USA, 2012, pp. 2851-2855.

[13] B. Wang, Y. Zheng, W. Lou, Y. T. Hou, Ddos Attack Protection in the Era of Cloud Computing and Software-defined Networking, *Computer Networks*, Vol. 81, pp. 308-319, April, 2015.

[14] A. AlEroud, I. Alsmadi, Identifying Cyber-attacks on Software Defined Networks: An Inference-based Intrusion Detection Approach, *Journal of Network and Computer Applications*, Vol. 80, pp. 152-164, February, 2017.

[15] M. Roesch, Snort: Lightweight Intrusion Detection for Networks, *Proceedings of the 13th USENIX Conference on System Administration*, Seattle, Washington, USA, 1999, pp. 229-238.

[16] E. Albin, N. C. Rowe, A Realistic Experimental Comparison of The Suricata and Snort Intrusion-Detection Systems, *26th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, Fukuoka, Japan, 2012, pp. 122-127.

[17] C.-J. Chung, P. Khatkar, T. Xing, J. Lee, D. Huang, Nice: Network Intrusion Detection and Countermeasure Selection in Virtual Network Systems, *IEEE Transactions on Dependable and Secure Computing*, Vol. 10, No. 4, pp. 198-211, July-August, 2013.

[18] M. Z. A. Aziz, K. Okamura, Leveraging Sdn for Detection and Mitigation Smtp Flood Attack through Deep Learning Analysis Techniques, *International Journal of Computer Science and Network Security*, Vol. 17, No. 10, pp. 166-172, October, 2017.

[19] B. Kosko, Fuzzy Cognitive Maps, *International Journal of Man-Machine Studies*, Vol. 24, No. 1, pp. 65-75, January, 1986.

[20] Q. Wu, D. Ferebee, Y. Lin, D. Dasgupta, An Integrated Cyber Security Monitoring System Using Correlation-based Techniques, *IEEE International Conference on System of Systems Engineering*, Albuquerque, NM, USA, 2009, pp. 1-6.

[21] C. Shen, J. Kim, L. Wang, Scalable Large-margin Mahalanobis Distance Metric Learning, *IEEE Transactions on Neural Networks*, Vol. 21, No. 9, pp. 1524-1530, September, 2010.

[22] R. L. S. Oliveira, C. M. Schweitzer, A. A. Shinoda and L. R. Prete, Using mininet for emulation and Prototyping Software-defined Networks, *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)*, Bogota, Colombia, 2014, pp. 1-6.

[23] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, R.

Smeliansky, Advanced Study of Sdn/Openflow Controllers, *Proceedings of the 9th central & eastern european software engineering*, Moscow, Russia, 2013, Article No. 1.

[24] T. H. Kobayashi, A. B. Batista, A. M. Brito, P. S. Pires, Using a Packet Manipulation Tool for Security Analysis of Industrial Network Protocols, *2007 IEEE Conference on Emerging Technologies and Factory Automation*, Patras, Greece, 2007, pp. 744-747.

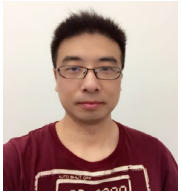
[25] A. Shiravi, H. Shiravi, M. Tavallaee, A. A. Ghorbani, Toward Developing a Systematic Approach to Generate Benchmark Datasets for Intrusion Detection, *Computers & Security*, Vol. 31, No. 3, pp. 357-374, May, 2012.

[26] T. Van Erven, P. Harremos, Rényi Divergence and Kullback-leibler Divergence, *IEEE Transactions on Information Theory*, Vol. 60, No. 7, pp. 3797-3820, July, 2014.

[27] H. V. Nguyen, L. Bai, Cosine Similarity Metric Learning for Face Verification, *Asian Conference on Computer Vision*, Queenstown, New Zealand, 2010, pp. 709-720.

Biographies

Xin Li received the Ph.D. degree in Zhejiang University, Hangzhou, China in 2006. He is currently the Professor of Computer Science at People's Public Security University of China. His research interests include big data, cyber security and video content analysis.



Zhijie Fan received his M.S. and Ph.D. degree in Engineering from Zhejiang University and Tongji University, China, in 2009 and 2019. He is currently the researcher of computer science at The Third Research Institute of Ministry of Public Security. His research interests include cyber security, cloud computing security, mobile security and communication security.

Ya Xiao received the B.S. degree in School of Computer Science and Engineering, Tongji University, Shanghai, China, in 2015. She is currently pursuing her Ph.D. degree in Tongji University of Computer Science and Engineering, Shanghai, China. Her research interests include cyber security, social networking and natural language processing.

Qian Xu received his M.S. degree from the Department of Computer Science and Technology, Tongji University, China, in 2012. He is now pursuing the Ph.D. degree at the Department of Computer Science and Technology, Tongji University. His research interests include cryptography and cloud security.

Wenye Zhu received the bachelor's degree from the Department of Computer Science, Tongji University, Shanghai, China in 2013, where he is now pursuing his Ph.D. degree.His research interests include threat intelligence, mobile security and private information retrieval.