Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

CIPA: A collaborative intrusion prevention architecture for programmable network and SDN

Xiao-Fan Chen [a,b,*], *Shun-Zheng Yu* [a,b]

[a] *School of Information Science and Technology, Sun Yat-Sen University, Guangzhou 510006, China*
[b] *SYSU-CMU Shunde International Joint Research Institute, Shunde, China*

ARTICLE INFO

Article history:
Received 5 December 2014
Received in revised form 19 July 2015
Accepted 26 November 2015
Available online 17 December 2015

Keywords:
Collaborative intrusion prevention
Collaborative intrusion detection system (CIDS)
Large-scale distributed attacks
Programmable network
Software-defined networks (SDN)

ABSTRACT

Coordinated intrusion, like DDoS, Worm outbreak and Botnet, is a major threat to network security nowadays and will continue to be a threat in the future. To ensure the Internet security, effective detection and mitigation for such attacks are indispensable. In this paper, we propose a novel collaborative intrusion prevention architecture, i.e. CIPA, aiming at confronting such coordinated intrusion behavior. CIPA is deployed as a virtual network of an artificial neural net over the substrate of networks. Taking advantage of the parallel and simple mathematical manipulation of neurons in a neural net, CIPA can disperse its lightweight computation power to the programmable switches of the substrate. Each programmable switch virtualizes one to several neurons. The whole neural net functions like an integrated IDS/IPS. This allows CIPA to detect distributed attacks on a global view. Meanwhile, it does not require high communication and computation overhead. It is scalable and robust. To validate CIPA, we have realized a prototype on Software-Defined Networks. We also conducted simulations and experiments. The results demonstrate that CIPA is effective.

1. Introduction

Nowadays, the Internet has been an inherent and important part of our daily life. The popularity of Internet results in a huge volume of different types of sensitive information, like financial transaction and individual privacy, being accessed via networks. This valuable information has attracted lots of hackers, which makes the Internet a target for a wide variety of on-line attacks. In the last few years, network attacks have become more complicated. Attackers have the ability to control a large amount of hosts to launch large-scale distributed attacks, such as flooding DDoS, stealthy scans and botnet. Symantec observed an underground economy advertisement in 2010 promoting 10,000 bots for $15. These bots can be used for spam, rogueware campaigns, and DDoS (Fossi et al., 2011). According to Symantec's 2014 annual threat report (Symantec, 2015), the monthly botnet-related ransomware activities increased by 500% from 100,000 in January 2013 to 600,000 in December 2013. The Sony PlayStation Network and Sony Entertainment Network were taken down by DDoS at August 2014, which cause great damage to the Sony Corporation (Hamilton, 2015). Radware (2015) shows that attack duration had increased and extra-large attacks had become common in 2014. Those large-scale coordinated intrusion behaviors threaten the security of the Internet.

The traditional single point IDS only monitors the network flow passing through it. Its view on the network state through the single point is too narrow to discover a coordinated intrusion that is distributed over the network. Moreover, itself may become target and suffer from the single-point-failure problem. Collaborative Intrusion Detection System, CIDS for short, has

* Corresponding author. Tel.: +86 15018410950.
E-mail addresses: chxfanz@gmail.com (X.-F. Chen), syu@mail.sysu.edu.cn (S.-Z. Yu).
http://dx.doi.org/10.1016/j.cose.2015.11.008
0167-4048/

a wider or global view on the network state. It thus has the capability to discover large-scale distributed attacks. CIDS usually consists of some detection units and one or more correlation unit(s). A detection unit detects the local traffic and sends alerts to the correlation unit(s) when it discovers abnormity. A correlation unit analyzes the alerts from detection units and other correlation units. The detection unit does the first time detection just with its local information. The correlation unit does the second time determination with more or global information.

Most of the existing CIDSs require deploying complete IDSs in multiple nodes of the network. The huge cost of such deployment may prevent them from a wide distribution of detection units over a large scale network. In this case, the correlation unit(s) may not be able to obtain the widest view on the network states. Their correctness may be affected by this limit. Besides, these CIDSs usually separate the detection phase and the determination phase. The accuracy and the view in the detection phase can affect the correctness of the determination phase, as reviewed in Section 2. To overcome those problems, we propose CIPA, a new collaborative intrusion prevention architecture. It is an ANN (Artificial Neural Network) based overlap network. It distributes its functions into programmable switches/routers to achieve a global view of the networks. Benefit from the programmable networks (McKeown et al., 2008; ONF, 2013; Suresh and Merz, 2011), we can readily deploy, modify and repeal CIPA over the network. It monitors network features and forms a global view on the network status. Based on the global view, CIPA can perform intrusion detection and mitigation to those discovered baleful on-line traffic with a high accuracy. The function in a switch is actually a lightweight program. It requires very low computation and storage capacities. The data transferred among the switches of CIPA are the calculated results, which brings little communication overhead. The features of requiring very low computation amount and communication overhead make CIPA deployable in a large scale network. To validate the power of CIPA, we test it with several sets of real-world data, including normal traffic, flooding DDoS, and worm attack. Fig. 1 is the diagrammatic sketch of the implementation of CIPA.

SDN (ONF, 2013) decouples the control plane and data plane of networks, which makes the network virtualization and programmability more powerful. It has been considered as one of key technologies to accelerate the evolution of Internet. SDN environment is more fit for deploying CIPA. Therefore, we realize a simple prototype of CIPA based on the openflow protocol of SDN and test it in a small real-world environment.

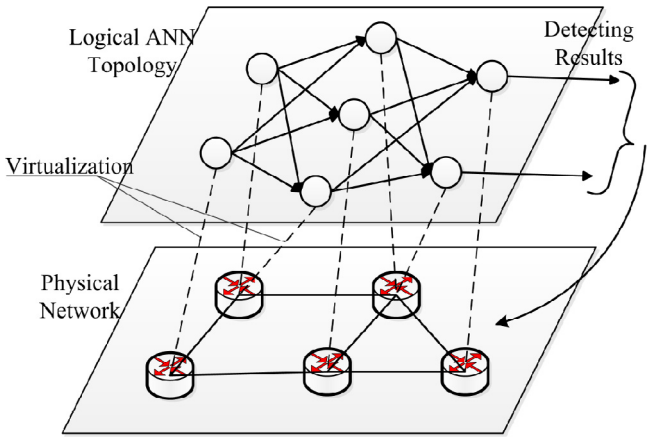

Fig. 1 – The implementation of CIPA.

The remainder of this paper proceeds as follows. Section 2 discusses the related works. Section 3 describes the distributed coordinated intrusions and their features, then discusses the architecture, mechanism and algorithm of CIPA. Section 4 shows several evaluation sceneries applied to CIPA and the corresponding results. Finally Section 5 concludes this paper.

2. Related work

2.1. *Coordinated intrusion detection/prevention system*

CIDS usually consists of the detection units and the correlation unit(s). A detection unit is responsible for detecting the traffic locally and sending alerts to the correlation unit(s). A correlation unit is in charge of analyzing alerts from different detection units and/or other correlation units. The detection units make the first detection with only a local view. The correlation units make the second determination with a more wider or global view. Therefore, CIDS is more possible to discover large-scale coordinated intrusion. It requires several devices, like servers, hosts and sensors, to work together. According to the structure that devices are organized, distributed IDS can be divided into three categories (Zhou et al., 2010).

The first one is the centralized structure. NSTAT (Kemmerer, 1997) is a centralized structure IDS aiming at detecting network wide intrusion. The client nodes, i.e. the monitors or probes, read and filter the audit trails on local hosts and then send them to a central server, i.e. the central IDS. The central server merges the data from multiple sources and do the analysis job. In CRIM (Cuppens and Miège, 2002), a central analyzer is utilized to correlate alerts from individual IDSs. However, its correlation system is semi-automatic since it relies on human interactions with the system for defining attack descriptions (Fung, 2011). Huang et al. (1999) and Snapp et al. (1991) proposed the similar prototypes of centralized structure. There are a powerful server playing the role of correlation unit, and a number of less powerful devices distributed in different places of the network to act as local IDSs, i.e. detection units. To reduce the communication overhead, the detection units send alerts to the central server only when they find abnormity. Such a centralized CIDS still suffers from two primary shortcomings: (a) the first time detection is based on just local information and it may push down the accuracy of the correlation unit; (b) the central server is the single point of failure.

The second one is the hierarchical structure. In Zhang et al. (2001), intrusion detection agents form a hierarchical organization. The agents of each tier are of the similar functions. Each agent monitors and detects the traffic locally, correlates the alerts from lower tier agents, and sends results to higher tier agents. AAFID (Balasubramaniyan et al., 1998) assigns a transceiver to monitor the state of a host. The transceiver can appoint agents to monitor the event of the host, collect data from the agents and process them. Monitors in AAFID are used to monitor the state of several hosts. They are hierarchically

organized. They will correlate the reduced information from transceivers and other monitors. In hierarchical CIDS, to tackle the single-point-failure problem, correlation units are hierarchically organized. The lower level correlation units collect alerts from detection units, analyze them and send alerts to the higher level one for further analysis. The information transformed in such CIDS is the abstraction or processed results. So the higher the level of the IDS is, the wider but more blurred view it will have. Obviously, the high level IDSs may still suffer from the problems similar to the centralized CIDS.

The last one is the fully distributed structure. Vlachos et al. (2004) is an early attempt in P2P-based CIDS. It can detect the virus or worm epidemic outbreak. Each peer in the CIDS consists of an IDS and an application called NetBiotic. The NetBiotic is used to exchange messages about the detected attacks with other peers. Tian et al. (2007) proposes a P2P-based intrusion detection scheme. It deploys a complete IDS on each of the selected peers of the overlay network to find DoS attacks and find new intruders with low false alert rate. Zhou et al. (2009) deploys its CIDS on a P2P overlay network too. In each IDS peer, when feature value is greater than its local support threshold, which is estimated by probabilistic approach, an alert will be triggered. The alert will be sent to a corresponding IDS peer for correlation. But each IDS peer has to maintain a lattice for each source IP address, which may cost too much storage space at a large scale network. Dumitrescu (2006) proposes an ANN-based distributed system to realize intrusion detection and network resource control. Every peer in Dumitrescu (2006) has a complete ANN to do the traffic analysis and they will exchange limited information with each other. Yegneswaran et al. (2004) proposes DOMINO to monitor Internet outbreaks for a large scale network. Gamer (2012), Janakiraman et al. (2003) and François et al. (2012) similarly propose fully distributed CIDS. In a fully distributed CIDS, each IDS peer has both a detection unit and a correlation unit. It disperses the correlation job of a central server to peers to overcome the single-point-failure problems. To reduce the communication overhead, detection units only send alerts to the adjacent correlation units. However, it may be difficult to guarantee the same detection accuracy as a centralized CIDS due to the view of the correlation units is not so wide.

To overcome the single-point-failure problem of the centralized/hierarchical CIDS, and the accuracy problem of the fully distributed CIDS, we propose a novel CIDS, i.e., CIPA. It consists of several ANNs that use switches over the entire network as their neurons. The input neurons of the ANNs are responsible for collecting data from all over the network, the hidden ones for processing the data in parallel, and the output ones for reporting the detection results.[1] Therefore, CIPA always has a global view on the network traffic and has no single-point-failure problem due to its fully distributing nature. Moreover, the detection based on a global view brings two more benefits: (i) some special coordinated attacks may cause only tiny changes to some network features. These evidences may be neglected by detection units with just local views, while they are apparent from the global view; (ii) the features of normal traffic in some parts of the network may vary radically because of fluctuation. They might be considered as abnormal by the CIDSs without central servers. In contrast, CIPA can reduce such false positive because the fluctuations of normal traffic in different parts of the network are usually asynchronous while those caused by some coordinated attacks are generally synchronous. It will be discussed in detail in Section 4.1. Besides, the existing CIDSs have to deploy a fully-functioning IDS on the switches and utilize a large amount of their computation resources. In contrast, CIPA disperses the computational capability to part or all of the switches. Each of them just requires little resources for simple computation like a neuron of the ANN.

[1] See Section 3.4 for the details of the procedure and algorithm.

2.2. *Intrusion detection/prevention in SDN*

During the last few years, there are some research works on the network security of SDN. In Braga et al. (2010), three OF (OpenFlow) switches are used to collect flow-based features and send them to the NOX controller (Tootoonchian, 2015). A SOM (Self-Organizing Maps) based IDS built in NOX is responsible for the detection job. But NOX does not merge the information from the OF switches. Shin et al. (2013) makes one OF switch acts as a proxy, which keeps the information of each TCP session/connection. It tries to block attacks related to TCP handshaking, like SYN flood, at the switch side. Different from these single point IDSs, Shirali-Shahreza and Ganjali (2013a, 2013b) introduce a flexible sampling framework for OpenFlow-based SDN. The sampling framework provides NOX controller the access to packet level information with low overhead. It is used in port scan detection (Shirali-Shahreza and Ganjali, 2013a). Defense-Flow (Radware, 2013), deployed on controller, takes the place as IDS to specially go against DoS/DDoS. It redirects the suspicious flow to the DefensePro mitigation devices. Then the clean traffic is forwarded to the targets by DefensePro. Mehdi et al. (2011) tackles the anomaly detection problem in SDN-based home/office networks. Xing et al. (2013) and Chung et al. (2013) focus on SDN-based cloud security.

To the best of our knowledge, our work is the first one to integrate a decentralized collaborative intrusion detection/prevention system into the openflow-based SDN. Because it is to be deployed in major part or all of the switches, instead of controller(s), it has no single-point-failure problem and is scalable in SDN.

3. CIPA

This section introduces the background knowledge of BP (Back Propagation) neural network (Haykin, 2007; Rumelhart et al., 2002), the features of coordinated attacks, the architecture of CIPA, different components of CIPA and their working principle.

3.1. *Back propagation neural network*

Neural Network is a branch of the Artificial Intelligence technologies. It imitates the working process of human brain. An ANN consists of neurons and links. Neuron is the basic function unit of the ANN. Neurons with the similar functions are placed in the same layer. They receive signals from neurons of the previous layer, do some simple mathematical

operation on the signals, and send the processed signals to neurons of the next layer. Links are the connections between neurons. In some sort of ANNs, such as BP net, a weight is assigned to a link, which represents the strength of the connection between the neurons. BP is one of the most common neural network structures. Its theory has been studied thorough and it has been widely used in many fields, including intrusion detection (Cannady, 1998). Therefore, we utilize BP neural net as the ANN of CIPA in this paper. A BP neural network usually consists of an input layer, one or more hidden layer(s), and an output layer. Fig. 2 shows a 3-layer BP neural network.

In a BP neural network, the neurons of the input layer, or input neurons, are used to accept the input vector, do some preprocessing, such as normalization, and send the preprocessed results to the neurons of the first hidden layer. The hidden neurons accept the results from the input neurons, sum them up and do the mapping according to their activation functions. When dealing with classification problem, the activation function that a hidden neuron performs is usually the nonlinear sigmoid function:

$$y_j = \sum_i \omega_{ij} \cdot x_i \tag{1}$$

$$z_j = \frac{1 - exp(-2 \cdot y_j)}{1 + exp(-2 \cdot y_j)} \tag{2}$$

where z_j is the output of the jth hidden neuron, y_j the weighted sum up result of the jth hidden neuron, x_i the output of the ith input neuron, and w_{ij} the weight between the ith input neuron and the jth hidden neuron. Then the hidden neurons send the calculated results to the neurons of the output layer or the next hidden layer, if existed. The output neurons process the values in the same way as what hidden neurons do, while their outputs are the final results of the BP net.

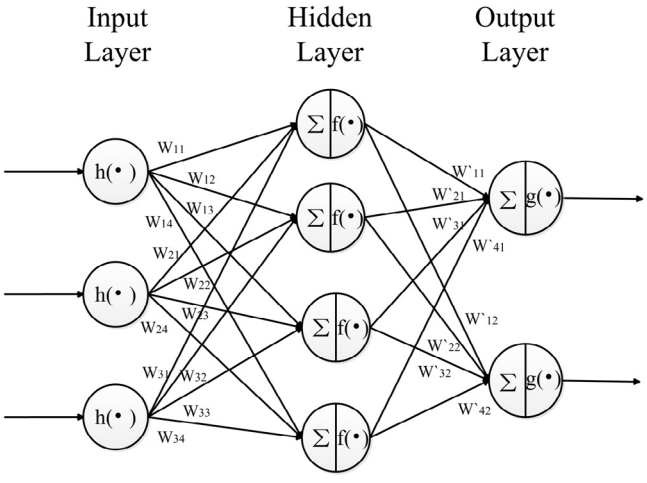

Fig. 2 – Example of a 3-layer BP neural network. w_{ij} is the weight between input neuron i and hidden neuron j. w'_{jk} is the weight between hidden neuron j and output neuron k. Σ is the weighted sum up function. $h(\bullet)$ is the preprocess function. $f(\bullet)$ and $g(\bullet)$ are the activation functions.

Table 1 – The network features can be monitored.

Features	Attack types
Packet rate	Flooding DDoS, flash crowds
Percentage of ICMP packet	icmp flooding, scan attack, worm
Percentage of TCP packet	tcp flooding, tcp worm
Percentage of UDP packet	udp flooding, udp worm
Percentage of large packet	Self-carried worm
Percentage of short packet	Scan attack
Percentage of SYN/ACK/RST	DDoS
Distribution of IP address	DDoS, scan
Distribution of port	Scan, worm
Interval of arriving packet	DDoS
Duration of each flow/session	SYN flooding
Packets/bytes per flow	Flooding DDoS, flash crowds
Percentage of single flow	DDoS
Growth of new flow	Scan attack
Special protocol of application layer	Worm, botnet

3.2. *Coordinated attacks and feature selection*

Coordinated attacks, like flooding attack and worm outbreak, are usually large-scale, distributed and synchronous. When such intrusions occur, the network behavior may be quite different from that of the normal situation. Some network features may change rapidly and simultaneously. By globally monitoring those features, we can capture the abnormal behavior over the network. Because different attacks may cause different abnormal behaviors and affect different features, a set of features should be carefully selected for detecting each type of attacks. More features should be selected for detecting multiple attacks. For example, a significant feature is the synchronous increase of packet rates appearing everywhere over the network. When large-scale stealthy scans occur, there will be a large number of ICMP packets and/or TCP packets with SYN/RST flag set in the network (Li et al., 2008). If UDP worm or UDP flooding occurs, the proportion of UDP packets may be much larger than usual (Li et al., 2008). The coordinated attacks will lead to the dramatic change of the distribution of IP address and port number (Li et al., 2008; Zhou et al., 2010). Based on the observations on known coordinate attacks, we choose some of the useful features, such as packet rate, the number of ICMP packets and the proportion of UDP packets, to monitor, as shown in Table 1. The table also shows what types of attacks can be detected by using the corresponding features.

3.3. *Architecture of CIPA*

CIPA is implemented as an ensemble of ANNs[2] overlaid on the network. For each ANN, the parallel and simple computational capabilities of neurons in ANN make it possible to disperse the functions of a traditional single point ANN over the entire network. Programmable switches of the network play the role of neurons in the neural network. Each switch of the network contributes a part of its computation and storage capacities to virtualize one or more neurons. Neurons virtualized in different switches connect to each other using logical links

[2] Ensemble of ANNs is discussed in detail in the last paragraph of Section 3.4.

and transfer the calculated results in these virtual connections. In this way, an ANN-based overlap network is constructed. Since BP neural nets are used as the ANNs of CIPA in this paper, there are three kinds of neurons in CIPA, i.e. neurons of input layer, hidden layer(s) and output layer in an ANN. A major part or all of the switches in the network are chosen to act as neurons of the ANNs. Nodes (i.e. neurons) on the same layer function in parallel. The input nodes collect in parallel the features reflecting the state of the network and send the preprocessed data to the hidden nodes. The hidden nodes are responsible for doing simple calculation and sending the outcomes to the nodes of the output layer or the next hidden layer in parallel. The output nodes produce the final results and, if necessary, issue alert messages. The final results can be as simple as just telling us whether the network is normal or not. The final results of the ensemble of ANNs can be sent to another ANN so that they can conclude more useful information, such as what kind of attack is taking place, or what stage it is now during a multi-step attack. When intrusion is detected, CIPA can send alert messages to some predetermined or all of the switches in the network. These switches can make up the intrusion mitigation components of CIPA. According to the alert messages and the scheme adopted by CIPA, these switches can locate the intrusion sources and build the filtering rules to mitigate the intrusion traffic. Fig. 3 illustrates the workflow of CIPA.

3.4. *Working principle of CIPA*

The input neurons are responsible for monitoring the network traffic passing through them, extracting features from the traffic, translating their format, and sending the preprocessed values to the connected nodes of the next hidden layer. We let the ensemble of ANNs have the same input neurons. Each input neuron monitors a set of features and gets their statistics in one or more given action cycles. In other words, different features may have different sampling intervals. For simplicity, we assume all the sampling intervals for different features are multiple action cycles. Therefore, in each action cycle, each input neuron can get a vector of samples. All of the input neurons can get multiple vectors in parallel. To simplify the operations of the ANNs and reduce the communication overhead of CIPA, the vector obtained by an input neuron in one action cycle is transferred as a whole to the next layer. For example, two features *num_pkt* and *num_unusedIP* are monitored, where *num_pkt* is the number of packets passing through the input neuron and *num_unusedIP* the number of unused IP addresses that the input neuron observed during its sampling interval. Assume that τ_a is an action cycle. The sampling intervals of *num_pkt* and *num_unusedIP* are respectively τ_a and $2\tau_a$. Then in one action cycle, the feature value vector will be {*num_pkt, null*}. In the next action cycle it will be {*num_pkt, num_unusedIP*}.

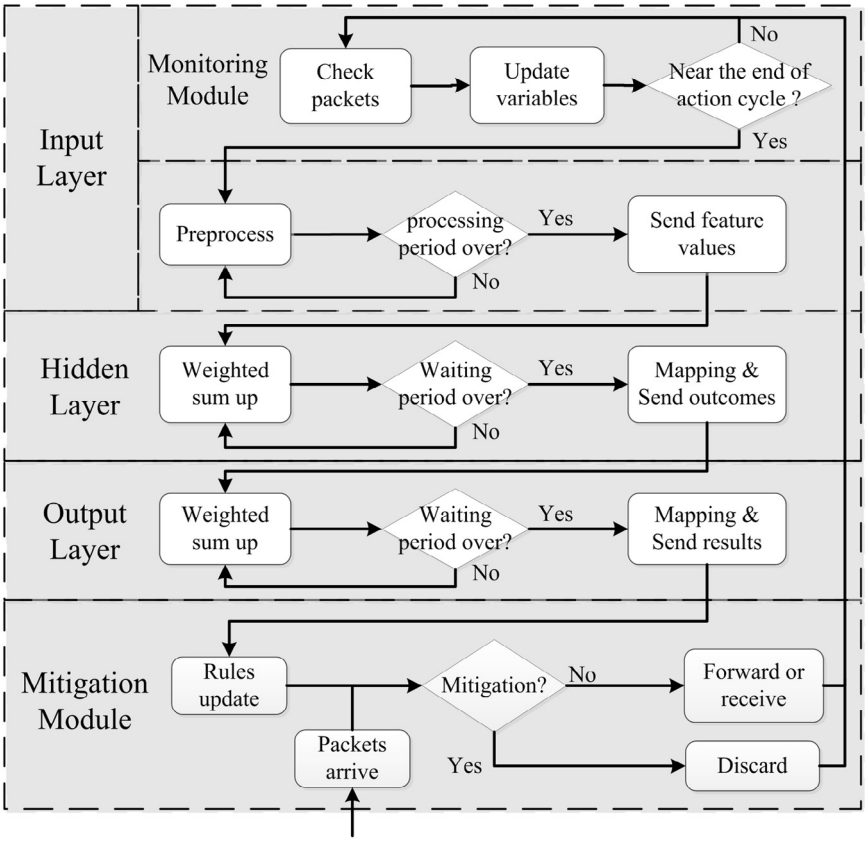

Fig. 3 – Workflow of CIPA. A packet arriving at an input neuron will be inspected for features extraction and variables update. The processed feature values or outcomes will be sent to the hidden layer and at last the output layer for further processing after some predefined intervals. Finally, the detection results are sent to update their mitigation rules.

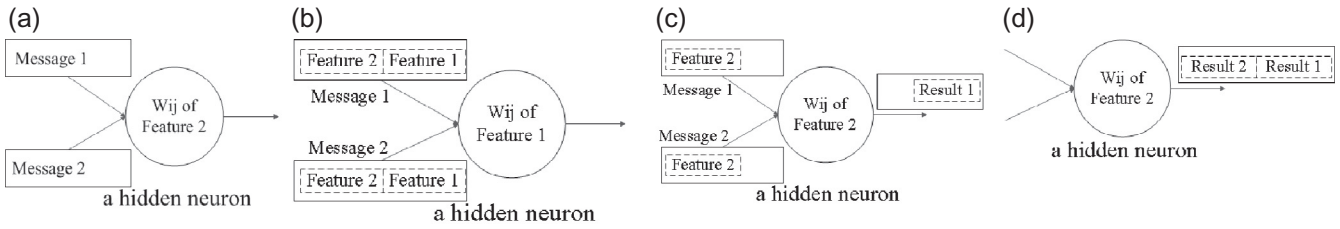

Fig. 4 – The procedure of time division multiplexing of a hidden neuron. (a) slot1: Receives messages. (b) slot2: Processes the values of feature 1. (c) slot3: Processes the values of feature 2. (d) slot4: Sends the results as a whole.

Different features have different ranges of observed values. Hence all samples obtained in the input layer have to be normalized before sent to the next hidden layer.

$$x = \frac{x_o - x_{min}}{x_{max} - x_{min}} \tag{3}$$

where x_o is the observed value, x_{min} the minimum, x_{max} the maximum, and x the normalized feature value.

The neurons of the first hidden layer accept the results from the input neurons. The ith hidden layer accept those from the (i – 1)th one. Since there may be more than one non-null features in a message, the neurons of the hidden layer process the features one by one. For each feature they do the simple mathematical manipulations and send all the outcomes of all the features together to the next hidden layer if exists, otherwise to the output layer. The simple mathematical manipulations refer to Eqs. (1) and (2). We note that w_{ij}s are different for various features. In this way, CIPA virtualizes multiple ANNs on the same ANN structure by time division multiplexing. Fig. 4 shows the procedure of time division multiplexing of a hidden neuron. The neurons of the output layer receive the results from the last hidden layer, do the mapping operation similar to the hidden neurons, and then send out the final results as the detection results of CIPA. When BP net is used in classification problem, the activation function of the output neurons can be the same as that of the hidden neurons.

In considering transmission delay of messages between neurons, a synchronization mechanism is introduced to synchronize the actions of all neurons. The synchronization mechanism of CIPA can be illustrated by Fig. 5. It presents an example of the working cycles of a 3-layer neural net. τ_a is the duration of an action cycle of the input neurons. During each action cycle, the input neurons extract the header and/or payload information of every received packet or information of every flow passing through them. Based on the extracted information, the features' variables are updated within the current action cycle. Before the end of τ_a, values of some features, whose sampling intervals become due, are normalized and sent to the hidden neurons within a short time interval τ_b. Before the hidden neurons start their working period, all of them must have received the normalized feature values from the input neurons. To ensure this, the hidden nodes wait for a period of τ_c for receiving the feature values from input layer. Suppose that the largest round trip time between connected neurons is RTT_{max}, then τ_c should be a little bit larger than $RTT_{max}/2$ such that all the hidden neurons have enough time to obtain feature values from the input layer. When waiting period is over, the processing period begins and the hidden nodes will deal with the values according to Eqs. (1) and (2), where τ_d is the maximum processing time. After that, all hidden nodes send the outcomes to the output nodes. Similarly, the output nodes spend the next two periods of τ_c and τ_d for receiving and processing all the outcomes from the hidden layer. τ_a is usually assumed to be several or tens seconds. τ_b, τ_c and τ_d are tens milliseconds. These four intervals will be informed to all participated nodes and will not be changed after initialized.

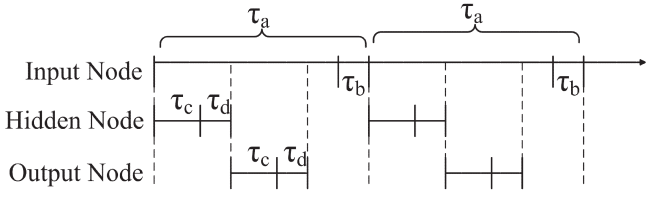

Fig. 5 – Example of the synchronization of a 3-layer neural net. τ_a is the action cycle of input neurons. τ_b is the processing period of input neurons. τ_c and τ_d are respectively the waiting period and processing period of both hidden and output neurons.

In this paper, we use two output neurons in CIPA. One produces the degree of abnormal z^a, and the other one produces the degree of normal z^n. We set the suspicious level from the viewpoint of the ith feature to be:

$$S_i = z_i^a - z_i^n \tag{4}$$

Output nodes will send z_i^a and z_i^n to all switches in CIPA. Then the switches calculate the final suspicious level of received packets/flows by the following formula.

$$S = \sum_i sign(h_i) \cdot S_i \tag{5}$$

$$sign(h_i) = \begin{cases} 1, & \text{if the ith feature appear in packet} \\ 0, & \text{else} \end{cases} \tag{6}$$

If S is larger than a preset threshold σ, then the corresponding packets should be discarded.

In considering the robustness of CIPA, at least one server is needed to allocate the resources of the programmable network. The server helps to construct CIPA according to the

user requirements and strategies, as well as the states of network devices. It is also responsible for the operation and maintenance of CIPA. In SDN, the controller can play this role. On the other hand, ANN is capable of dealing with incomplete and distorted data. Therefore, even if some inputs of neuron(s) are incomplete or distorted due to node crashing, timeout or noise, CIPA can still function as usual.

We note that ensemble of ANNs (Golovko et al., 2007; Hansen and Salamon, 1990; Zhang et al., 2005) is used to improve the generalization ability of the system. Ensemble means creating multiple neural networks and combining them to produce a desired output. Each of the ANNs is trained independently. When at work, they produce the outputs respectively. There is a model to make the final collective classification. Ensemble of ANNs has been proved to have better generalization ability than a single ANN (Golovko et al., 2007; Hansen and Salamon, 1990; Zhang et al., 2005). In the ensemble approach, one ANN is usually used for detecting one attack. The solutions in Zhang et al. (2005) and Golovko et al. (2007) are such examples. In the solution of Zhang et al. (2005), if there is a new attack that appears, a new ANN will be trained to detect the new attack and be added to the group of the second level misuse classifiers. In contrast, we use a new way of ensemble. In our solution, there are multiple ANNs stacking together to monitor a vector of network features, in which each ANN monitors one network feature. For example, six features are used in the simulations of Section 4.1, so there are six BP nets. The outcomes of all ANNs can be inputted into a second level ANN. This ANN acts as the final classifier. It can be trained to classify attacks. If the monitored network features are selected to be sufficient, the normal behavior and different attack behaviors over the network may appear in different areas of the feature space. This makes it possible to classify various attacks as well as their variants based on their similar features or similar impacts on the network. Some instances can be found in Section 4.5. When a new and quite different attack (or variant) appears, the second level ANN may count many times of unclassified events. The second level ANN can be retrained for classifying the new kind of attack, or a first level ANN can be added to monitor a new feature if it is required for detecting the new attack.

The pseudo-code fragment, i.e. Algorithm 1, illustrated the main detecting and mitigating procedure of CIPA, which is the sum up of Section 3.4.

3.5. *Training algorithm of BP net*

Generally, every training algorithm for BP neural network includes two phases. The first one is the forwarding phase during which the output of each layer is calculated successively. The second one is the backward phase during which the error is transited backward to fix the weights of all connections. After the forwarding phase, an error function, i.e. the objective function can be obtained. Different training algorithms have their own schemes to utilize the error function in the backward phase.

The BP net training algorithm used in CIPA is RPROP (Riedmiller and Braun, 1993). It is a resilient backpropagation approach. It converges faster than gradient-descent (Haykin, 2007) approach. In our scheme, the objective function in batch learning mode is:

$$E = \frac{1}{2}\sum_{p}\sum_{q}(y_{p,q} - d_{p,q})^2 \tag{7}$$

where p refers to the pth output, q refers to the qth training sample, $y_{p,q}$ is the real output, and $d_{p,q}$ is the excepted output.

In gradient-descent, the size of weight update is directly proportional to the learning rate η and the size of partial derivative $\frac{\partial E}{\partial \omega}$. Riedmiller and Braun (1993) believe that the benefit of carefully adapted learning rate can be upset by the unforeseeable behavior of the partial derivative. Hence, in resilient backpropagation, the derivative only determines the direction of weight update $\Delta\omega$. Riedmiller and Braun (1993) introduces an update-value Δ_{ij}. Its update follows the rule:

Algorithm 1 Main detecting and mitigating algorithm of CIPA

for each input neuron i **do**
  while action cycle is not over **do**
    inspects packets and update the feature variables
  end while
  for the k^{th} feature variable x_{ik_o} of the i^{th} input neuron **do**
    normalization: $x_{ik} = \frac{x_{ik\_o} - x_{ik\_min}}{x_{ik\_max} - x_{ik\_min}}$
  end for
  sends out $X_i = [x_{i1}, x_{i2}, ...]$ to all hidden neurons
end for
for each hidden neuron j **do**
  while waiting period is not over **do**
    receives the variables X_i from input neurons and update the variables
  end while
  for the k^{th} variable z_{jk} of the j^{th} hidden neuron **do**
    $y_{jk} = \sum_i \omega_{ij} \cdot x_{ik}$
    $z_{jk} = \frac{1 - exp(-2 \cdot y_{jk})}{1 + exp(-2 \cdot y_{jk})}$
  end for
  sends out $Z_j = [z_{j1}, z_{j2}, ...]$ to all neurons of the next hidden layer or output layer
end for
for each output neuron t **do**
  does the same things as what a hidden neuron does
  sends out $Z_t = [z_{t1}, z_{t2}, ...]$ to all the switches of CIPA
end for
for each switch of CIPA **do**
  if a packet arrives **then**
    for the k^{th} feature **do**
      $S_k = z_k^a - z_k^n$
    end for
    $S = \sum_k sign(h_k) \cdot S_k$, if the k^{th} feature appear in the packet, $sign(h_k) = 1$, otherwise $sign(h_k) = 0$.
    if $S > \sigma$ **then**
      discards the packet
    end if
  end if
end for

$$\Delta_{ij}(t)=\begin{cases} min(\eta^{+}\cdot\Delta_{ij},\Delta_{max}), & \text{if } \frac{\partial E(t)}{\partial \omega_{ij}(t)}\cdot\frac{\partial E(t-1)}{\partial \omega_{ij}(t-1)}>0 \\ max(\eta^{-}\cdot\Delta_{ij},\Delta_{min}), & \text{if } \frac{\partial E(t)}{\partial \omega_{ij}(t)}\cdot\frac{\partial E(t-1)}{\partial \omega_{ij}(t-1)}<0 \\ \Delta_{ij}(t-1), & \text{else} \end{cases} \tag{8}$$

where $0<\eta^{-}<1<\eta^{+}$

If $\frac{\partial E}{\partial \omega_{ij}}$ retains its sign, it can be increased by η^{+} to speed up the convergence. If the partial derivative $\frac{\partial E}{\partial \omega_{ij}}$ changes its sign, it means that the last update of ω_{ij} is too large and the algorithm has just jumped over a minimum of the error surface. So the update-value Δ_{ij} of the corresponding weight ω_{ij} has to be decreased with a factor η^{-}.

Next comes the renew of weight-update. If the partial derivative does not change its sign, the update of $\Delta\omega$ follows the rule:

$$\Delta\omega_{ij}(t)=\begin{cases} -\Delta_{ij}(t), & \text{if } \frac{\partial E(t)}{\partial \omega_{ij}(t)}>0 \\ +\Delta_{ij}(t), & \text{if } \frac{\partial E(t)}{\partial \omega_{ij}(t)}<0 \\ 0, & \text{else} \end{cases} \tag{9}$$

If the partial derivative change its sign, a method called backtracking (Riedmiller and Braun, 1993) would be used. The update of $\Delta\omega$ follows the rule:

$$\Delta\omega_{ij}(t)=-\Delta\omega_{ij}(t-1) \tag{10}$$

Then set $\frac{\partial E(t)}{\partial \omega_{ij}(t)}=0$. By this way, if the step of the algorithm is too big to miss a minimum in error surface, it will go back to the previous position.

At last, the weight can be renewed by the following rule:

$$\omega_{ij}(t+1)=\omega_{ij}(t)+\Delta\omega_{ij}(t) \tag{11}$$

$\omega(t+1)$ is the new weight after the $(t+1)$th iteration, $\omega(t)$ is the old weight before the $(t+1)$th iteration.

The pseudo-code fragment shows the kernel of the training algorithm of CIPA.

The training of BP net can be off-line, while the on-line update of the synaptic weights is needed to keep up with the secular change of network traffic and attack traffic. It is not necessary to use every sample that CIPA monitored at run time for on-line arguments update. A preset variable k is needed. Only one sample is used to update the model arguments in every k samples. The value of k depends on the network environment. For example, when attack is less or local normal traffic is more stable, k should be large enough to miss the informationless samples. The on-line learning can use the same algorithm as off-line training. The synchronization mechanism is similar to that described in Section 3.4, while the values are transformed in the opposite direction.

3.6. Implementation on openflow-based SDN

Openflow is the most common Southbound API protocol.[3] Both Open vSwitch (OVS, 2013) and POX controller (MurphyMc, 2015) are widely used in SDN field and they support openflow. Therefore, we build our SDN environment and develop CIPA based on Open vSwitch and POX. What Open vSwitches do are more or less the same as what programmable switches do mentioned in the previous sections. When receiving a packet, before parsing it and matching it in the flow table, Open vSwitch will send it to the input neuron for feature extraction and statistics update. Before the action cycle is over, input neurons will process the statistics and send them to the hidden neurons according to the neural forwarding table. When waiting period is over, hidden and output neurons will process the statistics. When the processing period is over, hidden neurons will send the outcomes to the neurons of next hidden layer or output layer. The output neurons will generate the final detection results of the whole ANN in parallel.

Algorithm 2 Update of weight ω

for all ω_{ij} **do**
  if $\frac{\partial E(t)}{\partial \omega_{ij}(t)}\cdot\frac{\partial E(t-1)}{\partial \omega_{ij}(t-1)}>0$ **then**
    $\Delta_{ij}(t)=min(\eta^{+}\cdot\Delta_{ij},\Delta_{max})$
    $\Delta\omega_{ij}(t+1)=-sign(\frac{\partial E(t)}{\partial \omega_{ij}(t)})\cdot\Delta_{ij}(t)$
    $\omega_{ij}(t+1)=\omega_{ij}(t)+\Delta\omega_{ij}(t)$
  else if $\frac{\partial E(t)}{\partial \omega_{ij}(t)}\cdot\frac{\partial E(t-1)}{\partial \omega_{ij}(t-1)}<0$ **then**
    $\Delta_{ij}(t)=max(\eta_{-}\cdot\Delta_{ij},\Delta_{min})$
    $\omega_{ij}(t+1)=\omega_{ij}(t)-\Delta\omega_{ij}(t-1)$
    $\frac{\partial E(t)}{\partial \omega_{ij}(t)}=0$
  else if $\frac{\partial E(t)}{\partial \omega_{ij}(t)}\cdot\frac{\partial E(t-1)}{\partial \omega_{ij}(t-1)}=0$ **then**
    $\Delta\omega_{ij}(t+1)=-sign(\frac{\partial E(t)}{\partial \omega_{ij}(t)})\cdot\Delta_{ij}(t)$
    $\omega_{ij}(t+1)=\omega_{ij}(t)+\Delta\omega_{ij}(t)$
  end if
end for

In SDN, an Open vSwitch has one IP address in the control plane and no IP address in the data plane. It means that Open vSwitches cannot talk to each other. However the messages transmitted between neurons require the communication between Open vSwitches in the data plane. Therefore, we build a neural forwarding table[4] in each Open vSwitch for it to the communication with other ones. In this table, each entry consists of the MAC address of destination port of another Open vSwitch and the output port ID of the Open vSwitch that this forwarding table belongs to. Fig. 6 is an example of the neural forwarding table. The controller is responsible for not only its

[3] Openflow is the communication protocol between controller and switch.

[4] It is similar to layer 2 forwarding table.

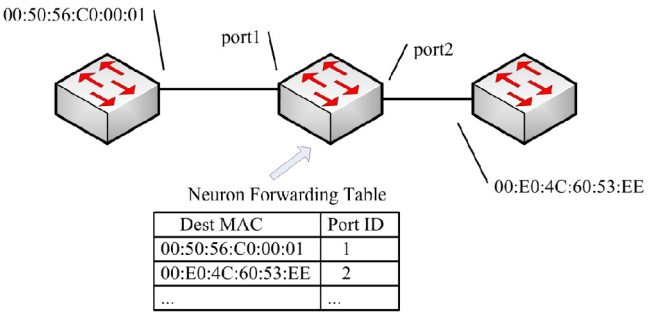

Fig. 6 – Example of neural forwarding table in an Open vSwitch.

normal job, but also the deployment and maintenance of CIPA. It is supposed that the controller knows about the topology of CIPA and can help Open vSwitches to build their neural forwarding tables.

Fig. 7 shows the template of neural message transmitted between Open vSwitches. The first three fields, taking up 14 bytes, are the same as the header of an Ethernet frame. The type field is set to a special value to mark this kind of message. The value in time stamp field can be used to judge whether the message is for the current detecting cycle. The next field tells the number of feature values carried in the message. Each feature value information takes up the following three fields. The feature field is the ID of the feature. The length field shows how many bytes that this feature value takes up. Then comes the real feature value. Finally, padding will be added to the tail of the message if necessary.

4. Evaluation

To test the performance of CIPA, we do some simulations about the detection of DDoS and worm attacks. We compare the results of CIPA with that of the scheme proposed by Gamer (2012), which is a new CIDS with fully distributed structure against large-scale distributed coordinated attacks. The simulation tool is OMNeT++4.3 (OMNeT++, 2013), running on Ubuntu 12.04. The CPU of our machine is Inter(R) Core(TM) i3-2100, and the memory is 8G.

We use BRITE (BRITE, 2013; Heckmann et al., 2003) to generate topologies. BRITE is able to generate scale free and random topologies. The degrees of some nodes in a scale free network are quite large. The attack traffic in these nodes are aggregated so that abnormal can be detected more easily by CIPA. Therefore, in order to make the intrusion detection more difficult, we choose random topologies in our simulations. The algorithm used by BRITE to generate random topologies is Waxman (Medina et al., 2001). Five sizes of topologies are used in our simulations, each of them being networks of 50, 100, 200, 500 and 1000 nodes, in which every node is a programmable switch. We generate 10 different random topologies for each size. Fig. 8 illustrates a 50-switch network topologies used in our simulations.

Normal, DDoS and Worm traffic are needed in the experiments. The normal traffic (CAIDA, 2013a) and DDoS (CAIDA, 2013c) traffic both come from website. We derive two pieces of different normal traffics from the download traces. One is used to train CIPA. The other one is used for the test of CIPA and Gamer. DDoS traffic is used only in test. As for worm traffic, we produce Witty traffic artificially according to Shannon and Moore (2004) and CAIDA (2013b) to imitate the propagation of worm attack. To prove that CIPA is capable of detecting other intrusions, we do some more simulations and experiments with other DDoS tool and worms. The details are presented in Section 4.5.

Since there is no available dataset that provides simultaneous parallel traffic traces on different switches (François et al., 2012), we imitate it through distributing the traffic over the simulated network. We map different IP addresses of traffic into different user IDs. Then we assign the user IDs to all switches in the network uniformly. In this way, each switch has an IP address segment. All user IDs in the same switch are set in adjacent, neighboring switches have similar IP address segments, and the traffic volumes in all directions in the network are more balanced. Hence a switch is the only entry for packets, whose source IP addresses belong to the address segment of that switch, to come into the network. We keep that each switch injects the traffic into the network at a steady rate. For example, the rate of normal flow is 2.28 Mbps for every programmable switch.

We train CIPA off-line with MATLAB. The structure of BP nets corresponding to five different network sizes are 50–9–2, 100–12–2, 200–16–2, 500–24–2 and 1000–34–2. The structure x–y–z means there are x nodes in the input layer, y nodes in the hidden layer and z nodes in the output layer. The switches equipped with hidden node function and/or output node function are randomly selected. Make $a = 2$, and the number of hidden nodes can be calculated by Eq. (12):

$$h = \sqrt{m+n} + a, a \in [1, 10] \tag{12}$$

where h is the number of hidden nodes, m the number of input nodes, and n the number of output nodes. After training, we obtain the weights of links. Then the weights are applied to CIPA when tested.

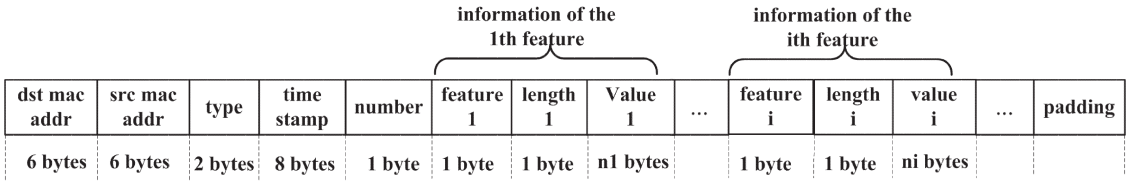

Fig. 7 – Template of neural message transmitted between Open vSwitches.

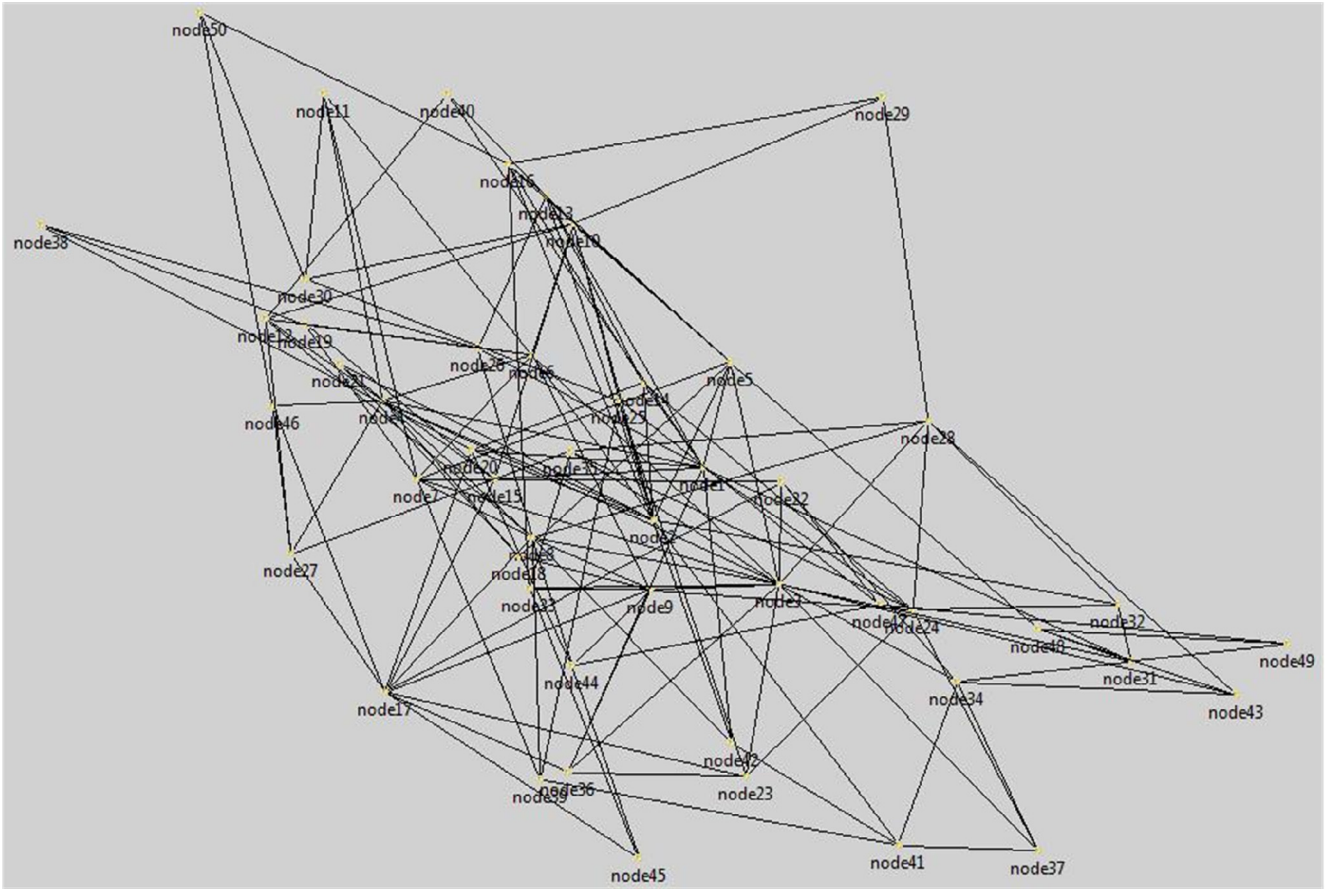

Fig. 8 – An example of random topology with 50 switches generated by BRITE.

4.1. *Detection of DDoS*

The first evaluation scenery is to access the detection capability and scalability of our system against DDoS compared with the scheme proposed by Gamer (2012). The simulation results and discussion are in Sections 4.1 and 4.2.

In order to test the performance of both CIDSs against DDoS of different volumes, we define three types of attack volumes, which are small(S), median(M) and large(L). The datarates of these three DDoS traffic after aggregating at victim are respectively 0.55 Mbps, 1.37 Mbps and 2.74 Mbps. The proportion of the datarate of aggregated DDoS traffic to that of normal traffic injected into network by each switch are respectively 12/50, 30/50 and 60/50.[5] DDoS is such a kind of attack that lots of zombies launch attacks against a single host or server simultaneously. So we randomly choose 30% of the nodes to act as attack sources.[6] To make the results more reliable, we uniformly pick 5% of the nodes to be the only victim by turns.[7] That is to say, there is only one victim node in each trial of simulation. The uniformly placed 5% nodes take turns to be this sink node.

In our simulations for CIPA, $\tau_a = 1$s, $\tau_c = 0.1$s, $\tau_b = \tau_d = 10$ ms. In the light of Section 3.2, in order to capture more intrusion with less features, we utilize 6 features in CIPA. They are respectively[8]:

- *pkt_num*: the number of all packets monitored.
- *icmp_ratio*: the proportion of icmp packets to all packets.
- *slen_ratio*: the proportion of short packets (<70B) to all packets.
- *llen_ratio*: the proportion of long packets (>700B) to all packets.
- *udp_ratio*: the proportion of udp packets to all packets.
- log_{10}(*syn_num/ack_num*): the base-10 logarithm of the proportion of packets with syn flag set to packets with ack flag set.

Without loss of generality, the sampling intervals of these features are all τ_a.

As for the implementation of Gamer, we firstly rank the switches according to their degrees. Then we randomly pick up 2/3 of the top 15% switches. We deploy the IDSs of Gamer in them. It means that 10% of the nodes in network make up the CIDS proposed by Gamer. Each IDS detects its local flow

[5] The data rate of normal traffic is almost 50% of the bandwidth of network links on average.

[6] It means that DDoS traffic comes into network through these nodes.

[7] Actually, these nodes act as the switches connecting the victims.

[8] All values are monitored or calculated during the current sample time interval. The short and long packet thresholds are obtained by trials. They are good enough to make CIPA perform well in our simulations.

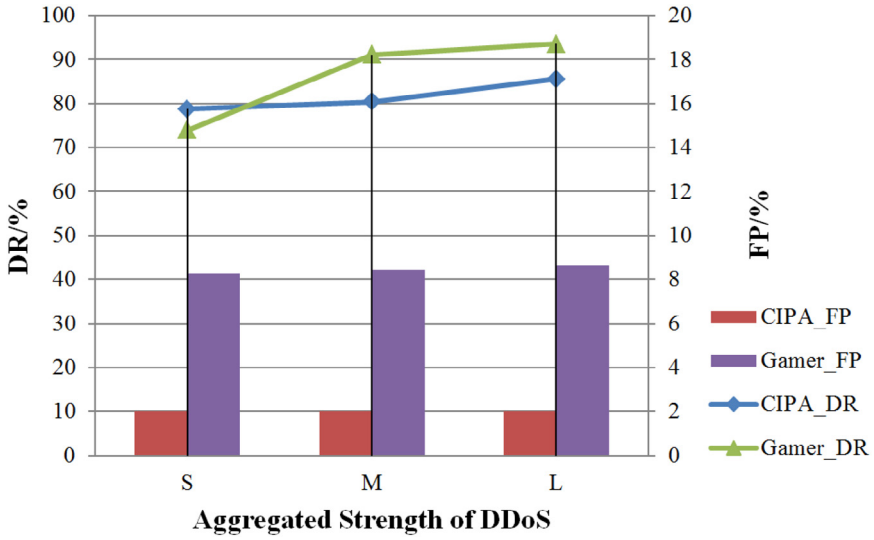

Fig. 9 – Detection results of CIPA against DDoS under 50 node simulated networks.

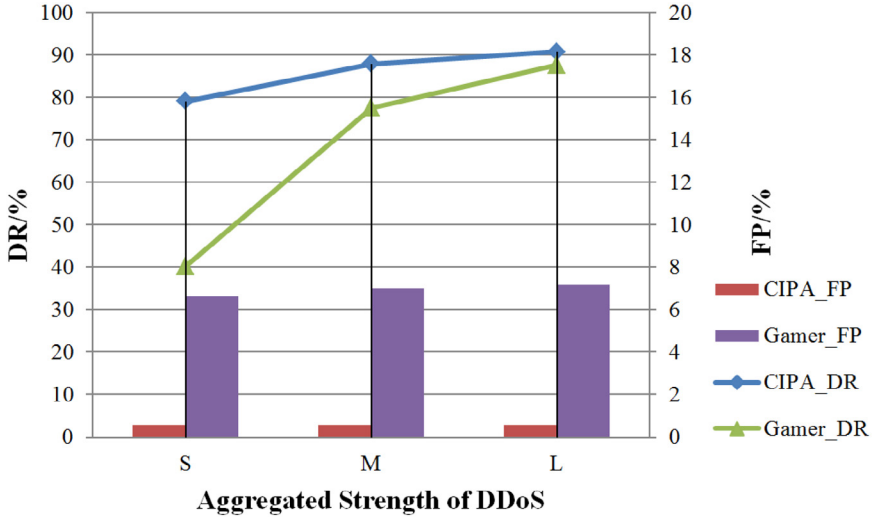

Fig. 10 – Detection results of CIPA against DDoS under 100 node simulated networks.

alone or with some information from neighbors. It sends messages about abnormity to its neighbor IDSs once it discovers intrusion.

After selecting a topology, the attack traffic and the victim node, one can do a trial of simulation. For each combination of topology, attack traffic and victim node, we do 10 trials of simulations to achieve the average result. Therefore, we have done 3 * (10 * 3 * 5% * N) trials to complete all the experiments under each given network size.[9] Each trial lasts 120 simulation seconds. The DDoS begins at the 30 s and lasts for 50 s. When DDoS occurs, the preset nodes replay the DDoS traffic. The destination is the only victim in the network.

In most papers about IDS/IPS, detection rate, DR for short and, false positive rate, FP for short, are used as the main metrics. These two metrics are also adopted to access the detection capacity of CIPA and Gamer. DR and FP used in this paper can be calculated with Eqs. (13) and (14).

$$DR = \frac{number\ of\ abnormal\ samples\ detected}{number\ of\ all\ abnormal\ samples} \tag{13}$$

$$FP = \frac{number\ of\ normal\ samples\ detected\ as\ abnormal}{number\ of\ all\ normal\ samples} \tag{14}$$

The simulation results about the detection of DDoS are showed in Figs. 9–11. All the values are the averages of the results from several trials of simulations under the corresponding network size and attack strength.

Since the simulations of network with 500 and 1000 nodes take too much time, we just do dozens of trials under such sizes of networks to prove that CIPA performs well in very large scale networks. The results are shown in Table 2. All the DRs and

[9] N is the network size. $2*(10*3*5\%*N)$ trials are for the training and test of CIPA. $1*(10*3*5\%*N)$ trials are for the test of Gamer.

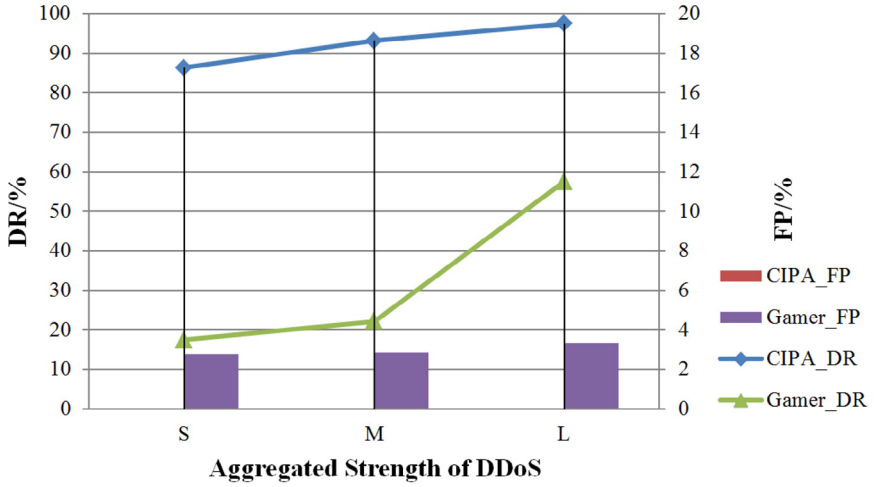

Fig. 11 – Detection results of CIPA against DDoS under 200 node simulated networks.

Table 2 – The simulation results of networks with 500 and 1000 nodes.

	Size/Nodes	500	1000
DDoS	DR/%	90.17	92.43
	FP/%	0	0
	msg/Mbps	6.18	18.83
Witty	DR/%	96.67	97.05
	FP/%	0	0
	msg/Mbps	6.18	18.83

FPs in Table 2 are the averages from the situation of small attack volume, i.e. the worst case. They are in accordance with the results and analysis from Sections 4.1–4.3.

Through Figs. 9–11, we can find that, in network of 50 nodes, the DR of CIPA is not as good as Gamer. But with the grow of network size, the DR of Gamer drops obviously while the DR of CIPA increases gradually. The FP of CIPA is much lower than that of Gamer in all sizes of networks.[10] In the same size of network, the DRs of both CIDS improve as the aggregated volume of attack traffic increases.

As the network becomes larger, DDoS of the same aggregated volume will be diluted to more attack sources. It makes the attack rate of each node decreases. In other words, the intrusion flow becomes more dispersed before aggregating at the victim. Every IDS in the scheme of Gamer does the first detection with only local view, so that they are not good at discovering the secret small attack flow. So, as the size of the network grows, attack flow becomes more distributed and the DR of Gamer decreases. Because most IDSs of Gamer are deployed on nodes of relatively large degree, and the adjacent IDSs share intrusion information, the DR of Gamer would not drop to zero.

From Figs. 9–11, we can see that FP of Gamer decreases on the whole as the network size increases. The reason for this is that IDSs of Gamer base their judgment on arguments whose on-line update will be influenced by weak attack flow. For instance, when the DDoS volume is weak, there should be more attack flows escaping from being detected. Those attack flows are treated as normal samples and used to update the thresholds. It lets the threshold move a little close to the statistical property of intrusion flow. Then there will be less attack flow to be detected and less fluctuating normal traffic to be misclassified as abnormal. It means the drop of DR and FP. On the other hand, FP drops a little as attack traffic decreases in the same size of network. It tells us that the effect on FP caused by the increase of attack volume is less than that of the growth of the network size.

No matter how dispersed the flow is, seen from the whole network, its aggregated volume is still large enough. When doing the first time detection, CIPA is already provided with global view. In this way, CIPA can resist the feature fluctuation of normal traffic and capture the intrusion behavior at a sight of the entire network. What is more, due to the property of ANN, the more dispersed the DDoS flow is, the more monitors may be affected. The follow-up calculation nodes are more likely to find out the abnormity. Consequently, as the network grows, the DDoS traffic becomes more scattered and then the DR of CIPA increases.

Why the feature fluctuation of normal traffic has less effect on CIPA? It can be explained as follows: there are time-lags among the fluctuation of normal traffic of different links, which means no global synchronous phenomenon occurs. Then a few input neurons will be affected by it. So the probability of misclassification in the following calculation is lower. Moreover, as the network size grows, because of no global synchronous, the proportion of input neurons influenced by the local fluctuation would be less. Thus the FP of CIPA decreases. In our simulations, the weights of ANNs are not updated during testing. So the FP of CIPA is almost the same under the same size of the network and normal traffic.

4.2. *Communication and computation overhead*

The scalability of a CIDS means how its performance and overhead change as the network size grows or the attack volume increases. Section 4.1 shows us that the detection capacity of

[10] The FP of CIPA in network of 200 nodes is zero.

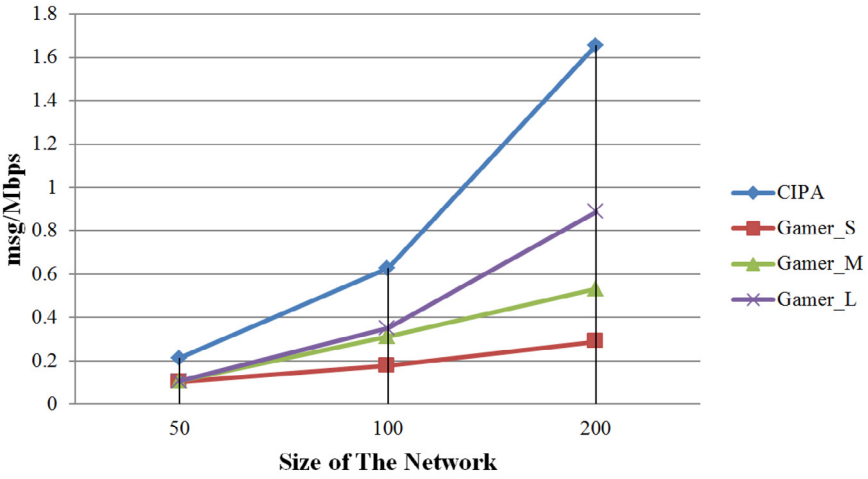

Fig. 12 – Comparison of communication overhead of two CIDS when DDoS occurs.

CIPA increases when the network becomes larger. In this subsection, the communication and computation overhead of the two CIDS schemes will be discussed.

When network is suffering from flooding attack, if CIDS has to send out a large amount of messages to realize correlation, the network will be more jammed. Bad scalable CIDS may even cause new flooding during correlation process. All in all, the communication overhead caused by correlation process of CIDS should be within an acceptable level no matter how large the network is and how heavy the attack is. The communication overheads corresponding to the experiments in Section 4.1 are shown in Fig. 12.

Fig. 12 shows that: (a) Under the same attack volume, the communication overhead of Gamer increases as the network size grows. (b) In the same size of network, the overhead of Gamer increases along with the increases of attack volume. The reason for it is that IDSs of Gamer send out messages only when they find intrusion by themselves. The more IDSs find abnormity, the more messages are sent. However, the communication overhead of CIPA only increases along with the growing of network size. The messages mainly come from the periodical sending of traffic features by input nodes. The overhead exists no matter there is intrusion or not.

For the given network sizes and attack volumes used in our simulations, the communication overhead of Gamer is less than that of CIPA. However, even in network of 200 nodes, the instantaneous messages generating rate of an input node is 8 kbps, and the rate of messages converging to a hidden node is 100 kbps. Both are much lower than the normal traffic generating rate of each node and volume of the aggregated attack.

The computation overhead of CIPA mainly comes from two parts: (a) the packet parsing and variables updating operations of input neurons; (b) the ANN computation of the whole network. We assume that the number of input neurons is N. According to Eq. (12), the number of hidden neurons is about $N^{1/2}$. The number of output neurons is a constant. The computation complexity of hidden layer is $N^{1/2} * O(N) = O(N^{3/2})$. The computation complexity of output layer is $O(N^{1/2})$. Then the complexity of ANN computation is $O(N^{3/2}) + O(N^{1/2}) = O(N^{3/2})$. The complexity of packet parsing and variables updating operations of a neuron is $O(c) = O(1)$, where c is constant. Hence, the computation complexity of CIPA is $N * O(c) + O(N^{3/2}) = O(N^{3/2})$. In the simulation, the number of switches is the same as that of input neurons, i.e. N. The computation complexity of Gamers scheme comes from the packet parsing and variables updating operations and the detection algorithm of the IDSs. The detection algorithm needs the alerts from adjacent IDSs. We assume there are k adjacent IDSs. Generally, k increases linearly with N. Then the computation complexity of detection algorithm is $N * 10\% * O(k * c) = O(N^2)$. The overhead for packet parsing and variable updating operations is $N * 10\% * O(c) = O(N)$. Hence, the computation complexity of Gamers scheme is $O(N^2) + O(N) = O(N^2)$. It is larger than the computation complexity of CIPA.

To sum up, CIPA satisfies two features of scalability well.

4.3. *Detection of witty worm*

Since there is no experiments about worm in Gamer (2012), we only provide the detection results of CIPA against Witty Worm in this subsection. We analyze the Witty traffic trace from CAIDA (2013b) and the prorogation of Witty (Shannon and Moore, 2004). Then we decide to imitate the outbreak of Witty artificially. The details are as follows. The topologies, normal traffic, flow features, and weights of links are almost the same as those at DDoS experiment. We do not need to retrain the ANNs shown in Section 4.1 if the topologies and the features of normal traffic are not changed. The Witty attack begins from 70 s. It is forced to stop at 120 s. When attack begins, the randomly selected 5% nodes play the roles of initial attack sources. They generate attack packets according to the property of Witty. Other nodes randomly open some ports. Once a node is infected, it becomes a new attack source and starts to send out Witty packets. After sending out 20,000 Witty packets, Shannon and Moore (2004) says that the injected host will crash. At this moment, instead of letting the node down, we just stop it from

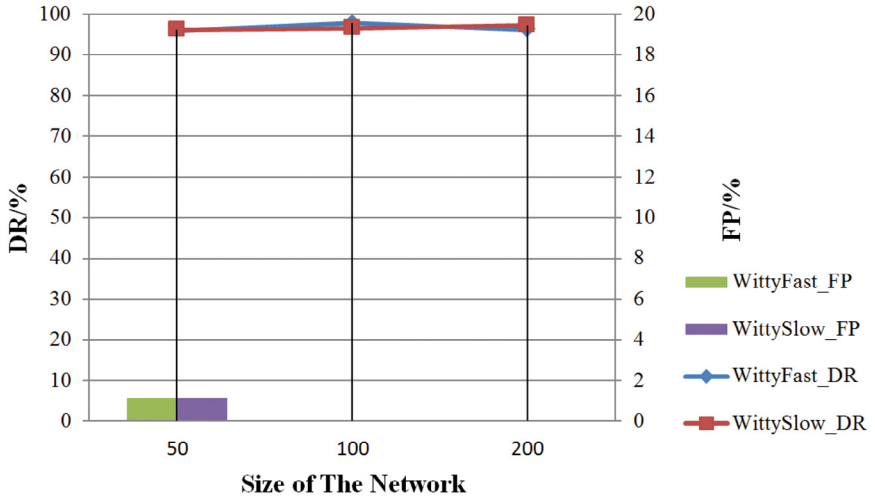

Fig. 13 – Detection results of CIPA against Witty under different sizes of simulated networks.

sending Witty packet.[11] We control the attack rates of the sources and realize two types of attack rates, which are WittyFast and WittySlow. The datarates of them are respectively 54.72 kbps and 27.36 kbps. In other words, the proportions of the datarate of each attack source to normal traffic are 1.2/50 and 0.6/50. The results are in Fig. 13. We execute the simulations in the similar way as Section 4.1.

From Fig. 13, we can know that the DRs of CIPA against Witty of two different rates are high, all beyond 96%. Meanwhile, the FPs are below 1.5%.[12] Compare Fig. 13 with Figs. 9–11, we can easily find that CIPA is much more good at detecting Witty worm than DDoS. This is because DDoS is a fixed many-to-one attack while Witty is an one-to-many spreading attack. The hosts infected by Witty become the new sources of the worm propagation. So that more input nodes detect the features of Witty and CIPA is more likely to discover the anomaly.

No matter there exist attacks or not and what kinds of attacks are happening, the communication and computation overhead of CIPA is the same as that showed in Section 4.2.

4.4. *Experiments in openflow-based SDN*

In this paper, we deploy CIPA in the kernel space of Open vSwitch. Since linux kernel does not have ready complex mathematic functions, like exponential function, we utilize Taylor series (Wikipedia, 2015) to approximate them. The controller is POX. The openflow v1.0 is used. Considering the resource limitation, we evaluate the performance of CIPA in a real-world openflow-based SDN with the topology showed in Fig. 14.

We use three PCs to build this environment. Each PC runs 3 or 4 VMware virtual machines as illustrated in Table 3. The OSs of virtual machines are all Ubuntu 12.04. The version of Open vSwitch is 1.9.3. The architectures of ANNs are all 3–2–2. Table 3 also shows what neurons are in each switch. The transform function in hidden and output neurons are the nonlinear sigmoid functions.

In each host, we use Scapy (2013) to generate the normal traffic and/or attack traffic. According to Braga et al. (2010), we also set the ratio of different protocols in normal traffic as: TCP 85%, UDP 10% and ICMP 5%. We add some changes to the ratio during some random time intervals to make the normal traffic more fluctuant. As for attack types, IPscan, UDP flood and SYN flood are used. In the experiment of this section, $\tau_a = 10$ s, $\tau_c = 1$ s and $\tau_b = \tau_d = 0.1$ s. Since the network size is small, there are not so many packets in each second. We set large τ_a to monitor as many packets as possible in each sample cycle such that the ratio of different protocols and the statistics approximate to real.

The features we selected here are *icmp_ratio*, *udp_ratio* and (*syn_num* – *ack_num*)*_ratio*.[13] Without loss of generality, the sampling intervals of these features are all τ_a. Each round of experiment lasts for 1000 s. The normal traffic presents during all the experiment period. Three waves of attack traffic for every scenario are injected into the network. One lasts for 100 s. The other two last for 150 s. Then every round of the experiment generates 100 feature samples. 60 samples of them are generated during the normal traffic period, while other 40 samples come from the mixed flow of normal and attack traffic. Table 4 shows which hosts act as attackers or victims in different attack scenarios and the corresponding experiment results. We run each scenario 10 times and the results are the averages. The experiment results in Table 4 show that CIPA works well in our openflow-based SDN environment. It has high detection rate and low false positive rate when going against these three types of intrusions. We think that the reasons for its good performance are as follows: the network is relatively small and simple; from Fig. 14 and Table 4, we can know that attack traffic goes through all Open vSwitches in each intrusion scenario. The more input neurons are affected, the more likely that CIPA is able to discover the intrusion.

[11] Then this node recovers, opens some ports, generates or forwards packets and waits to be injected again.

[12] The FPs of CIPA under network of 100 and 200 nodes are both zeros.

[13] (*syn_num* – *ack_num*)*_ratio* means the proportion of the numeric difference between packets with syn flag set and packets with ack flag set to all packets.

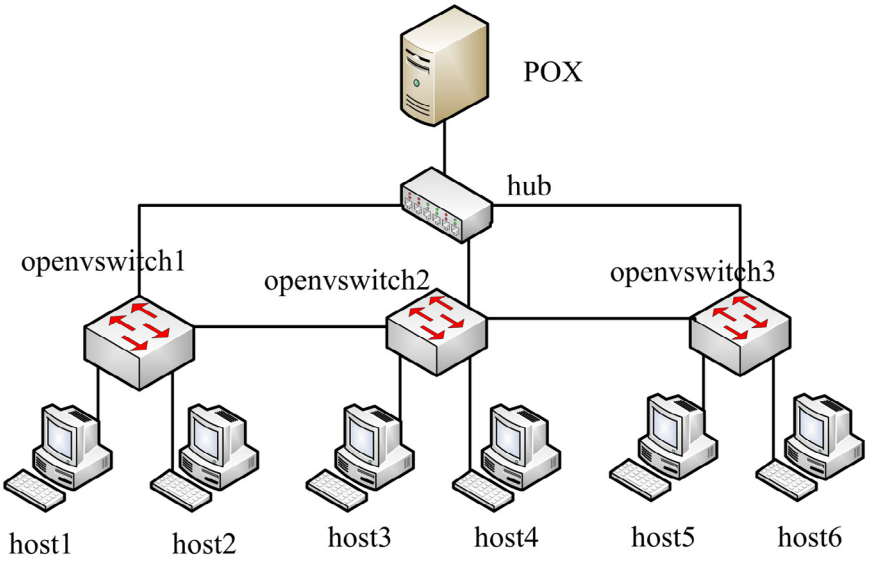

Fig. 14 – Topology of SDN experiment.

4.5. *Other simulations and experiments*

To prove that CIPA is capable of detecting other intrusions, we do some more simulations and experiments. We use LOIC (Low Orbit Ion Cannon) (Batishchev, 2015), a popular DDoS tool in the SDN environment to achieve more realistic scenarios. The experiment environment is the same as that of Section 4.4. The normal traffic and the parameters settings of the experiments are also the same. We use host1 and host5 as the attackers to launch UDP, TCP and HTTP DDoS attacks. The victim is still host2. The results are showed in Table 5. We utilize the trained ANNs used in Subsection 4.4 and we do not need to retrain it. The detection results are still good enough.

Since we cannot use attack tools in OMNet++ directly, we study how LOIC works and realize the similar DDoS behavior in the simulations. The simulation environment, the normal traffic and the parameters settings are the same as those of Section 4.1. But we just do several simulations to test the performance of CIPA against LOIC. The DDoS type we choose is UDP attack. The results are shown in Fig. 15. In addition, we study different families of worms and select two of them, i.e. Slammer (Moore et al., 2003) and Conficker (Shin and Gu, 2010; CAIDA, 2015; Porras et al., 2009). They are both famous worms and Conficker is a relatively new kind of worm. Similar to Witty, we learn how they propagate and imitate their behaviors in the simulations. The simulation environment and the parameter settings are almost the same as those of Section 4.3. We also just do several simulations to test the performance of CIPA against Slammer and Conficker. The results are shown in Fig. 15. We utilize the trained ANNs used in Sections 4.1 and 4.3. We do not need to retrain them. The detection results prove that CIPA work well.

All in all, once the network features are properly selected and CIPA is well trained, instead of detecting just some specific distributed attacks, CIPA can detect more intrusions. The reasons for its good performance are as follows.

We train the ANNs of CIPA with only normal traffic so that CIPA is an anomaly detection system. The weights of the ANNs represent the outline of normal traffic and can be used to discover many anomalies or attacks. The similar works can be found in Zhang et al. (2005) and Fan et al. (2001). In the solution of Zhang et al. (2005), the ANN, acting as the first level anomaly classifier, are trained with only normal traffic. The results of anomaly detection are good enough. While in Fan et al. (2001) the ANN is trained with normal and artificial

Table 3 – Virtual machines on each PC and neurons in each Open vSwitch. The architectures of ANNs are all 3-2-2. The last column illustrates the neurons in each Open vSwitch. The letters i, h and o respectively represent an input, hidden and output neurons.

Computer	Network adapter	VMware	Neurons
PC1	2	Open vSwitch1, host1, host2, POX	i, h, o
PC2	3	Open vSwitch2, host3, host4	i, o
PC3	2	Open vSwitch3, host5, host6	i, h

Table 4 – The attackers, victims and experiment results of each attack scenario in SDN environment.

Scenario	Attackers	Victims	DR (%)	FP (%)
IPscan	host1	host2–6	96.8	3.16
UDP flood	host1, host5	host2	97.5	3.17
SYN flood	host1, host5	host2	95.5	2.33

Table 5 – Experiment results of CIPA against DDoS on SDN. The DDoS traffic is generated by LOIC.

Attack type	DR (%)	FP (%)
UDP attack	97.1	3.32
TCP attack	95.2	2.34
HTTP attack	94.6	2.67

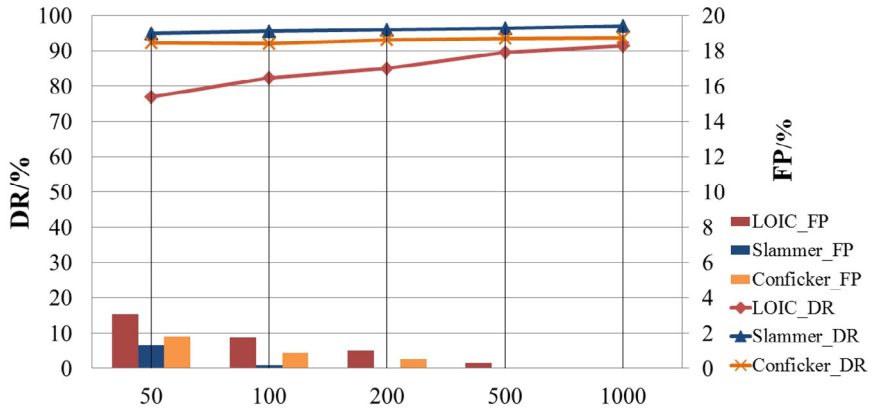

Fig. 15 – Detection results of CIPA against LOIC, Slammer and Conficker under different sizes of simulated networks.

abnormal samples, which is helpful for the ANN to detect abnormal traffic. The difference from CIPA is that they are not distributed.

Then we use an approach similar to ensemble of ANNs (Golovko et al., 2007; Hansen and Salamon, 1990; Zhang et al., 2005) to improve the generalization ability of the system. The details are presented in the last paragraph of Section 3.4. In considering that variants of an attack often have similar features, and some of different attacks may cause similar effects on the networks, a given set of features can be used to detect many attacks and their variants. For example, Slammer and Witty worms are both self-carried UDP worms. During their fast propagation, they will scan the network. The network features, such as *pkt_num*, *icmp_ratio* and *udp_ratio*, will be affected and become higher. Another example is Conficker. Conficker is a relatively new kind of TCP worms and it targets several Microsoft Windows Operating Systems. It seems that Conficker has less in common with Witty. However, after we have studied the traffic trace of Conficker (CAIDA, 2015) and its propagation properties (Porras et al., 2009), we find that they still have something in common. They both scan the network. They both affect these features: *pkt_num*, *icmp_ratio* and *udp_ratio*.

5. Conclusion

This paper proposed a distributed intrusion prevention system, CIPA, for programmable network and SDN, aiming at intrusion detection and mitigation. The architecture, core algorithm and implementation of CIPA were described in details.

CIPA is an ANN-based CIDS. It disperses the computational capability to part or all of the switches in the network. Each of them just requires little resources for the simple computation of neurons. Based on a global view, CIPA is good at detecting large scale intrusion behavior even if the attack volume observed at each node/link is weak compared with the normal traffic. Taking advantages of the virtualization capability and programmability of programmable switches, CIPA is easy to be deployed and adjusted to fit to different detection strategies. The parallel and simple computational capabilities of neurons in ANN make CIPA scalable and applicable in a large scale network with lower computation and communication overhead. Simulations showed that CIPA does better than the existing CIDS proposed by Gamer (2012) when aiming at detecting flooding DDoS. CIPA also performs very well at discovering Witty, Slammer and Conficker worm outbreak. The scalability of CIPA is proved good as well. At last, we realized a prototype of CIPA for openflow-based SDN and evaluated it in a small real-world SDN environment. The experiment results showed that CIPA has good performance when implemented in SDN.

As for future works, we plan to deploy CIPA in a large scale real-world environment for detecting more sorts of intrusions.

REFERENCES

Balasubramaniyan JS, Garcia-Fernandez JO, Isacoff D, Spafford E, Zamboni D. An architecture for intrusion detection using autonomous agents. In: Proceedings of the 14th annual of computer security applications conference. IEEE. 1998. p. 13–24.

Batishchev A. LOIC: a network stress testing application, <http://sourceforge.net/projects/loic/>; 2015.

Braga R, Mota E, Passito A. Lightweight DDoS flooding attack detection using NOX / OpenFlow. In: Proceedings of the 35th IEEE conference on local computer networks (LCN). IEEE; 2010. p. 408–15.

BRITE. BRITE: Boston university representative internet topology generator, <http://www.cs.bu.edu/brite/>; 2013.

Cannady J. Artificial neural networks for misuse detection. In: Proceedings of the National information systems security conference. 1998. p. 368–81.

CAIDA. The CAIDA anonymized internet traces 2012 dataset, <https://data.caida.org/datasets/passive-2012/>; 2013a.

CAIDA. The CAIDA dataset on the witty worm, <https://data.caida.org/datasets/security/witty/>; 2013b.

CAIDA. The CAIDA "DDoS attack 2007" dataset, <https://data.caida.org/datasets/security/ddos-20070804/>; 2013c.

CAIDA. Network telescope: three days of conficker, <https://data.caida.org/datasets/security/telescope-3days-conficker/>; 2015.

Chung CJ, Khatkar P, Xing T, Lee J, Huang D. NICE: network intrusion detection and countermeasure selection in

virtual network systems. IEEE Trans Dep Sec Comput 2013; 1.
Cuppens F, Miège A. Alert correlation in a cooperative intrusion detection framework. In: Proceedings of the IEEE symposium on security and privacy. IEEE; 2002. p. 202–15.
Dumitrescu C. INTCTD: a peer-to-peer approach for intrusion detection. In: Proceedings of the 6th IEEE international symposium on cluster computing and the grid, vol. 1. IEEE; 2006. p. 89–92.
Fan W, Miller M, Stolfo SJ, Lee W, Chan PK. Using artificial anomalies to detect unknown and known network intrusions. In: 2013 IEEE 13th international conference on data mining. IEEE Computer Society; 2001. p. 123.
Fossi M, Egan G, Haley K, Johnson E, Mack T, Adams T, et al. Symantec Internet security threat report trends for 2010, vol. 16. 2011 20.
François J, Aib I, Boutaba R. FireCol: a collaborative protection network for the detection of flooding ddos attacks. IEEE/ACM Trans Netw 2012;20(6):1828–41.
Fung C. Collaborative intrusion detection networks and insider attacks. JoWUA 2011;2(1):63–74.
Gamer T. Collaborative anomaly-based detection of large-scale Internet attacks. Comput Netw 2012;56(1): 169–85.
Golovko V, Kachurka P, Vaitsekhovich L. Neural network ensembles for intrusion detection. In: Intelligent data acquisition and advanced computing systems: technology and applications, 2007. IDAACS 2007. 4th IEEE workshop on. IEEE. 2007. p. 578–83.
Hamilton D. DDoS attack takes Playstation network and Sony Entertainment network offline, <http://www.datacenterknowledge.com/archives/2014/08/25/ddos-attack-takes-playstation-network-sony-entertainment-network-offline>; 2015.
Hansen LK, Salamon P. Neural network ensembles. IEEE Trans Pat Anal Mach Intel 1990;12(10):993–1001.
Haykin S. Neural networks: a comprehensive foundation. 3rd ed. Upper Saddle River, NJ: Pearson Prentice Hall.; 2007. p. 122–217.
Heckmann O, Piringer M, Schmitt J, Steinmetz R. On realistic network topologies for simulation. In: Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research. ACM; 2003. p. 28–32.
Huang M, Jasper R, Wicks T. A large scale distributed intrusion detection framework based on attack strategy analysis. Comput Netw 1999;31(23):2465–75.
Janakiraman R, Waldvogel M, Zhang Q. Indra: a peer-to-peer approach to network intrusion detection and prevention. In: Proceedings of the 12th IEEE international workshops on enabling technologies: infrastructure for collaborative enterprises (WETICE). IEEE. 2003. p. 226–31.
Kemmerer R. NSTAT: a model-based real-time network intrusion detection system. University of California-Santa Barbara Technical Report TRCS97 1997; 18.
Li P, Salour M, Su X. A survey of Internet worm detection and containment. IEEE Commun Surv Tutor 2008;10(1): 20–35.
McKeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, et al. OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Comput Commun Rev 2008;38(2):69–74.
Medina A, Lakhina A, Matta I, Byers J. BRITE: universal topology generator from a users perspective. Technical Report; Technical Report, BUCS-TR-2001; 2001.
Mehdi SA, Khalid J, Khayam SA. Revisiting traffic anomaly detection using software defined networking. In: Proceedings of the conference on recent advances in intrusion detection. Springer; 2011. p. 161–80.
Moore D, Paxson V, Savage S, Shannon C, Staniford S, Weaver N. Inside the slammer worm. IEEE Secur Priv 2003;1(4):33–9.
MurphyMc, The POX Controller, <https://github.com/noxrepo/pox/>; 2015.
OMNeT++. Welcome to the OMNeT++ Community!, <http://www.omnetpp.org/>; 2013.
ONF. Software-defined networking: the new norm for networks, <https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf>; 2013.
OVS. Open vSwitch, an open virtual switch, <http://openvswitch.org/>; 2013.
Porras P, Saidi H, Yegneswaran V. Conficker c analysis. SRI International; 2009.
Radware. DDoS Protection as a Native SDN Application by Radware & Big Switch Networks. <https://www.radware.com/Solutions/SDN-Resources>; 2013.
Radware. 2014–2015 Global Application and Network Security Report, <http://www.radware.com/ert-report-2014/>; 2015.
Riedmiller M, Braun H. A direct adaptive method for faster backpropagation learning: the rprop algorithm. In: Proceedings of the IEEE international conference on neural networks. IEEE; 1993. p. 586–91.
Rumelhart D, Hinton G, Williams R. Learning representations by back-propagating errors. Cogn Model 2002;1:213.
Scapy, Scapy, <http://www.secdev.org/projects/scapy/>; 2013.
Shannon C, Moore D. The spread of the witty worm. IEEE Secur Priv 2004;2(4):46–50.
Shin S, Gu G. Conficker and beyond: a large-scale empirical study. In: Proceedings of the 26th annual computer security applications conference. New York, NY, USA: ACM; ACSAC '10; 2010. p. 151–60.
Shin S, Yegneswaran V, Porras P, Gu G. AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks. In: Proceedings of the 2013 ACM SIGSAC conference on computer & communications security. ACM; 2013. p. 413–24.
Shirali-Shahreza S, Ganjali Y. Efficient implementation of security applications in openflow controller with FleXam. In: Proceedings of the 21st IEEE annual symposium on high-performance interconnects (HOTI). IEEE; 2013a. p. 49–54.
Shirali-Shahreza S, Ganjali Y. FleXam: flexible sampling extension for monitoring and security applications in openflow. In: Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking. ACM; 2013b. p. 167–8.
Snapp S, Brentano J, Dias G, Goan T, Heberlein L, Ho C, et al. DIDS (distributed intrusion detection system)-motivation, architecture, and an early prototype. In: Proceedings of the 14th National Computer Security Conference. 1991. p. 167–76.
Suresh P, Merz R. ns-3-click: click modular router integration for ns-3. In: Proceedings of the 4th international ICST conference on simulation tools and techniques. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering); 2011. p. 423–30.
Symantec. 2014 Internet Security Threat Report, <http://www.symantec.com/security_response/publications/threatreport.jsp>; 2015.
Tian D, Liu Y, Li B. Anomaly intrusion detection methods for peer-to-peer system. In: Proceedings of the IFIp international

conference on network and parallel computing workshops. IEEE; 2007. p. 127–30.
Tootoonchian, The NOX Controller, <https://github.com/noxrepo/nox/>; 2015.
Vincent Zhou C, Leckie C, Karunasekera S. Decentralized multi-dimensional alert correlation for collaborative intrusion detection. J Netw Comput Appl 2009;32(5):1106–23.
Vlachos V, Androutsellis-Theotokis S, Spinellis D. Security applications of peer-to-peer networks. Comput Netw 2004;45(2):195–205.
Wikipedia. Taylor series, <http://en.wikipedia.org/wiki/Taylor_series>; 2015.
Xing T, Huang D, Xu L, Chung CJ, Khatkar P. SnortFlow: a openflow-based intrusion prevention system in cloud environment. In: Proceedings of the second GENI research and educational experiment workshop (GREE). IEEE; 2013. p. 89–92.
Yegneswaran V, Barford P, Jha S. Global intrusion detection in the domino overlay system. In: Proceedings of NDSS, vol. 2004. San Diego, CA: 2004.
Zhang C, Jiang J, Kamel M. Intrusion detection using hierarchical neural networks. Pattern Recogn Lett 2005;26(6):779–91.
Zhang Z, Li J, Manikopoulos C, Jorgenson J, Ucles J. HIDE: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification. In: Proceedings of the IEEE workshop on information assurance and security. 2001. p. 85–90.
Zhou C, Leckie C, Karunasekera S. A survey of coordinated attacks and collaborative intrusion detection. Comput Secur 2010;29(1):124–40.

Xiao-Fan Chen received the B.S. degrees from Sun Yat-Sen University, Guangzhou, China. He is currently working toward the Ph.D. degree at Sun Yat-Sen University. His research interests are in network security, software-defined networks and AI algorithms.

Shun-Zheng Yu (M'08) received the B.Sc. degree from Peking University, Beijing, China, and the M.Sc. and Ph.D. degrees from the Beijing University of Posts and Telecommunications, China. He was a visiting scholar at Princeton University and IBM Thomas J. Watson Research Center during 1999–2002. He is currently a Professor at the School of Information Science and Technology, Sun Yat-Sen University, Guangzhou, China. His recent research interests include networking, traffic modeling, anomaly detection and algorithms for hidden semi-Markov models.