TDDAD: Time-Based Detection and Defense Scheme Against DDoS Attack on SDN Controller

Jie Cui[1,2], Jiantao He[1,2], Yan Xu[1,2], and Hong Zhong[1,2(✉)]

[1] School of Computer Science and Technology, Anhui University, Hefei 230039, China
cuijie@mail.ustc.edu.cn, Hjt_1994@outlook.com, {xuyan,zhong}@ahu.edu.cn
[2] Anhui Engineering Laboratory of IoT Security Technologies, Anhui University, Hefei 230039, China

Abstract. Software defined network (SDN) is the key part of the next generation networks. Its central controller enables the high programmability and flexibility. However, SDN can be easily disrupted by a new DDoS attack which triggers enormous Packet_IN messages. Since the existing solutions focus on checking current network states with content feature to detect the attack, they can possibly be misled. In this paper, we propose a detection and defense scheme against the DDoS attack based on the time feature. Specifically, the time feature is the hit rate gradient of the flow table. We first extract the temporal behavior of an attack. A back propagation neural network is trained to extract an attack pattern and used to recognize an attack. Then either a defense or recovery action will be taken. We test our scheme with the DARPA 1999 intrusion detection data set and compare our scheme with another method using sequential probability ratio test (SPRT). The experiment and evaluation show that our scheme enables the real-time detection, effective defense and quick recovery from DDoS attacks.

Keywords: DDoS · SDN · BPNN · Time feature · Dynamic recovery

1 Introduction

Software defined network (SDN) is a new network architecture that separates the control and data planes. A central control plane enables the global view of the network, which makes SDN programmable and more flexible than a traditional network. Using SDN can drastically increase the throughput and forwarding speed in IoT [3] or DCN [1]. However, there are a variety of security concerns in SDN [2]. One problem is that the central data plane of SDN may lead to the single point failure, especially when a DDoS attack happens [9].

Especially, there exists an SDN-specific DDoS attack against SDN controller. Different from the traditional DDoS attack, an attack against the controller aims to overload the controller and disrupt the entire network. It utilizes the operations defined by the OpenFlow protocol [11] to launch an attack. In SDN,

W. Susilo and G. Yang (Eds.): ACISP 2018, LNCS 10946, pp. 649–665, 2018.
https://doi.org/10.1007/978-3-319-93638-3_37

the OpenFlow (OF for short) switch encapsulates the packet that matches no flow entry into a Packet_IN message and sends it to the controller. The controller decapsulates the Packet_IN message, then calculates the route and finally pushes a flow entry to the OF switch. To launch a DDoS attack, an attacker only need to send enormous fake packets to the OF switch. Since the packets are fake, they match no existing flow entry and trigger enormous Packet_IN messages. Finally, the controller resources for handling the Packet_IN messages will be exhausted [15].

Normally, the attacker sends random packets to the OF switch as long as each packet has a different IP address or port from others. Because the attacker can combine different attacks, such as SYN Flooding, ICMP Flooding, and UDP Flooding [19], to confuse us, the traditional detection method may be unsuitable for the hybrid attack. Meanwhile, the attacker uses the OF switch to attack the controller instead of attacking the controller directly. Therefore, the packets can be divided and sent to serval switches, but the Packet_IN messages shall still be sent to the controller automatically. Thus, it may not work to detect the attack against the controller with the traditional method in a single switch.

1.1 Related Work

As SDN is likely to become the heart of the next generation networks, security issues in SDN become very important. There are many studies on securing SDN with a focus on detecting DDoS attacks in SDN.

In 2010, Rodrigo Braga et al. [4] proposed a method for detecting a DDoS attack using OpenFlow. They combined the statistics from an OF switch with a self-organized map (SOM) to detect a DDoS attack. By using this method, they detected the DDoS attack without the specific middleware and got a good accuracy. However, this method was designed for detecting a traditional DDoS attack but not a DDoS attack against the controller. It is unknown to us whether this method work to detect a DDoS attack against the controller.

Though studies on the traditional DDoS detection are still active [5,17,20], many researchers realize that the detection of the DDoS attacks against a controller is more important. Thus, plentiful methods for detecting a DDoS attack against controller have been proposed [7,8,14,16,18].

Mousavi and St-Hilaire [13] proposed an entropy-based detection of the DDoS attacks against SDN controllers in 2015. They used the entropy of IP addresses to detect an attack. When an attack occurs, the attacker sends enormous fake packets to the OF switch. Consequently, the entropy of incoming packets increases sharply, which can be used to detect the attack. This method shows a great promptness, but it may make a misjudgment when the attacker tries to confuse it [6]. It is not a universal method though it shows a great promptness.

The hit rate of a flow entry can be simply represented by the amount of the packets that match the flow entry in a fixed time. For example, if a flow entry is hit 20 times in 1 second, then its hit rate is 20. One primary characteristic of the attack is that almost every packet has different IP address from another, which causes the low rate of the flow table. Ping Dong et al. [6] proposed a detection

method with SPRT. They defined a flow entry with an extremely low hit rate as "low-traffic flow". First, they used SPRT to analyze the DARPA 1999 intrusion detection data set. Then, they used SPRT with the experienced parameters to detect an attack. This method reduces the possibility of misjudgment since the normal flows are always continuous but the low-traffic flows are not. It is a universal method to detect the attack against controller, but they still ignored the temporal characteristics of a DDoS attack, which could have achieved a faster detection.

In all methods mentioned before, the feature used to detect an attack is always a content feature, such as the entropy and the statistics from OF switch. Thus, the attacker can confuse these detection systems by changing the contents of malicious packets, such as the distribution of the IP addresses [6]. We notice that the principle of the attack is to trigger massive Packet_IN messages, thus the attack must cause a sharp decrease in the hit rates. This change is an inherent behavior feature of the attack. Therefore, we can use the gradients of hit rate as the time feature to detect the attack.

1.2 Our Contributions

In this paper, we propose a method to detect and defend the DDoS attack against controller based on time feature. It relies on the detection of the attacking behavior according to the time feature. First, we collect statistics from the OF switch, such as the count and duration of each flow entry. Then we calculate the hit rate with the count and duration of flow entry. The hit rate gradient is the gradient between two successive hit rates and it shows the changing rate of the hit rates. For example, if the hit rate changes from 2 to 3, the hit rate gradient is 1.5. With the multiple successive hit rates, we can calculate hit rate gradients as the time feature. Then we use the time feature to train a BPNN. Finally, a DDoS attack can be detected with the BPNN, and the appropriate response will be taken. In the comparison with the methods mentioned before, [6,13] particularly, our method has the following advantages:

- First, we use the temporal feature to predict the next state of the network in some respects. By contrast, the methods mentioned before use the current network state to detect a DDoS attack. For example, method [6] use low-traffic flow, which is a current state of flow table, to detect the attack. Thus, our method can detect the attack earlier.
- Second, the temporal feature is also the inherent behavior feature of the attack and we use it to detect the attack. It reduces the false positive (FP) rate caused by attacker's strategy. The method [13], for example, can be confused by the attacker and make a misjudgment.
- Finally, we implement a defense method and a dynamic port recovery mechanism, which makes our scheme more complete than method [13] and method [6]. The defense method interdicts attack flows effectively, while the port recovery mechanism decreases the negative effect on the legitimate services.

We have to admit that there are still shortages in our scheme. BPNN is a classical machine learning algorithm. Compared with other algorithms, it shows some limitations due to its simple structure. Though our defense method can effectively protect the controller, it causes the unavailability of the legitimate services sometimes. The time to recover a victim port is hard to predict and control. Fortunately, the experiment and evaluation show that our scheme is still feasible.

The rest of this paper is organized as follows. In Sect. 2, we are going to introduce the OpenFlow protocol and the BPNN. Details of the method and experiment are shown in Sects. 3 and 4. Finally, we will present conclusions in Sect. 5.

2 Background

The OpenFlow protocol is the key protocol of SDN while BPNN is a useful algorithm in the field of pattern recognition. Attacker utilizes the principle of OpenFlow to launch an attack and we use the BPNN to detect this attack, so it is necessary to introduce the OpenFlow and BPNN first.

2.1 OpenFlow

The OpenFlow protocol was proposed by professor N. McKeown in 2008. It suggests separating a network into the control plane and the data plane independently, communicating via a secure channel. The overview of OpenFlow architecture is shown in Fig. 1.

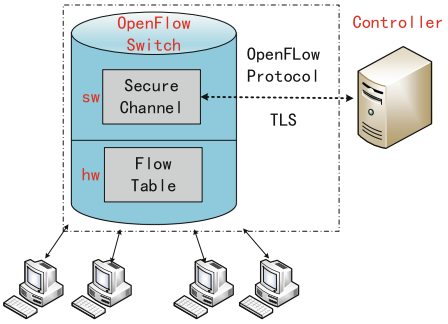

Fig. 1. OpenFlow architecture

In an OF switch, packets are always handled as the "flows". When a packet arrives at an OF switch, the switch tries to match the packet with the flow table which contains many flow entries. If the packet matches any flow entry, the switch will handle the packet as the instruction and update the counter of

the flow entry. The packet that matches no flow entry should be encapsulated into Packet_IN message and sent to the controller. Controller makes a decision, then pushes a flow entry to the OF switch. The process of handling a packet in the OF switch is shown in Fig. 2.

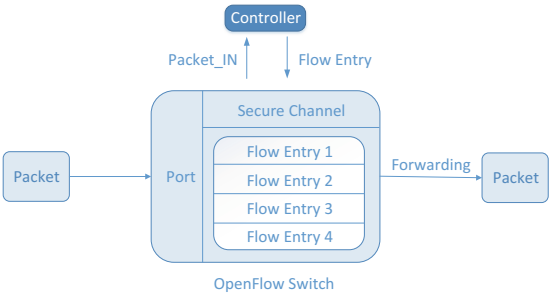

Fig. 2. Handle process of packet in SDN

According to the OpenFlow 1.0, a flow entry is mainly divided into three parts: match field, counter, and instructions. The match field consists of many segments from a packet, such as IP address, MAC address, port, type of protocol etc. The match field is used to match packet and determine which instruction should be chosen. The counter stores statistics of the packets matching this flow entry. As for the instructions, they tell the switch what kind of action should be taken, such as dropping, queuing or forwarding. Three main parts of a flow entry are shown in Fig. 3.

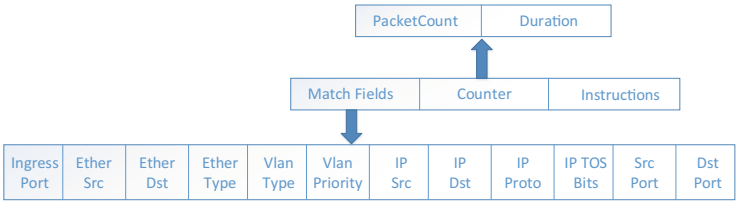

Fig. 3. Flow entry

From Fig. 3 we can see two important segments: "PacketCount" and "Duration". Once we get these two statistics from the OF switch, we calculate the hit rate of the flow table. Furthermore, we can calculate the gradient of hit rate with successive hit rates.

2.2 BPNN

Back propagation neural network(BPNN) [10] is one of the most useful artificial neural networks. It is a supervised neural network. A BPNN commonly consists of three layers: input layer, hidden layer, and output layer. The input layer and the output layer are single-layer structure, but the hidden layer can be a multi-layer structure. A basic BPNN is shown in Fig. 4.

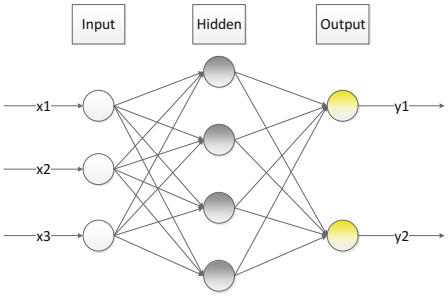

Fig. 4. Back propagation neural network

We choose BPNN to recognize the patterns of the attack since it makes our method adaptive to the hybrid attacks. Though attacker can confuse a detection system by combining different types of DDoS attacks, the behavior characteristics of the attack are still unchanged. BPNN needs a long time to train while little time to calculate, which helps us save a significant time and resource. The structure of BPNN may seem to be too simple, but it is already sufficient for an effective detection.

3 Our Scheme

In this section, we are going to present the details of our scheme. The scheme consists of 5 modules, including:

- Statistics collection module. Statistics collection module collects the statistics for port P in period T_1. The statistics that collected include the duration of each flow entry, the packet count of each flow entry and the number of flow entry for port P, represented by *Duration*, *FlowCount*, *Num* respectively.
- Feature extraction module. Feature extraction module calculates the average hit rate e of flow table in period T_1, then calculates the gradient of average hit rate in T_1, finally arranges n the gradients orderly in period $T_2 = n \cdot T_1 (n = 2, 3, \ldots)$.
- Attack detection module. Attack detection module mainly consists of a well-trained BPNN. The BPNN was trained with massive historical samples off-line. With a well-trained BPNN, attack detection module can recognize the DDoS attack patterns in real time.

– Attack defense module. Attack defense module defends the attack in real time. It pushes a flow entry to the corresponding OF switch, then the switch drops all packets arriving at victim port P. The result is that malicious packets from the attacker are entirely intercepted.
– Port recovery module. Port recovery module helps to recover the victim port automatically. We implement this module with the help of the flow entry pushed by attack defense module.

The whole working process of our scheme is shown in Fig. 5. And in the following sections, we will explain how every module works in details.

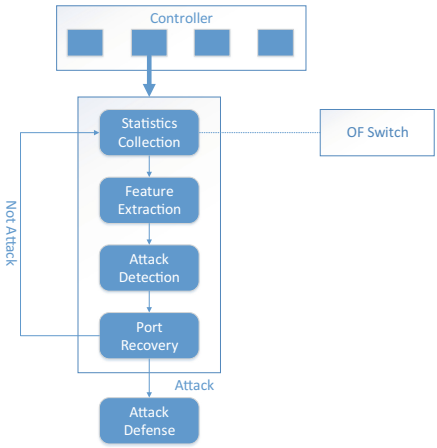

Fig. 5. Working process

3.1 Statistics Collection Module

Statistics collection module is embedded in the controller. It asks the controller for the statistics of port P in period T_1. The statistics that collected at time t include the duration of each flow entry, the packet count of each flow entry and the number of flow entry for port P. We order these statistics as a vector (P, *Duration, FlowCount, Num*).

3.2 Feature Extraction Module

Feature extraction module is also embedded in the controller. We get the time feature by calculating flow table's hit rate gradients in this module. First, we calculate the average hit rate of flow table for port P at time t. The equation we used is shown as Eq. (1) below.

$$e_t = \frac{\sum_{f \in F_P} FlowCount}{Num \cdot \sum_{f \in F_P} Duration} \tag{1}$$

The f is an existing flow entry in the flow table of the OF switch, and F_P is the set of flow entries in the flow table for port P.

We use Eq. (1) to calculate the average hit rates at time t and $t + T_1$, then calculate the gradient of average hit rates Δe between t and $t + T_1$ as Eq. (2) below.

$$\Delta e = \frac{e_{t+T_1}/Num_{t+T_1}}{e_t/Num_t} \tag{2}$$

Equation (2) can be transformed as Eq. (3) below.

$$\frac{e_{t+T_1}}{e_t} = \Delta e \cdot \frac{Num_{t+T_1}}{Num_t} \tag{3}$$

As we mentioned before, a DDoS attack against controller triggers enormous Packet_IN messages and the malicious packets are formed with a spoofed header. Therefore, the amount of flow entries increase rapidly while the average hit rate decrease rapidly. We can see from Eq. (3), the hit rate gradient Δe represents the relationship between the average hit rate and the number of flow entries. That is why we choose Δe as an important parameter to detect the DDoS attack.

Finally, we arrange these gradients orderly in period $T_2 = n \cdot T_1 (n = 2, 3, \ldots)$ and get a vector of gradients $\boldsymbol{e} = (\Delta e_1, \Delta e_2, \Delta e_3, \ldots, \Delta e_n)$. The vector $\boldsymbol{e}$ is the time feature and the input of the BPNN.

3.3 Attack Detection Module

After we get the inputs from feature extraction module, the next step is to train the BPNN. Since the BPNN is supervised, we have to label the inputs before training. After labeling, we get the training samples ($\boldsymbol{e}$, *Abnormal*, *Normal*). The first dimension is a gradients vector, the second and the third dimension are traffic patterns representing the DDoS attack and the normal traffic respectively.

In the detection phase, we use the well-trained BPNN to detect an attack in real time. First, we activate the statistic collection module and the feature extraction module to get a new $\boldsymbol{e}$, then put it into the BPNN without labeling, finally the BPNN judges which class this $\boldsymbol{e}$ belongs to. And we take the corresponding action according to the result.

We should point out that BPNN can be continuous updated online, even it has already been employed. The matter is the design of a correction strategy. Though a single mislabeled sample make little influence on BPNN, multiple mislabeled samples may lead to a totally wrong result due to the snowball effect. A correction strategy is necessary. But it involves the optimizing of a machine learning algorithm and is pretty complex. So we choose a static BPNN instead until we can design an effective correction strategy.

3.4 Attack Defense Module

Once a DDoS attack has been detected, this module is going to be activated. Attack defense module will push a special flow entry to the OF switch. The flow entry is shown in Table 1.

Table 1. Defense flow entry

Priority	Match field	Counter	Instructions
Minimal	In_Port = P	$\cdots$	Action = *Drop*

The function of this flow entry is to directly drop all packets arriving at port P. It cuts off the way from attacker to the controller so that the controller can be well protected. First, the new coming packets trigger no Packet_IN message but update the "Counter" of the flow entry. Meanwhile, the existing flow entries for legitimate services can still match their packets, since they have a higher priority. As a result, existing services will not be interdicted, but new services need a longer time to establish a link.

3.5 Port Recovery Module

Since the attack defense module has shut down port P for new flows, the new legitimate services can be shut down, too. In order to decrease the negative influence due to the defense method, we design a port recovery module to recover the victim ports dynamically. The main idea of this method is contrary to the detection method. In the detection method, we regard a low hit rate as a symbol of the attack. As for recovery method, we can expect that there will still exist massive packets arriving at the port P if the attack still goes on. Thus, the defense flow entry must show an extremely high hit rate since each unknown packet hits this flow entry. By calculating the hit rate of defense flow entry and comparing it with other flow entries representing legitimate flows, we can judge whether port P is still under attack or not. The complete recovery process is shown as below.

1. We traverse the flow entries of all the legitimate flow entries and calculate the average hit rate of them as Eq. (4). If there is no or very few legitimate flow entries for P, we turn to other ports. F_{normal} is the set of flow entries that belongs to all normal ports.

$$\bar{e} = \frac{\sum_{f \in F_{normal}} FlowCount}{\sum_{f \in F_{normal}} Duration} \tag{4}$$

2. Then, we calculate the hit rate of the flow entry as Eq. (5).

$$e = \frac{FlowCount}{Duration} \tag{5}$$

3. Finally, we compare these two hit rates. If the result satisfies Eq. (6), we judge that the victim port is out of the attack and remove the flow entry.

$$\frac{e}{\bar{e}} < \lambda \tag{6}$$

If we make assumptions that (1) attacker sends packets in a speed p_1 (2) attack lasted for time t_1 (3) legitimate users send packets in an average speed p_2 (4) it takes time t_2 to recover the victim port since the attack ended. Consequently, we can somehow transfer Eq. (6) into Eq. (7).

$$\frac{p_1 \cdot t_1 + p_2 \cdot t_2}{p_2 \cdot t_2} < \lambda \tag{7}$$

Furthermore, Eq. (7) can be transformed into Eq. (8).

$$\frac{p_1 \cdot t_1}{p_2 \cdot (\lambda - 1)} < t_2 \tag{8}$$

From Eq. (8), we can see that the time needed to recover the victim port depends on the attack scale and the λ we set. The larger attack scale is and the smaller value of λ we set, the longer time it needs to recover the victim. Normally, a port that had suffered from a large-scale attack is more likely to be attacked again, thus it will be better to set a longer recovery time for it. However, a long recovery time makes legitimate services unavailable sometimes. The existence of λ makes the time controllable. We can set a suitable value of λ to achieve the effective defense and reduce the recovery time, which will be shown in the experiment. In general, if the network is time-sensitive, we set a larger λ; if the network is security-sensitive, we set a smaller λ. In our concern, the set of a suitable λ requires an experienced administrator, and it is better if we can enable adaptive λ.

4 Experiment and Evaluation

In the experiment, we trained BPNN with the DARPA 1999 Intrusion Detection Data Set [12]. In this way, we used the DARPA 1999 Intrusion Detection Data Set as the source of network flow, then programmed to simulated the controller and switch instead of the real one. We used a computer with 2.6 GHz Intel Core i7-6700HQ CPU and 8G RAM to perform the experiment, also, the OS of the computer is Windows 10×64.

4.1 Experiment Parameter

The parameters of experiment include: the period of statistics collection T_1, the period of feature extraction T_2, the parameters of BPNN, a number threshold of flow entries and the threshold 4λ. We set these parameters as Table 2 shows.

We set the T_1 as 100 packets time, which means statistics collection module collects statistics when 100 packets arrive at port P every time. The reasons

Table 2. Experiment parameters

Parameter	Value
T_1	100 packets
T_2	$5 \cdot T_1$
Input layer nodes	5
Hidden layer nodes	40
Output layer nodes	2
Learning rate	0.25
Positive (attack) samples	500
Negative (normal) samples	1500
Number threshold	30
λ	9

are to decrease the time of detection and lessen the burden of the controller. Actually, we also collect the statistics when the time changes. This operation causes that the time of detecting an attack is not always an integer. The T_2 decides dimension of the input vector. With a smaller dimension, we can make a judgment earlier, but it increases the FP value. Through several experiments, we find it suitable to set the T_2 as 5 times of the T_1. The number threshold is a check parameter to decrease FP value. Only when the BPNN gives a positive result and the number of flow entries exceeds this threshold will we judge that port P is under attack. Finally, we performed several times experiments to determine the value of λ.

4.2 Accuracy and Recall

In the detection of the attack, the accuracy and recall are two of the most important measurements. The DARPA 1999 Intrusion Detection Data Set includes the training data and testing data and contains 56 types of attack, 201 attack instances. After training BPNN with this data set, we begin to test it. Figure 6 shows the attack instances actually occurred on April 5th, 1999.

As we mentioned before, we simulated the controller and switch to test our scheme with the data. But, it did not decrease the accuracy and recall, since the statistics supposed to be got from controller and switch can be extracted directly from the data set.

Theoretically, there are 4 types of attacks that trigger massive Packet_IN messages. The attack principles of 4 kinds of attacks are shown in Table 3.

Since ipsweep attack did not occur, we evaluate our method by detecting 3 attacks.

Figure 7 shows the attack instances detected by [6]. We can see that 3 attack instances have been detected. Figure 8 shows the attack instances we detected. Our method detected smurf and neptune, but we did not detect portsweep.

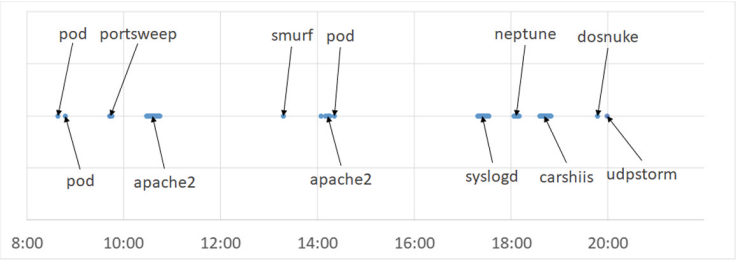

Fig. 6. Attack instances actually occurred

Table 3. Attack types

Attack	Principle
portsweep	Attacker sends a few packets, normally, 1 packet to every port of the target host to determine which port is available to attack
ipsweep	Attacker sends a few packets, normally, 1 packet to every host of the target network to determine which host is available to attack
smurf	Attacker sends massive ICMP packets with forged source IP address to the target host, then the target host replies to all nonexistent source hosts, and become too busy to handle other legitimate packets
neptune	Attacker sends massive SYN packets with different ports to target host, then the target host replies every SYN packet and waits, finally all the ports of target host are occupied

However, the portsweep attack detected by [6] triggers only 1 Packet_IN message per second. In fact, portsweep is not quite similar to the attack we try to detect. Though we did not detect portsweep, we detected another hidden attack and called it "burst". It represents a burst of TCP flow with random ports. Compared with portsweep, it is more likely to be a DDoS attack against controller, since it triggers massive Packet_IN messages. The "burst" is not defined as an attack in the DARPA 1999 Intrusion Detection Data Set, because the data set is used for the traditional network intrusions. Also, this flow is not considered as an attack instance in [6], either.

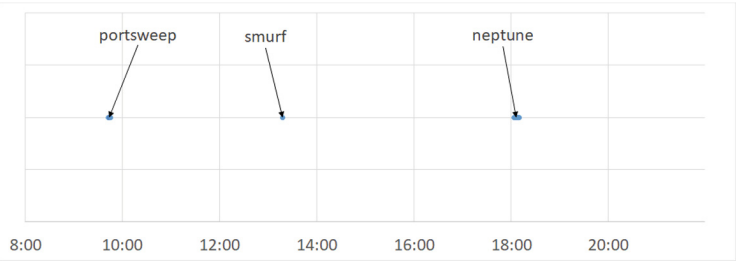

Fig. 7. Attack instances detected by method [6]

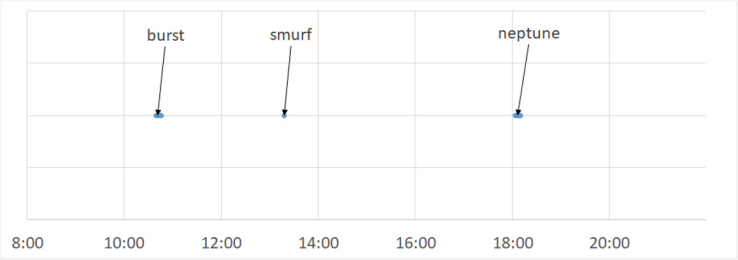

Fig. 8. Attack instances detected by our method

In fact, the value of the number threshold can be one of the factors that decide the detecting accuracy. The value of number threshold is the amount of the flow entries in the flow table. The accuracy is shown in Table 4.

Table 4. Threshold test

Number threshold	Actual attacks	Detected attacks	Misjudgment	Error rate
5	6	50	44	0.89
10	6	18	12	0.67
15	6	9	3	0.33
20	6	6	0	0

The 6 attack instances actually occurred are 4 “burst” instances, 1 smurf instance and 1 neptune instances. From Table 4 we can see, when number threshold increases, the number of misjudgment decreases. For this dataset, it is enough to set number threshold as 20. Since we set number threshold as 30 to ensure the accuracy, the attacks we detected in the experiment can be confirmed to be the real attack. However, the increase of number threshold may decrease the recall. For example, portsweep triggered only 1 Packet_IN message per second, therefore the amount of the flow entries did not exceed the number threshold and we did not detect it.

4.3 Promptness

Benefiting from using the gradients of hit rates rather than the hit rate itself, our method detected attack earlier, since we are detecting a tendency of the attack. The time it takes to detect 6 attack instances are shown in Table 5.

Table 5. Time for detection

Attack	Time (every 100 packet time)
burst 1	4.41
burst 2	4.45
burst 3	4.48
burst 4	4.45
smurf	4.76
neptune	4.75

As we mentioned before, we also collect statistics when time changes, such as 4.41 shown in Table 5. The average time our method takes to detect an attack is 4.55 times of period T_1, in another word, we detect a DDoS attack through the first 455 packets. Method [6] shows that they detect a DDoS attack by observing 6 successive "low-traffic flows", which means that it takes at least 6 periods of observation to detect an attack. Obviously, our method can detect the attack with fewer observations. It can be expected that an advanced machine learning algorithm shall help our scheme show a better performance.

4.4 Versatility

The time feature is extracted from the statistics in OF switch. It does not involve the content characteristics of flows, such as the IP address and protocol, but involves the behavior characteristics. The time feature can be used to detect all attacks aiming at disabling controller with massive Packet_IN message since their attack principles are the same. Consequently, our method can detect hybrid attacks in different scales or protocols.

4.5 Recovery

We have detected 3 kinds of attack and 6 attack instances, then activate the attack defense module 6 times. All the recovery time are shown in Table 6. The time is represented as seconds it passed from 08:00:00, for example, 08:00:05 is represented as "5 s".

From Table 6 we can see that the average time needed to recover a victim port is 31 seconds. Moreover, if we set a larger λ, we may even defend 4 "burst" attack instances with a single flow entry, then the controller only suffered from 441 packets. In our opinion, a time-sensitive network needs a smaller λ while security-sensitive network needs a larger λ. Since it need an experienced administrator to set a suitable λ, we may have to find a way to determine the value of λ automatically.

Table 6. Time for recovery

Attack	Start	End	Recover	Time
burst 1	9609 s	9611 s	9657 s	46 s
burst 2	9729 s	9731 s	9737 s	6 s
burst 3	9849 s	9851 s	9866 s	15 s
burst 4	9969 s	9971 s	9998 s	27 s
smurf	19088 s	19090 s	19127 s	37 s
neptune	36241 s	36651 s	39760 s	55 s

5 Conclusion

In this paper, we proposed a detection and defense method based on the temporal features of the DDoS attack against the controller. We calculated the gradients of the average hit rate with the statistics collected from OF switch. Then we used the gradients as the inputs of a BPNN. A well-trained BPNN can successfully detect a DDoS attack against a controller, then we defend against the attack effectively. Furthermore, we designed a method to dynamically recover the victim port. Our method was evaluated using the DARPA 1999 intrusion detection data set and compared with [6]. The results show that our method can detect an attack with greater speed and accuracy.

Acknowledgments. The work was supported by the National Natural Science Foundation of China (No. 61572001, No. 61502008, No. 61702005), The Natural Science Foundation of Anhui Province (No. 1508085QF132, No. 1708085QF136), and the Excellent Talent Project of Anhui University. The authors are very grateful to the anonymous referees for their detailed comments and suggestions regarding this paper.

References

1. Abdelmoniem, A.M., Bensaou, B., Abu, A.J.: SICC: SDN-based incast congestion control for data centers. In: IEEE International Conference on Communications, ICC 2017, Paris, France, 21–25 May 2017, pp. 1–6 (2017). https://doi.org/10.1109/ICC.2017.7996826
2. Akhunzada, A., Ahmed, E., Gani, A., Khan, M.K., Razzak, M.I., Guizani, S.: Securing software defined networks: taxonomy, requirements, and open issues. IEEE Commun. Mag. **53**(4), 36–44 (2015). https://doi.org/10.1109/MCOM.2015.7081073
3. Bizanis, N., Kuipers, F.A.: SDN and virtualization solutions for the internet of things: a survey. IEEE Access **4**, 5591–5606 (2016). https://doi.org/10.1109/ACCESS.2016.2607786
4. Braga, R., de Souza Mota, E., Passito, A.: Lightweight DDOS flooding attack detection using NOX/OpenFlow. In: Proceedings of the 35th Annual IEEE Conference on Local Computer Networks, LCN 2010, 10–14 October 2010, Denver, Colorado, USA, pp. 408–415 (2010). https://doi.org/10.1109/LCN.2010.5735752

5. Dayal, N., Srivastava, S.: Analyzing behavior of DDOS attacks to identify DDOS detection features in SDN. In: 9th International Conference on Communication Systems and Networks, COMSNETS 2017, 4–8 January 2017, Bengaluru, India, pp. 274–281 (2017). https://doi.org/10.1109/COMSNETS.2017.7945387
6. Dong, P., Du, X., Zhang, H., Xu, T.: A detection method for a novel DDOS attack against SDN controllers by vast new low-traffic flows. In: 2016 IEEE International Conference on Communications, ICC 2016, 22–27 May 2016, Kuala Lumpur, Malaysia, pp. 1–6 (2016). https://doi.org/10.1109/ICC.2016.7510992
7. Huang, X., Du, X., Song, B.: An effective DDOS defense scheme for SDN. In: IEEE International Conference on Communications, ICC 2017, 21–25 May 2017, Paris, France, pp. 1–6 (2017). https://doi.org/10.1109/ICC.2017.7997187
8. Kotani, D., Okabe, Y.: A packet-in message filtering mechanism for protection of control plane in OpenFlow networks. In: Proceedings of the 10th ACM/IEEE Symposium on Architectures for Networking and Communications Systems, ANCS 2014, 20–21 October 2014, Los Angeles, CA, USA, pp. 29–40 (2014). https://doi.org/10.1145/2658260.2658276
9. Kreutz, D., Bessani, A.N., Feitosa, E., Cunha, H.: Towards secure and dependable authentication and authorization infrastructures. In: 20th IEEE Pacific Rim International Symposium on Dependable Computing, PRDC 2014, 18–21 November 2014, Singapore, pp. 43–52 (2014). https://doi.org/10.1109/PRDC.2014.14
10. LeCun, Y., Bengio, Y., Hinton, G.E.: Deep learning. Nature **521**(7553), 436–444 (2015). https://doi.org/10.1038/nature14539
11. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G.M., Peterson, L.L., Rexford, J., Shenker, S., Turner, J.S.: OpenFlow: enabling innovation in campus networks. Comput. Commun. Rev. **38**(2), 69–74 (2008). https://doi.org/10.1145/1355734.1355746
12. MITLincolnLaboratory: DARPA 1999 Intrusion Detection Data Set. https://www.ll.mit.edu/ideval/docs/attackDB.html
13. Mousavi, S.M., St-Hilaire, M.: Early detection of DDOS attacks against SDN controllers. In: International Conference on Computing, Networking and Communications, ICNC 2015, 16–19 February 2015, Garden Grove, CA, USA, pp. 77–81 (2015). https://doi.org/10.1109/ICCNC.2015.7069319
14. Kokila, R.T., Selvi, S.T., Govindarajan, K.: DDOS detection and analysis in SDN-based environment using support vector machine classifier. In: 2014 6th International Conference on Advanced Computing (ICoAC), pp. 205–210, December 2014
15. Shin, S., Gu, G.: Attacking software-defined networks: a first feasibility study. In: Proceedings of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN 2013, Friday, 16 August 2013, pp. 165–166. The Chinese University of Hong Kong, Hong Kong (2013). https://doi.org/10.1145/2491185.2491220
16. Wang, R., Jia, Z., Ju, L.: An entropy-based distributed DDOS detection mechanism in software-defined networking. In: 2015 IEEE TrustCom/BigDataSE/ISPA, 20–22 August 2015, Helsinki, Finland, vol. 1, pp. 310–317 (2015). https://doi.org/10.1109/Trustcom.2015.389
17. Xu, Y., Liu, Y.: DDOS attack detection under SDN context. In: 35th Annual IEEE International Conference on Computer Communications, INFOCOM 2016, 10–14 April 2016, San Francisco, CA, USA, pp. 1–9 (2016). https://doi.org/10.1109/INFOCOM.2016.7524500

18. Yan, Q., Gong, Q., Deng, F.: Detection of DDOS attacks against wireless SDN controllers based on the fuzzy synthetic evaluation decision-making model. Ad Hoc Sens. Wirel. Netw. **33**(1–4), 275–299 (2016). http://www.oldcitypublishing.com/journals/ahswn-home/ahswn-issue-contents/ahswn-volume-33-number-1-4-2016/ahswn-33-1-4-p-275-299/
19. Zargar, S.T., Joshi, J., Tipper, D.: A survey of defense mechanisms against distributed denial of service (DDOS) flooding attacks. IEEE Commun. Surv. Tutor. **15**(4), 2046–2069 (2013). https://doi.org/10.1109/SURV.2013.031413.00127
20. Zhang, Y.: An adaptive flow counting method for anomaly detection in SDN. In: Conference on Emerging Networking Experiments and Technologies, CoNEXT 2013, 9–12 December 2013, Santa Barbara, CA, USA, pp. 25–30 (2013). https://doi.org/10.1145/2535372.2535411