


Research Article

An Adaptive IP Hopping Approach for Moving Target Defense Using a Light-Weight CNN Detector

Xiaoyu Xu ^{1,2} Hao Hu,² Yuling Liu,^{3,4} Hongqi Zhang,² and Dexian Chang²

¹State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, China

²Zhengzhou Information Science and Technology Institute, Zhengzhou 450001, China

³Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China

⁴School of Cyber Security, University of Chinese Academy of Sciences, Beijing 101408, China

Correspondence should be addressed to Xiaoyu Xu; xxyin1992@163.com

Received 24 September 2020; Revised 24 June 2021; Accepted 31 July 2021; Published 15 August 2021

Academic Editor: Mohamed Amine Ferrag

Copyright © 2021 Xiaoyu Xu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Scanning attack is normally the first step of many other network attacks such as DDoS and propagation worm. Because of easy implementation and high returns, scanning attack especially cooperative scanning attack is widely used by hackers, which has become a serious threat to network security. In order to defend against scanning attack, this paper proposes an adaptive IP hopping in software defined network for moving target defense (MTD). In order to accurately respond to attacker's behavior in real time, a light-weight convolutional neural network (CNN) detector composed of three convolutional modules and a judgment module is proposed to sense scanning attack. Input data of the detector is generated via designed packets sampling and data preprocess. The detection result of the detector is used to trigger IP hopping. In order to provide some fault tolerance for the CNN detector, IP hopping can also be triggered by a preset timer. The CNN driving adaptability is applied to a three-level hopping strategy to make the MTD system optimize its behavior according to real time attack. Experiments show that compared with existing technologies, our proposed method can significantly improve the defense effect to mitigate scanning attack and its subsequent attacks which are based on hit list. Hopping frequency of the proposed method is also lower than that of other methods, so the proposed method shows lower system overhead.

1. Introduction

According to Symantec's 2019 report [1], a growing number of people and organizations display an interest in compromising operational computers via network. The static properties of network make the state and behavior of information system predictable, so attackers can not only launch attack effectively, but also escape detection easily [2]. Methods of network and host properties randomization such as Moving Targets Defense (MTD) have been recommended as a countermeasure against reconnaissance that attacks the static and predictable property of network [3]. IP hopping is one of the key technologies of MTD. It frequently changes IP addresses of protected nodes in network in order to prevent attackers from creating effective hit list.

Existing researches [4–22] have proved that IP hopping technology is an effective method to defend against scanning attack which is normally the first step of many other network attacks such as DDoS [23] and propagation worm [24]. The game between scanning attacker and IP hopping defender is shown in Figure 1. Scanning attacks are generally manifested as attackers continuously release different forms of probe packets. Attackers will know which hosts in the network are potential targets according to the response packets received, while IP hopping technology frequently changes the used IP addresses of protected hosts. On the one hand, IP hopping makes the protected hosts avoid scanning attack to some extent. On the other hand, IP hopping could make the probed targets of attacker invalid in attacker's hit list in a short time. For example, once the attacker probed a target

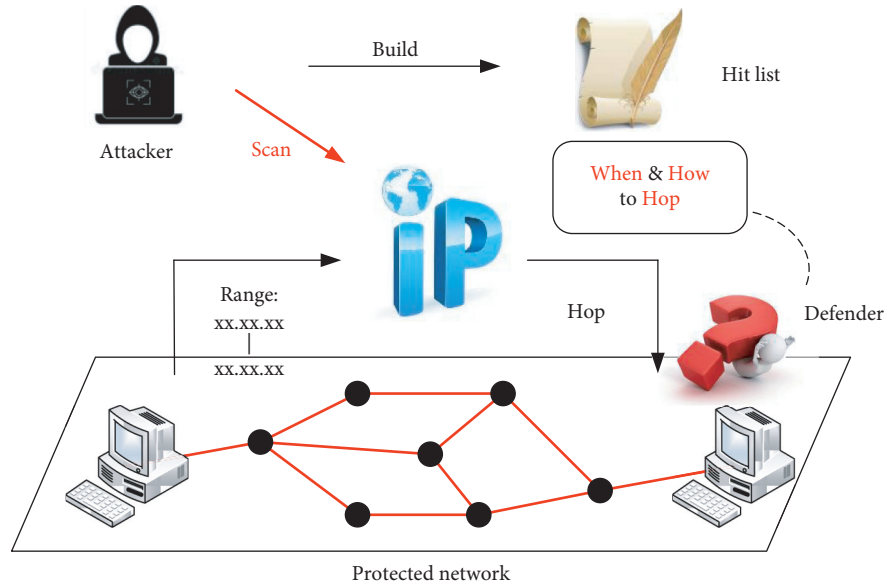


FIGURE 1: Background: game between the scanning attacker and IP hopping defender.

and added its IP to the hit list, the IP may be invalid soon under the protection of IP hopping. The main challenge of defender is to decide when and how to hop.

IP hopping technology is firstly proposed and deployed in legacy network [2, 4–12] and proved to be an excellent defensive means. In recent years, software defined network (SDN) [25] has been widely studied and considered as the next network technology. SDN provides an operation mode of separating data forwarding and rule control. The flexibility of SDN brings conveniences and supports many new network technologies. Meanwhile, SDN also faces endless network threats. IP hopping technology applied to SDN has also been researched in many papers [11–20]. They usually use OpenFlow [26, 27] to develop a MTD architecture for IP hopping defense. The controller in SDN controls the mutation of IP of protected hosts. The controller also manages switches in protected network to ensure timely and accurate forwarding of packets in situation of constantly changing terminal IP addresses. The structure of SDN provides flexible convenience to deploy IP hopping defense technology.

Faced with scanning attack, most of the researches do not pay enough attention to the behavior of attackers. The existing IP hopping defense systems lack the necessary perceptual ability to the attackers' behavior. For instance, when no attacker invades the protected network, IP addresses of nodes keep hopping. In addition, the IP hopping system may probably be disruptive to legitimate user behavior. Some researches propose adaptive IP hopping methods [16, 19–22]. Adaptive methods can configure IP hopping scheme and execute IP hopping according to behavior of attackers so as to achieve elaborate defense effect. However, there is still much to improve in existing adaptive methods in reducing the success rate of scanning attacks and reducing the average life time of scanned targets in hit list.

This paper proposes a novel adaptive IP hopping defense method in SDN. In our method, a light-weight convolutional neural network (CNN) [28, 29] is used to perceive the

attackers' scanning behavior and guide the configuration of next IP hopping. Experiments show that our proposed method can significantly improve the survival rate of protected hosts under scanning attack while reducing the average lifetime of targets in attackers' hit list to avoid DDoS and other subsequent attacks. The main contribution of this paper can be concluded as follows. Firstly, we propose a sampling and data preprocess method of attacker's behavior data in SDN. Secondly, we designed a light-weight CNN structure which can be used to detect attacker's behavior in real time and drive IP hopping defense. Thirdly, applying CNN driving adaptability to three-level hopping strategy, a novel adaptive IP hopping in SDN is proposed and proved to achieve better defense effect via considerable amount of experiments.

The rest of this paper is organized as follows. Section 2 gives an overview of related works. Section 3 introduces the motivation of our proposed method. Section 4 presents our proposed method in detail. The implementation and experiments are presented in Section 5. Finally, Section 6 concludes this paper.

2. Related Works

Researches of IP hopping for MTD can be classified into two categories according to implementation platform, which are legacy network and SDN. We list both technologies in legacy network and SDN in detail.

Traditional IP address randomization techniques such as DHCP [30] or NAT [31] do not develop the potential of IP randomization in network defense completely for the reasons of infrequency and traceability [2]. IP hopping technologies were firstly proposed and developed in legacy network. They have been tested to effective measures to contain different kinds of scanning attacks and other attacks based on hit list such as worms. Krylov and Kravtsov [4] proposed an IP hopping method which is deployable and

effective to hide content and destination server. Zhao et al. [5] analyzed shortcomings of service migration and proposed new technique called “Middle Agent” which was applied to the end-hopping prototype system. Clark et al. [6] showed via analysis that refreshing and reassigning the IP addresses would disrupt the normal communication, so they introduced an optimal method to minimize the disruption. Zhuang et al. [7] defined some key concepts that might be used to formally talk about MTD system; they also discussed some essential problems of the MTD system. Yackoski et al. [8] introduced a new network architecture called Self-shielding Dynamic Network Architecture (SDNA) allowing multiple types of dynamics. Cho et al. [9] concluded the design principles, key methodologies, important algorithms, and some relevant key techniques about MTD. Dunlop et al. [10] proposed a hopping system based on IPv6 address called MT6D to handle the limit of number of IP addresses while ensuring the acceptable speed of UDP requests. Morrell et al. [11] explored the ratio of clients and server of MT6D and discovered some network limits. Miao et al. [12] pointed out that IP hopping defense in SDN is inefficient because of flow table matching which introduces high latency. They proposed the use of vector packet processing to accelerate IP hopping defense in legacy network. Experiment results showed that they effectively reduced the hit rate under scanning attack while maintaining the data processing capability.

In SDN environment, Jafarian et al. [13] firstly proposed and deployed IP hopping in SDN; they proposed a MTD method named Random Host Mutation (RHM) which mutates IP addresses on the switches and keeps the mutation process transparent to the users (hosts). However, the hopping caused a certain amount of system overhead and cannot adapt to the behavior of the attacker. It is easy for attackers to find out the regular pattern of hopping defense. Jafarian et al. [14] then proposed a spatiotemporal address mutation that binds the changed IP address to the host. The source identity further improves the security of information system. However, the defense system requires sufficient address space, which is an additional overhead of defense. Krylov et al. [15] proposed a method countermeasure of DDoS attack called IP Fast Hopping, which is an implementable network layer software solution. This method also lacks the necessary perception of attack behavior and cannot adapt to the change of different attack behavior. On the basis of paper [13], Jafarian et al. [2] proposed an effective address randomization method which improves the unpredictability by fast mutation and constrained configuration. They proposed to use two-level hopping scheme to improve the efficiency of defense and further enhance the uncertainty of hopping. A flaw is that its attack perception ability is not sensitive enough, and it cannot adapt to the variety of attacks, such as irregular scanning frequency. Jafarian et al. [16] proposed an adversary-aware IP address randomization, which uses hypothesis testing to character behavior of attacker. Experiment shows a method in [16] which significantly slows down attack and increases its detectability. However, the accuracy of hypothesis test method depends on the number of known samples, which reduces the

efficiency of attack perception. MacFarland and Shue proposed a new hopping method in SDN which provides protection for information system without any modification on clients [17]. This transparent defense method is similar to [13] in effect and cannot adapt to the changes of attacker behavior. In [18], Chang et al. not only randomize the IP address to achieve the purpose of defense, but also solve the problem of IP address synchronization between network nodes. This scheme is another implementation method in SDN. However, its synchronization behavior inevitably results in additional system overhead. In [19], Lei et al. deployed adversary strategy awareness module in MTD system and proposed a novel technique called self-adaptive end-point hopping technique (SEHT). In SEHT, IP hopping is triggered and configured with the guide of adversary strategy awareness. The result of three-level hopping is a more refined method, which improves the unpredictability of defense. However, similar to [16], its perception method is not sensitive to the change of aggressive behavior. In [20], Smith et al. also introduce intrusion detection to trigger MTD system. In their work, the intrusion detection is based on neuroevolution of augmented topologies algorithm (NEAT) and is real time in operation. However, the accuracy of detection still needs to be improved. In [21], in order to maximize unpredictability of network mutation, Zhang et al. use adversary strategy awareness with hypothesis test to make mutation strategy selection. Similar to [16], the accuracy of hypothesis testing depends on the number of valid samples, which is an additional overhead. In [22], Ma et al. use anomalous awareness in [32] to drive a self-adaptive end-point hopping defense. However, the anomalous awareness based on information distance is not accurate enough. A smart attacker may be scanning without anomalousness on information distance.

In conclusion, IP hopping for MTD is an effective method that can defend against network attack especially scanning attack. However, some existing methods do have to be improved in their adaptability, which means the MTD system should respond to attack behavior accurately in real time and make fine-grained adjustment on hopping strategy.

3. Motivation

3.1. Attack Analysis. In this paper, we mainly focus on one kind of cooperative scanning attack. This cooperative scanning indicates that a number of hosts act as scanners and sample (probe) IP addresses in the protected network. The whole IP space will be divided and assigned to the scanners and will be scanned uniformly. The purpose of attacker is testing which IP addresses are currently used online, so that he can prepare for following attacks such as DDoS. A cautious attacker usually scans IP addresses without repetition to minimize failed probes [33].

We assume that the scanning attack starts from the outer network but is a propagating scanning. Each newly probed host may get infected and start acting as a scanner. Once a host is probed, it takes time t_p to infect the host. Scanners can share the IP address space that has not been scanned.

Each scanner has two important attributes which are frequency of scanning packet (FSP) and proportion of scanning packet (PSP). FSP denotes the number of probes sent per second. PSP denotes the proportion of probe packets in all packets sent by the scanner.

We believe that, compared with network data flow produced by normal host for legal communication, data flow produced for scanning attack does have distinguishing features. These features make it possible to detect scanning attack from network data flow without being disturbed by normal probes. "Normal probes" means probes released for legal communication.

3.2. Promotion Idea. In adaptive IP hopping system, the result of attack awareness is used to guide the configuration of IP hopping and trigger IP hopping. Both the accuracy and timeliness of attack awareness directly affect the validity of hopping. The accuracy demands not only detecting malicious packet (false rejection rate) but also not being disturbed by normal probes (false acceptance rate).

However, the accuracy and timeliness of previous adaptive IP hopping methods still need to be improved. The method in [20] uses a nearly fully connected structure to build a light-weight detection network. The structure of full connection makes it difficult to balance the capacity and lightweight of the network, so it is difficult for the detection to achieve balance between accuracy and timeliness. The method in [19] simply judges failed probe packets as malicious. However, not failed probe packets can also be malicious ones, and failed probe packets may also be legal ones. The method of data sampling in [19] also needs to be improved.

Regarding accurately aware behavior of attack in real time, in this paper, we firstly propose a new method of data sampling. The proposed sampling method collected all probe packets in order to include all malicious scanning packets. A part of nonprobe packets that may be related to scanning attack are also included. We then propose a novel CNN structure to detect sampled data and drive IP hopping system including guiding hopping configuration and triggering hopping. We try to make hopping defense react each illegal probe to improve effectiveness of hop and reduce invalid hops. Architecture of CNN driving adaptive IP hopping is shown in Figure 2. The reason for choosing CNN as the detector is as follows: firstly CNN's local perception processing can mine the data correlation and remove the redundant information in data so as to ensure the detection accuracy; secondly, CNN's weight sharing and pooling process can reduce the number of connections and weights in neural network so as to build a light-weight detector. Thirdly, CNN has been used in intrusion detection system and proved an effective tool to detect behavior of attackers [34–37]; thus, previous works can be used for reference in this paper.

4. The Proposed Method

4.1. Architecture and Workflow. The proposed IP hopping system evades and prevents scanning attacks by dynamically changing IP address of protected hosts, thus increasing the usage difficulty of vulnerabilities and backdoors and

ensuring the security of protected network. The adaptability of proposed method is reflected in two aspects: adaptive IP hopping configuration and adaptive IP hopping trigger. The CNN detector is used to make the attack behavior aware automatically and to not only guide hopping configuration but also trigger hopping.

The flowchart of our proposed method is composed of 4 modules as shown in Figure 3 which are packets sampling, data preprocess, trained CNN detector, and hopping strategy execution. In packets sampling module, we collect packets which may be related to scanning attack. Data preprocess module is used to form the collected packets into a data matrix as the input of the following CNN detector. The CNN detector is a designed module which is used to judge whether the input data correspond to a malicious host or not. Finally, IP hopping strategy is activated by the judgment result of CNN detector. Besides, the hopping can also be triggered by a preset timer in case that the CNN detector makes missed alert.

4.2. Packets Sampling. Input data of CNN should include packets of normal probe or scanning attack and their related packets. In order to ensure the accuracy of detection, the input data should be as complete as possible but as little redundant as possible. Considering that the scanning behavior of attackers is variable, for example, attacker can launch scanning attack packets in different FSPs and PSPs. Smart attacker can also hide his scanning attack packets among background data stream. Considering that the attack data traffic may have the above characteristics, two different sampling methods are used at the same time, which are continuous sampling and precise sampling. Continuous sampling is used to sample not only probe packet itself but also the after background stream packet, so as to make it possible for the subsequent processing to mine the relationship between the attack packet and the background stream. Precise sampling is used to collect all probe packets since each probe packet could be an attack packet. Precise sampling collects only probe packets. Both precise sampling and continuous sampling are touched off by any of the eight kinds of packets which could be used to probe including ARP request, ICMP echo, ICMP time stamp, ICMP net-mask, TCP SYN, TCP ACK, TCP FIN, and UDP empty.

4.2.1. Continuous Sampling. For every probe packet, continuous sampling collects some following packets from the same source host. The following packets and probe packet itself form the sampled packets of one continuous sampling.

Considering different packets from the same source that host may reach and that are forwarded by different switches, once any switch receives a probe packet, it packets in the probe packet to the controller and raises an event (it is called "event" in RYU platform) of continuous sampling. After that, it is ordered by the controller that every switch in the controlled network should packet in a certain number of following packets sent by the source host of the probe packet in a specified time.

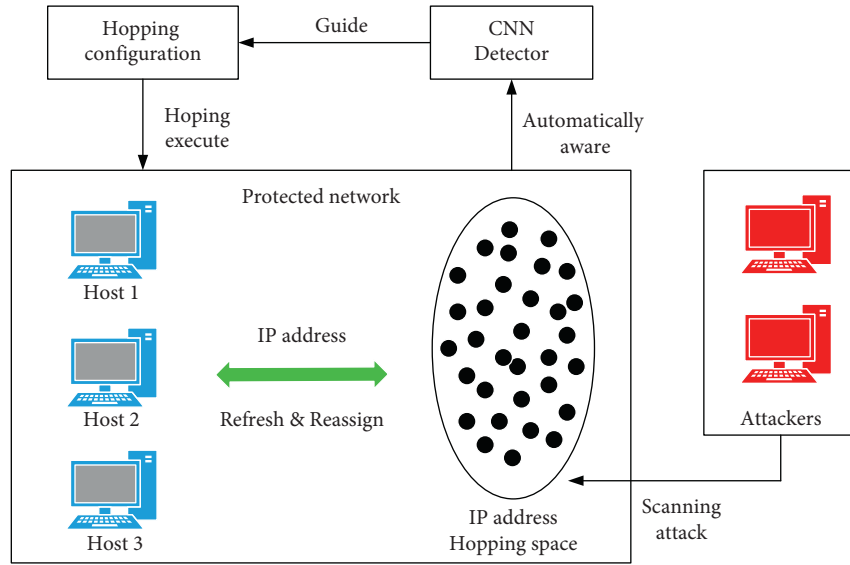


FIGURE 2: Architecture of CNN driving adaptive IP hopping.

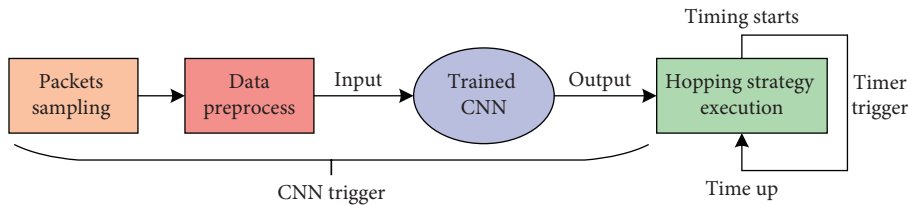


FIGURE 3: The flowchart of the proposed IP hopping system.

4.2.2. *Precise Sampling.* Precise sampling only pays attention to probe packets. For every probe packet, precise sampling collects some previous probe packets and some following probe packets from the same source. The previous probe packets, following probe packets, and probe packet itself form the sampled packets of one precise sampling.

Implementation of precise sampling is similar to continuous sampling, but precise sampling only collects probe packets. Once any switch receives a probe packet, it packets in the probe packet to the controller and raises an event of precise sampling. After that, it is ordered by the controller that every switch in the controlled network should packet in a certain number of following probe packets from the same source in a specified time. The controller has already stored all previous probe packets in the network, and it selects a certain number of previous probe packets from the same source.

4.3. *Data Preprocessing.* Our designed CNN detectors will operate on data extracted from sampled packets in order to make a classification whether the packet and its source host are illegal or not. Data extracted from one continuous sampling and one precise sampling combine an input data of CNN detector. The method of data extraction is as follows.

Each packet is essentially a binary bitstream. In order to further remove invalid parts from sampled data, only parts that may be related to scanning attack are preserved, while

others are discarded. Five parts of each packet (if exist) are preserved, which are

- (1) Fixed part of the IP datagram header (160 bits)
- (2) Fixed part of the TCP message header (160 bits)
- (3) UDP message header (64 bits)
- (4) ICMP message header (64 bits)
- (5) Ethernet header and ethernet ARP fields of ARP message (42 bits)

Content of the packet that does not belong to the above five parts will be discarded. We also notice that since IP hopping system makes the protected network ever-changing, the judgment whether one probe packet is illegal or not should consider the current network situation. For this purpose, we add a flag bit at the end of the binary bitstream extracted from each probe packet, as shown in Figure 4. If the probe packet hits a currently used IP address, the flag bit is set 1; otherwise it is set 0.

We do not convert these binary bitstreams to integers as [20] does for the purpose of avoiding data confusion and preserving data integrity. Data extracted from each sampled packet were placed end-to-end forming vector F . In order to adapt to convolutional process in CNN detector, vector F is then converted to matrix M . Zero padding is used to ensure a two-dimensional matrix.

Since the number of sampled packets may probably be different in each sampling, and the types of packets obtained

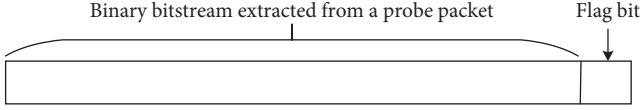


FIGURE 4: Binary bitstream extracted from a probe packet and the added flag bit.

from each sampling are also different, size of M is not fixed. To adapt different size of input data, we introduce global pooling to be the last pooling layer in the structure of CNN, which is described in the next subsection.

4.4. Designed CNN Detector. CNN has been widely used in network intrusion detection such as [34–37]. Meanwhile, because of its local perceptual processing mode, CNN has special advantage in discovering the local to local correlation in data. As analysis in packets sampling and data process, the local to local correlation in our data may probably be related to the classification (malicious or not). For the structural design of CNN, we have kept adjusting the structure of CNN empirically in a large number of experiments. The adjustment process includes using different activation functions and using convolution kernels of different sizes to make sure the structure is suitable for the judgment on our processed attack traffic data.

The CNN detector is designed to be deployed on the controller of SDN. We set two requirements of the deployment of the designed CNN: firstly, there should be no special requirements for hardware performance of the controller; secondly, time consumed in one execution of judgment should not be too long to avoid data piling up. The CNN should be a light-weight detector. The structure of the designed CNN is shown in Figure 5, which is composed of three convolutional modules and a judgment module. Each convolutional module is composed of a convolutional layer, a batch normalization layer [38], a ReLU layer [39], and a pooling layer. The reason for using ReLU as activation is that our input data is a binary matrix. ReLU makes different activation for the element greater than 0 and less than 0. The judgment module is composed of a fully connected layer and a Softmax layer.

In Figure 5, the boxes with “Conv” mean convolutional layer. The formulas $a \times (b \times c \times d)$ in “Conv” boxes show the size and shape of convolutional kernel. A convolutional layer inputs d matrixes and outputs a matrixes, and $b \times c$ is the size of convolutional kernel on each input matrix. The process of a convolutional layer is shown in

$$X_j^{\text{out}} = \sum_{i=1}^d X_i^{\text{in}} * K_{ij} + B_j, \quad (1)$$

where X_j^{out} is the j th output matrix of the layer, X_i^{in} is the i th input matrix of the layer, K_{ij} is a convolutional kernel operated on X_i^{in} and output to X_j^{out} , and B_j is the bias matrix added to the j th output matrix. In (1), $*$ means convolution operation.

The boxes in Figure 5 with “Pooling” mean pooling layer. In the boxes of pooling layer, “size” means the size of pooling

window and “s” means the stride of pooling operation. The process of pooling layer can be described in

$$x = \text{pool}_{\text{ave}}(R) = \frac{1}{|R|} \sum_{r \in R} r, \quad (2)$$

where x is an output neuron, ave means we use average pooling in each pooling layer, R means a pooling region in the input matrix, and r is a neuron in R . Global pooling is operated in the last pooling layer, which means the size of pooling window flexibly is equal to the size of each input matrix of the layer. Global pooling enables the CNN to adapt to different size of input data. The process of convolution and pooling is shown in Figure 6.

The boxes in Figure 5 with “ReLU” mean ReLU layer which is an activation layer. These layers improve the nonlinear factors of CNN and enable the model to fit a problem that linear model cannot. The process of ReLU layer is described in

$$x^{\text{out}} = \max(0, x^{\text{in}}), \quad (3)$$

where x^{out} is an output neuron and x^{in} is an input neuron of ReLU layer.

The boxes in Figure 5 with “BN” mean batch normalization layer. BN layer first normalizes elements in each input feature map to zero-mean and unit-variance to ensure that the input neuron to ReLU falls in the region near value 0, and hence the gradient back-propagation would not fall into poor local minima.

The box in Figure 5 with “Fully connected” means fully connected layer. In this layer, each output neuron is connected to each input neuron with a weight parameter. The process of this layer is shown in

$$x_j^{\text{out}} = \sum_{i=1}^{N_f} x_i^{\text{in}} \times \omega_{ij}, \quad (4)$$

where x_j^{out} means the j th output neuron of this layer, x_i^{in} means the i th input neuron of this layer, N_f means the number of input neurons of this layer, and ω_{ij} is the weight parameter of connection between x_j^{out} and x_i^{in} .

The box in Figure 5 with “Softmax” means Softmax layer. This layer normalizes the value of neuron to the range of [0, 1] to indicate probabilities of the input data belonging to each class. The process of Softmax layer is shown in

$$x_j^{\text{out}} = \frac{e^{x_j^{\text{in}}}}{\sum_{i=1}^{N_s} e^{x_i^{\text{in}}}}, \quad x_j^{\text{out}} \in [0, 1], \quad (5)$$

where x_j^{out} means the j th output neuron of this layer, $e^{x_j^{\text{in}}}$ and $e^{x_i^{\text{in}}}$ mean j th and i th input neurons of this layer, and N_s means the number of input neurons and is also the number of output neurons of this layer. In our designed CNN, both fully connected layer and Softmax layer output two neurons; one indicates “normal” and the other indicates “malice.”

Denote $D(\cdot)$ as the process of the trained CNN detector and X as a matrix of input data. Judgment of the CNN detector is shown in (6) where result “1” means “malice” and result “0” means “normal.”

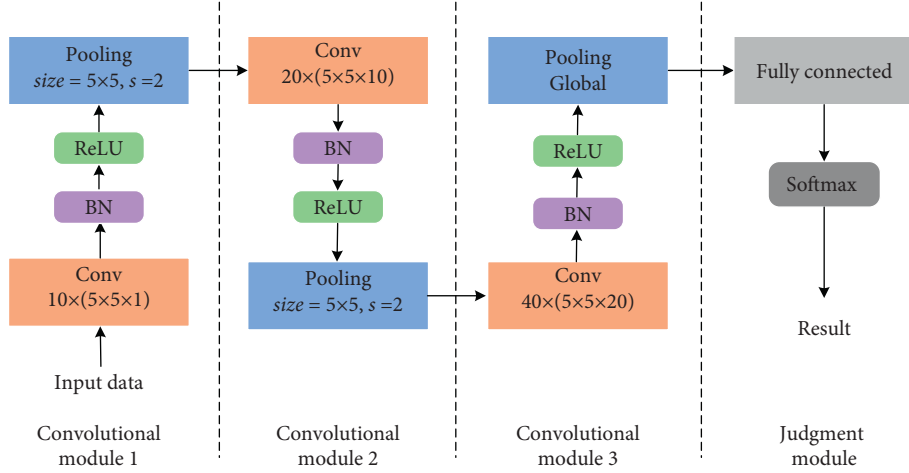


FIGURE 5: Structure of designed CNN.

$$D(X) = \begin{cases} 0, & \text{normal,} \\ 1, & \text{malice.} \end{cases} \quad (6)$$

It is obvious that packets sampling occupies the channel between the switch and the controller to some extent. Also, data process and CNN detection consumes the resources on the controller side. In order to mitigate the impact on network QoS, one judgment result will also be applicable for other probe packets from the same source in 1 minute. In the following 1 minute, probe packets from the same source will not raise packets sampling. This also prevents the defense system from being overburdened by an excessive number of probe packets.

4.5. IP Hopping Strategy. In this paper, we provide each protected host with a fixed real IP (rIP) and a hopping virtual IP (vIP). In our IP hopping system, IP hopping can be touched off by two kinds of events: the first is the system timer (periodic hopping) and the second is the judgment result of the CNN detector (triggered hopping). Periodic hopping and triggered hopping run independently. The set of periodic hopping is in case that CNN makes missed alert and triggered hopping was not executed in some situation.

The notions used in our IP hopping strategy are listed in Table 1. Assume the whole protected network has k subnets which are represented by $\{S^1, S^2, \dots, S^k\}$. Total l protected hosts are distributed in the k subnets. The l protected hosts are represented by $\{h^1, h^2, \dots, h^l\}$. IP_W , IP_R , and IP_V represent the whole IP set, the real IP set, and the virtual IP set, respectively, so $IP_W = IP_R \vee IP_V$, $rIP \in IP_R$, and $vIP \in IP_V$. It is impossible to choose vIP from the whole IP_V in each hopping, because a range of vIPs can only be assigned to one physical subnet at a given time. We adopt a three-level hopping to assign available vIP space, which are base hopping, low-frequency hopping, and high-frequency hopping. As shown in (7), the IP_V set is divided into m_B number of base hopping range (BHR) according to the number and scale of subnets, $m_B \geq k$.

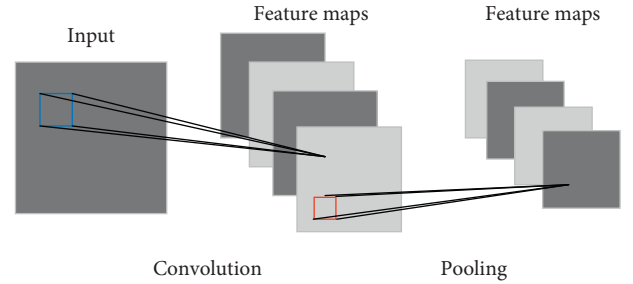


FIGURE 6: Convolution and pooling in convolutional neural network.

$$IP_V = \sum_{b=1}^{m_B} R_{BHR}^b. \quad (7)$$

Then, as shown in (8), each BHR is divided into m_L number of low-frequency hopping range (LHR) according to the number of hosts in subnet. Each LHR contains several vIPs, and these vIPs are used for high-frequency hopping.

$$R_{BHR}^b = \sum_{l=1}^{m_L} R_{LHR}^l. \quad (8)$$

4.5.1. Periodic Hopping. The IP hopping system periodically refers to three levels of hopping which are base hopping, low-frequency hopping, and high-frequency hopping. Base hopping changes the IP address range assigned to a given subnet with time interval T_{BH} . Low-frequency hopping changes the IP address ranges assigned to a given host within the corresponding BHR with time interval T_{LH} . High-frequency hopping changes the vIP assigned to a given host within the corresponding LHR with time interval T_{HH} .

Adaptive IP hopping configuration is an important aspect of adaptability in proposed method. To characterize attacker's scanning behavior, we associate a weight ω_B with each BHR, a weight ω_L with each LHR, and a weight ω_H with each protected host. Higher weight for a range indicates

TABLE 1: The notions used in the IP hopping strategy.

Character	Description
rIP, vIP	Real IP address and virtual IP address
S^i, h^i	The i th subnet and the i th host in the protected network
IP_W, IP_R, IP_V	The sets of all IP addresses, real IP addresses, and virtual IP addresses
R_{BHR}^b, R_{LHR}^l	The b th BHR and the l th LHR in a BHR
m_B, m_L	The number of BHR and the number of LHR in a BHR
T_{BH}, T_{LH}, T_{HH}	Time interval of base hopping, low-frequency hopping, and high-frequency hopping
$\omega_B, \omega_L, \omega_H$	Weights of each BHR, LHR, and protected host to characterize attacker's scanning behavior
r_{ij}^t	A flag to show whether R_{BHR}^i is allocated to S^j in t th BHR assignment scheme
P_{BH}, P_{LH}	Thresholds to touch off base hopping and low-frequency hopping
HT_B^i, HT_L^i	Hit times of the vIP pool of S^i and h_i
N^{S^i}	The number of hosts in S^i

higher attacker activity in that range in the last interval. The value of ω_B , ω_L , and ω_H is computed according to the malicious probe packets in the last interval. These weights will guide the configuration of next hopping.

(1) *Base Hopping*. In base hopping, each subnet will be assigned with one or more BHRs. Denote ω_B^i as the weight of the R_{BHR}^i and N_{BHR}^i as the number of vIPs in it. Computation of weight ω_B^i is in Algorithm 1.

When more than one BHR assignment schemes is available, we choose the scheme that achieves minimal standard deviation of sums of weights of BHRs assigned to a subnet, in order to ensure nearly equal security of each subnet. Denote r_{ij}^t as a flag. $r_{ij}^t = 1$ means R_{BHR}^i is allocated to S^j in the t th assignment scheme, and $r_{ij}^t = 0$ means R_{BHR}^i is not allocated to S^j in the t th assignment scheme. In the t th assignment scheme, we compute the standard deviation of sums of weights of BHRs assigned to S^i as (9)–(13). Vector W denotes the set of ω_B ; vector R_i^t denotes the set of flag r_{ij}^t ; s_i^t denotes the sum of weights of BHRs assigned to S^i in the t th BHR assignment scheme. μ^t denotes the mean, and σ^t denotes the standard deviation.

$$W = (\omega_B^1, \omega_B^2, \dots, \omega_B^{m_B}), \quad (9)$$

$$R_i^t = (r_{i1}^t, r_{i2}^t, \dots, r_{im_B}^t), \quad (10)$$

$$s_i^t = R_i^t \cdot W^T, \quad (11)$$

$$\mu^t = \frac{1}{m_B} \sum_{i=1}^{m_B} s_i^t, \quad (12)$$

$$\sigma^t = \sqrt{\frac{1}{m_B} \sum_{i=1}^{m_B} (s_i^t - \mu^t)^2}. \quad (13)$$

We compare σ^t in each assignment scheme and select the assignment scheme with minimal σ^t for next base hopping.

(2) *Low-Frequency Hopping*. Once a subnet is assigned with one or more BHRs, these BHRs are divided into several LHR. For low-frequency hopping, each host will be assigned with one or more LHRs. Each LHR has a weight ω_L . The

computing method of ω_L and the selection of LHR assignment scheme are the same as those of base hopping, so they are not elaborated here.

(3) *High-Frequency Hopping*. Once a host is assigned with one or more LHRs, vIPs in these LHRs form a vIP pool. In high-frequency hopping, a vIP in the pool will be selected and used in next interval. For high-frequency hopping, denote ω_H^i as the weight of the i th vIP in the pool of a host. Computation of weight ω_H^i is in Algorithm 2.

For one host, only one vIP can be used at one time. All vIPs in the pool are sorted according to their weights. The vIPs with higher weights will be preferred.

4.5.2. *Triggered Hopping*. In our IP hopping system, the hopping can also be touched off by the judgment result of the CNN detector. Base hopping, low-frequency hopping, and high-frequency hopping will be active in three cases, respectively.

- (1) *Base Hopping*. As in Algorithm 3, in a time interval T_{BH} , if the number of hit times of the vIP pool of a subnet divided by the number of host in the subnet has reached the threshold P_{BH} , a base hopping for the whole protected network should be touched off. Denote HT_B^i as the number of hit times of the vIP pool of S^i .
- (2) *Low-Frequency Hopping*. As in Algorithm 4, in a time interval T_{LH} , if the number of hit times of the vIP pool of host h^i has reached the threshold P_{LH} , a low-frequency hopping for the corresponding subnet should be touched off. Denote HT_L^i as the number of hit times of the vIP pool of h^i .
- (3) *High-Frequency Hopping*. As in Algorithm 5, in a time interval T_{HH} , if a host is hit by a malicious probe packet, a high-frequency hopping for the host should be touched off.

The proposed periodic hopping and triggered hopping are both adaptive hopping strategies. On the one hand, they can make protected hosts avoid using IP addresses that attackers may seriously scan, so as to improve the survival rate protected hosts under scanning attack. On the other hand, once a protected host is hit in scanning attack, its used


```

Require: All detected data  $X$  in last base hopping interval
For all probe packets in the last base hopping interval
  If  $D(X) == 1$  && the target IP is in  $R_{\text{BHR}}^i$ 
     $\omega_B^i ++$ ;
  End if
End for
 $\omega_B^i = \omega_B^i / N_{\text{BHR}}^i$ ;
Return  $\omega_B^i$ 

```

ALGORITHM 1: Computation of weight ω_B^i in base hopping in periodic hopping.

```

Require: All detected data  $X$  in last high-frequency hopping interval
Require: vIP pool of the host for high-frequency hopping
For all probe packets in the last high-frequency hopping interval
  If  $D(X) == 1$  && the target IP is the  $i$ th vIP in the pool
     $\omega_H^i ++$ ;
  End if
End for
Return  $\omega_H^i$ 

```

ALGORITHM 2: Computation of weight ω_H^i in high-frequency hopping in periodic hopping.

```

Require: All detected data  $X$  after last base hopping
Require: vIP pool of  $S^i$ 
Require: Preset threshold  $P_{\text{BH}}$ 
 $\text{HT}_B^i = 0$ ;
For all probe packets
  If  $D(X) == 1$  && the target IP is in the vIP pool of  $S^i$ 
     $\text{HT}_B^i ++$ ;
  End if
End for
If  $\text{HT}_B^i / N^S \geq P_{\text{BH}}$ 
  Triggering base hopping;
End if
Return Null

```

ALGORITHM 3: Base hopping in triggered hopping.

```

Require: All detected data  $X$  after last low-frequency hopping
Require: vIP pool of  $h^i$ 
Require: Preset threshold  $P_{\text{LH}}$ 
 $\text{HT}_L^i = 0$ ;
For all probe packets
  If  $D(X) == 1$  && the target IP is in the vIP pool of  $h^i$ 
     $\text{HT}_L^i ++$ ;
  End if
End for
If  $\text{HT}_L^i \geq P_{\text{LH}}$ 
  Triggering low-frequency hopping for the corresponding subnet;
End if
Return Null

```

ALGORITHM 4: Low-frequency hopping in triggered hopping.

```

Require: All detected data  $X$  after last high-frequency hopping
Require: Currently used IP of  $h^i$ 
If  $D(X) == 1$  && the target IP is at use by  $h^i$ 
    Triggering high-frequency hopping for  $h^i$ ;
End if
Return Null

```

ALGORITHM 5: High-frequency hopping in triggered hopping.

IP address may immediately hop to another one to disable the scanned target in attacker’s hit list.

5. Implementation and Evaluation

In this section, we firstly introduce the hardware and software platform of our implementation. Then the used network topology and the behavior of attackers are briefly described. Finally we propose three performance indexes to experiment and evaluate the performance of our proposed method.

5.1. Hardware and Software Platform. To investigate the effectiveness and scalability of proposed approach, we implemented it on an OpenFlow controller [26] that manages a Mininet network [40]. We used Mininet to create a network of OpenFlow switches (Open vSwitch kernel switches). Ryu platform is used on the controller to deploy the application of proposed IP hopping. We used Tensorflow 1.0 [41] to train and save the CNN model, and the trained model is called from Ryu application to enforce a judgment. The CNN model is trained in GPU mode with a Nvidia K40c, but it is used in CPU mode with Intel Core i7 8700 for the purpose of no high hardware requirements of controller.

We use Mininet to create a virtual network which is managed by a remote controller. The virtual network is shown as Figure 7. The created virtual network contains 2^{10} hosts. These hosts are distributed in 2^5 subnets. The number of hosts in each subnet is random. The subnets are connected with each other with OpenFlow switches. Wireshark is used in our Mininet to collect and form data set of probe packets.

To show the effectiveness of the proposed method against hit list attacks, cooperative scanning attack is launched on some host and then may propagate to others. We assume that the scanners are aware of the IP resource pool of the protected network.

5.2. Performance Evaluation. Three performance indexes are used to experiment and evaluate the performance of our proposed method which are accuracy and performance of CNN detection, survival rate of protected hosts, and the average lifetime of targets in attack’s hit list. Technologies for performance comparison with our proposed method are OF-RHM [13], SEHT [19], and the method in [12]. In order to ensure fairness, OF-RHM and method in [12] are adjusted to three-level hopping which is consistent with SEHT and proposed method.

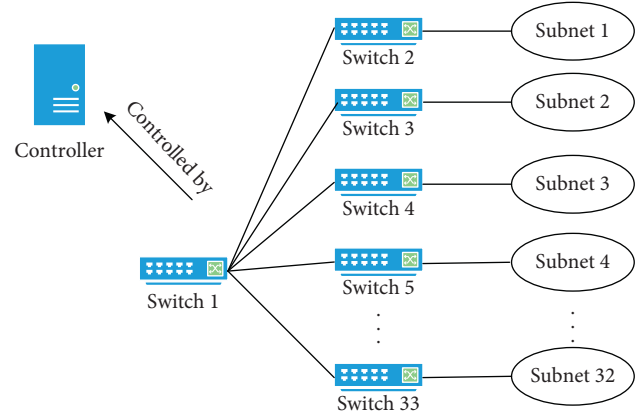


FIGURE 7: The topology of Mininet virtual network.

5.2.1. Performance of CNN Detection. Accuracy and time efficiency are important indexes of CNN and determine the effectiveness of adaptability of our proposed method. Data extracted from one continuous sampling and one precise sampling combine a sample. We totally collected 10000 samples and these samples make up our data set. The data set is divided into three sets, which are training set, validation set, and testing set. The training set consists of 7000 samples and is used in training the CNN model; validation set consists of 1000 samples and is used to monitor the real time accuracy performance of CNN in the process of training; testing set consists of 2000 samples and is used to finally test the accuracy and time efficiency of CNN detection. Our data set is composed of samples collected from scanners with PSP 20%, 40%, 60%, and 80% and FSP 0.5 s, 1 s, and 2 s. The training phase takes 7.29 hours on our platform.

In order to ensure real time of detection, if the amount of data in the queue to be detected reaches three, undetected data except the new arrival one will be discarded (data loss).

Denote Acc as the accuracy of CNN detection, FRR as the false rejection rate, and FAR as the false acceptance rate. Testing performance is shown in Table 2. The proposed CNN performs better under higher PSP.

Denote TE as time efficiency, which means time consumed in one test including the time of data preprocess and CNN detection. Although the test of CNN is set in CPU mode, TE is only 0.019377 s. Another adaptive hopping method SEHT [19] directly treats all failed probe packets as scanning attacks, so it takes nearly no time to discover scanning attack. However, not distinguishing between scanning attack and normal probe in SEHT makes the result

TABLE 2: Testing performance of the CNN detection.

PSP (%)	Acc	FRR	FAR
20	0.7604	0.2113	0.2679
40	0.8215	0.1357	0.2213
60	0.8483	0.0928	0.2107
80	0.9667	0.0179	0.0487

inaccurate. Moreover, SEHT adopts hypothesis tests based on Sibson entropy to discriminate scanning attack strategy. We obtain that TE of hypothesis test in SEHT is about 0.015377 s, which is only slightly better than our proposed CNN. CNN could be retrained through transfer learning to acquire ability of multiclassification to discriminate scanning attack strategy while barely increasing TE.

For further research, we calculate the rate of data loss in different number of scanners under nonpropagating scanning. RDL denotes the rate of data loss. Table 3 shows RDLs in different number of scanners (NoS). Obviously, higher FSP and larger NoS cause higher RDL. The reason is that, with higher FSP and larger NoS, the CNN may probably find it more difficult to handle the coming packets in time.

5.2.2. Survival Rate of Protected Hosts under Cooperative Scanning Attack. The attacker carries out uniform and unrepeatable scan on the IP space used in protected network. Survival rate of protected hosts denotes the proportion of hosts which are not probed by attacker through a round of attack.

(1) Survival Rate under Different Initial Scanners. The survival rates of protected hosts under propagating cooperative scanning attack using different quantity of initial scanners are shown in Figures 8–11. Figure 8 shows survival rate under cooperative worm using 50 scanners; Figure 9 shows survival rate under cooperative worm using 100 scanners; Figure 10 shows survival rate under cooperative worm using 150 scanners; Figure 11 shows survival rate under cooperative worm using 200 scanners. The hyperparameters are set as follows: $T_{HH} = 20$ s, $T_{LH} = 40$ s, $T_{BH} = 160$ s, PSP = 60%, FSP = 2 s, $P_{BH} = 0.5$. P_{LH} is equal to half the number of IPs in the IP pool of corresponding host.

In a static network, the survival rate reaches 0 after spending 101 s–247 s because of using uniform active scanning. Method of OF-RHM can lower the reduction of survival rate and finally improve the minimum survival rate to some extent. The two adaptive methods can significantly lower the reduction of survival rate and finally improve the minimum survival rate. Moreover, performance of our proposed method is slightly better than SEHT. Meanwhile, the defense performance of method in [12] is similar to that of OF-RHM. The proposed method achieves the best result among the five methods, because it can accurately identify scanning attack packets and guide IP hopping of hosts to avoid attack.

(2) Minimal Survival Rate under Different Time Intervals. Minimal survival rate denotes the proportion of hosts which are not probed after a round of scanning attack. Table 4 shows ten different time intervals used in our experiments.

TABLE 3: Rates of data loss (RDL) in different number of scanners.

FSP (s)	NoS			
	50	100	150	200
0.5	0	0.3512	0.4019	0.5467
1	0.3421	0.5381	0.6011	0.6604
2	0.5156	0.7402	0.8351	0.8603

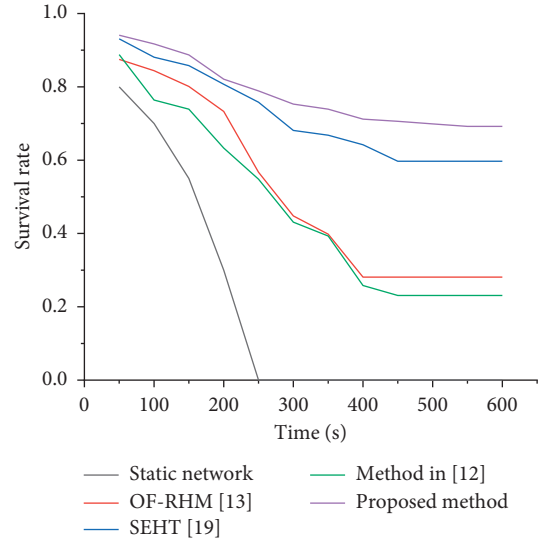


FIGURE 8: Survival rate under cooperative scanning (50 scanners).

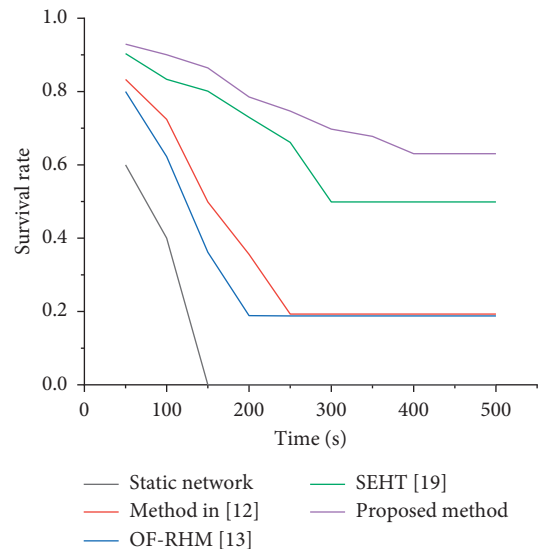


FIGURE 9: Survival rate under cooperative scanning (100 scanners).

The minimum survival rates under cooperative worm on IP hopping systems with different time intervals are shown in Figures 12–15. Figure 12 shows the minimum survival rate using 50 scanners; Figure 13 shows the minimum survival rate using 100 scanners; Figure 14 shows the minimum survival rate using 150 scanners; Figure 15 shows the minimum survival rate using 200 scanners. The hyperparameters are set as follows: PSP = 60%, FSP = 2 s,

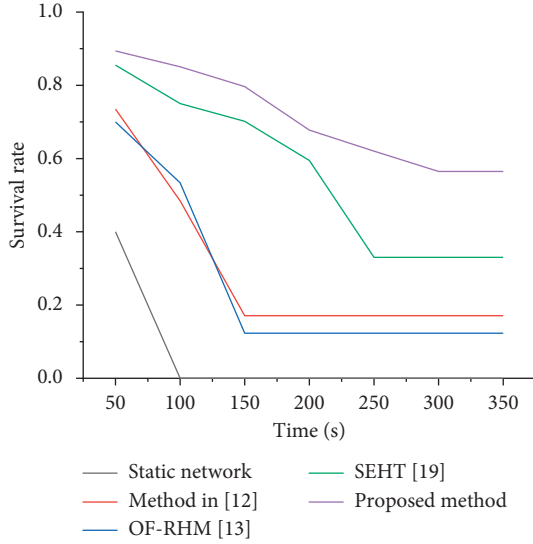


FIGURE 10: Survival rate under cooperative scanning (150 scanners).

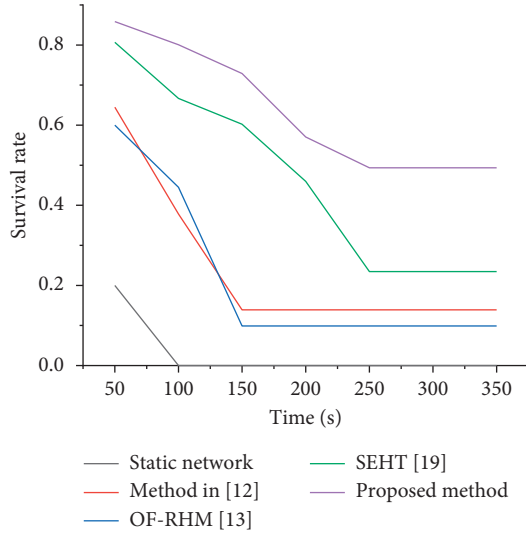


FIGURE 11: Survival rate under cooperative scanning (200 scanners).

TABLE 4: P1–P10: different settings of three-level time interval (s).

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
T_{HH}	5	10	15	20	25	30	35	40	45	50
T_{LH}	10	20	30	40	50	60	70	80	90	100
T_{BH}	40	80	120	160	200	240	280	320	360	400

$P_{BH} = 0.5$. P_{LH} is equal to half the number of IPs in the IP pool of corresponding host.

The minimum survival rates under cooperative worm in network with OF-RHM decrease from P1 to P10 and finally reach 0 under P6 to P10. In network with OF-RHM, long time interval makes the network equivalent to a static one. The minimum survival rates under cooperative worm in network with SEHT decrease slightly between P1 and P7 and then go stabilized. The minimum survival rates under

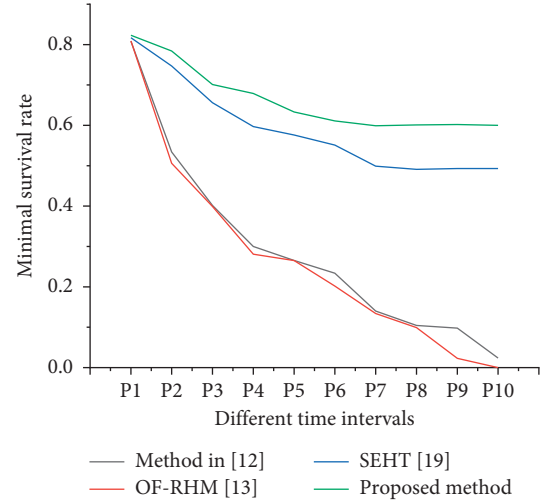


FIGURE 12: The minimal survival rates under cooperative scanning with different time intervals (50 scanners).

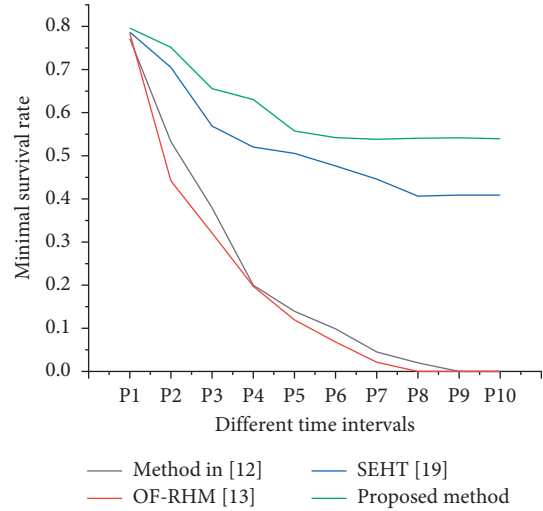


FIGURE 13: The minimal survival rates under cooperative scanning with different time intervals (100 scanners).

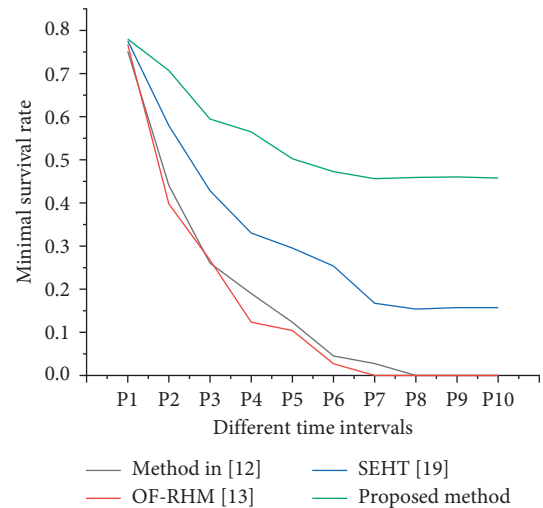


FIGURE 14: The minimal survival rates under cooperative scanning with different time intervals (150 scanners).

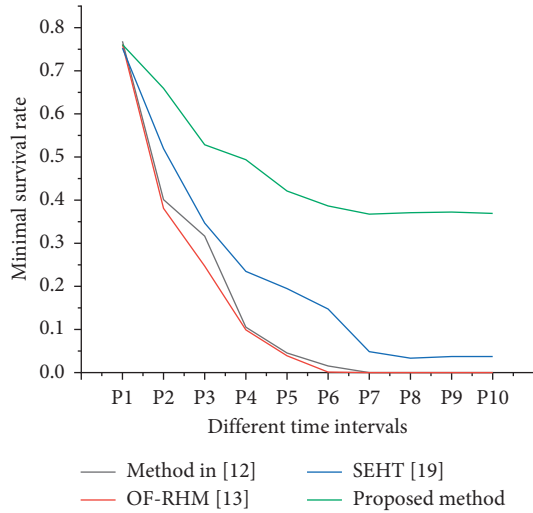


FIGURE 15: The minimal survival rates under cooperative scanning with different time intervals (200 scanners).

cooperative worm in network with the method in [12] are similar to that of OF-RHM. The minimum survival rates under cooperative worm in network with our proposed method decrease the most slightly in the three methods. Because scanning behavior of attacker triggers IP hopping with different levels, our proposed method gets the best performance in the three methods. Using our proposed method, defender can set greater time interval to reduce network resource overhead and the possibility of packet loss.

(3) *Minimal Survival Rate under Different PSP and FSP.* The value of PSP affects the performance of our designed CNN detector, and the value of PSP affects rates of data loss. The minimal survival rates under different PSP and FSP are shown in Table 5. The hyperparameters are set as follows: $T_{HH} = 20$ s, $T_{LH} = 40$ s, $T_{BH} = 160$ s, $P_{BH} = 0.5$, $NoS = 50$, and P_{LH} is equal to half the number of IPs in the IP pool of corresponding host.

As shown in Table 5, compared with SEHT, our proposed method performs better significantly when $PSP = 60\%$ and $PSP = 80\%$; when $PSP = 40\%$, our proposed method performs slightly better than SEHT; when $PSP = 20\%$, our proposed method performs slightly worse except for $FSP = 0.5$ s. With the decrease of PSP value, the CNN makes more false rejection which weakens the advantage of our proposed method gradually. Lower FSP indicates better performance both in the proposed method and in SEHT. Also notable is that performance of SEHT is not sensitive to change of PSP for the reason that SEHT treats all failed probe packets as scanning attack.

5.2.3. *Average Lifetime of Targets in Attack’s Hit List.* Worm-type virus and other hit list based network attacks need time long enough to launch. For example, propagation of worm from one host to another must take enough time. If VIP of target host hopped before the completeness of worm propagation, the worm propagation fails. We compute the average lifetime of targets in attack’s hit list. Reducing lifetime of targets in attack’s hit list is beneficial to reduce the

TABLE 5: Minimal survival rate under different PSP and FSP.

PSP (%)	FSP					
	Proposed method			SEHT [19]		
	0.5 (s)	1 (s)	2 (s)	0.5 (s)	1 (s)	2 (s)
20	0.699	0.557	0.451	0.693	0.655	0.579
40	0.704	0.677	0.594	0.690	0.661	0.595
60	0.799	0.732	0.679	0.701	0.667	0.597
80	0.833	0.767	0.769	0.699	0.672	0.601

success rate of worm propagation. Lifetime of a target is explained as the duration between receiving response of probe packet from the target host and hopping of IP address of the target host. Figure 16 takes host 1 as an example to explain the meaning of lifetime.

Figure 17 shows the average lifetime of targets using 50 scanners; Figure 18 shows the average lifetime of targets using 100 scanners; Figure 19 shows the average lifetime of targets using 150 scanners; Figure 20 shows the average lifetime of targets using 200 scanners. The hyperparameters are set as follows: $PSP = 60\%$, $FSP = 2$ s, $P_{BH} = 0.5$. P_{LH} is equal to half the number of IPs in the IP pool of corresponding host.

The average lifetime of targets in our proposed method is significantly lower than methods without CNN assistance. However, we also find that as the number of scanners increases, advantage of our proposed method is weakened gradually. That may be probably because of the increasing rate of data loss.

Moreover, we make a comparison of average lifetime between targets caused by discarded data and data not discarded. The result of the comparison is shown in Table 6. In Table 6, “NoS” means number of scanners. Once probe packets of attacker are detected, the lifetime of targets caused by these packets can be controlled in less than 1 second. The adaptability of our proposed method driven by CNN can obviously reduce the average lifetime of targets in attack’s hit list. Moreover, improve the efficiency of attack awareness and reduce data loss may probably be our future work.

5.2.4. *Hopping Frequency.* Once the IP address of a host hops, communication to the host will withstand a short time high delay for the reason of querying and loading new flow table. High delay may affect experience of legal users, so high hopping frequency may not be user friendly. Denote HF_{RHM} , HF_{SEHT} , and HF_{PM} as the actual hopping frequency of RHM, SEHT, and the proposed method. Hopping frequency here is defined as hopping times divided by time. The hyperparameters are set as follows: $PSP = 60\%$, $FSP = 2$ s, $P_{BH} = 0.5$, $NoS = 50$. P_{LH} is equal to half the number of IPs in the IP pool of corresponding host.

Comparison of hopping frequency in normal period is shown in Table 7. In normal period without attack, their hopping frequencies are tested as $HF_{[12]} = HF_{RHM}$; $HF_{PM} < HF_{SEHT}$ under a set time interval. The proposed method performs better than SEHT for the reason of nearly not being disturbed by normal probes. Hopping frequency of RHM and the method in [12] is equal to T_{HH} . OF-RHM

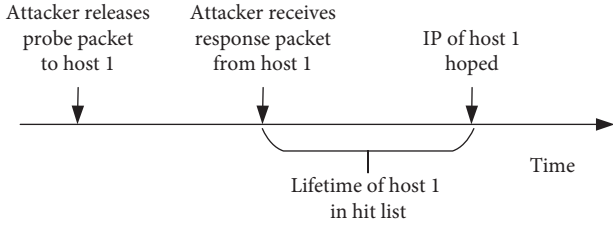


FIGURE 16: Lifetime of host 1 in attacker’s hit list.

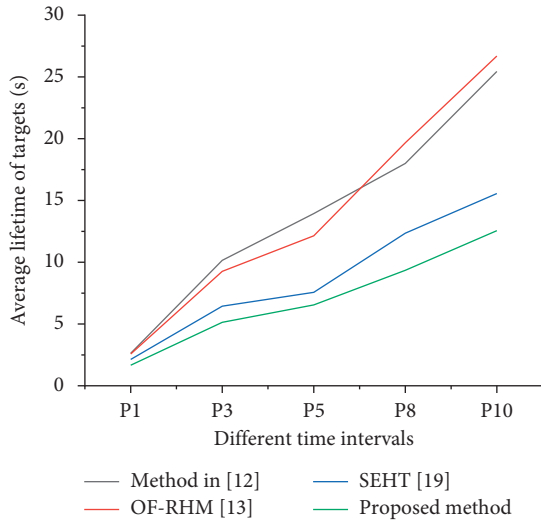


FIGURE 17: Average lifetime of targets in attacker’s hit list against different time intervals (50 scanners).

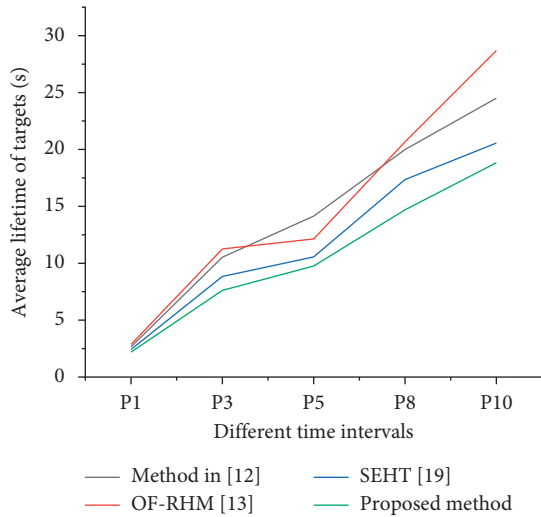


FIGURE 18: Average lifetime of targets in attacker’s hit list against different time intervals (100 scanners).

and the method in [12] achieve the best performance because of not conducting any adaptive strategy.

Comparison of hopping frequency in attack period is shown in Table 8. In attack period, their hopping frequencies are tested as $HF_{PM} < HF_{SEHT} < HF_{RHM} = HF_{[12]}$ when achieving same survival rate. The proposed method acts

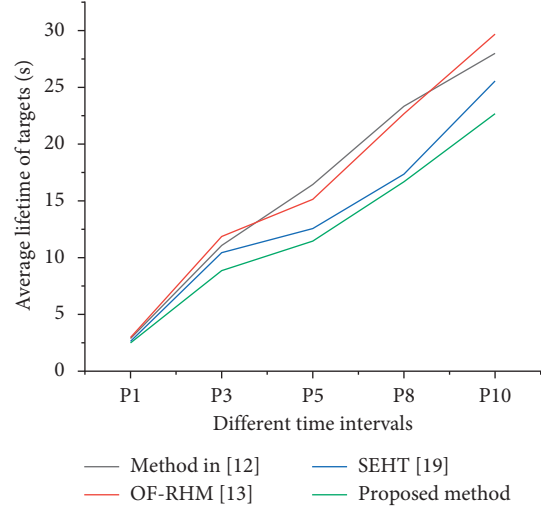


FIGURE 19: Average lifetime of targets in attacker’s hit list against different time intervals (150 scanners).

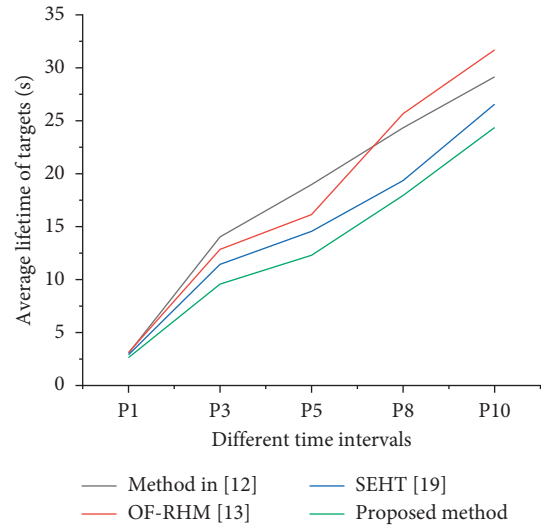


FIGURE 20: Average lifetime of targets in attacker’s hit list against different time intervals (200 scanners).

according to behavior of attack carefully to make each hop as effective as possible. The proposed method performs about 4%–14% better than SEHT and 21%–29% better than OF-RHM and the method in [12].

5.3. Performance Overhead and Limitation. While making some performance breakthroughs, our proposed method does have performance overhead and limitation. Firstly, as detailed in Section 4.2, both the used switches and controller should have enough cache space to temporarily store packets related to scanning attack. Secondly, training the CNN detector needs some devices and takes a period of time. Thirdly, our proposed method is designed and deployed in SDN environment. It cannot be used in legacy network environment directly. Adopting our proposed method to legacy network will be our future work.

TABLE 6: The average lifetime (s) of targets in attack's hit list caused by discarded data and data not discarded in the method of IP hopping with CNN assistance.

NoS	State	P1	P3	P5	P8	P10
50	Discarded	2.44	9.13	11.88	17.32	23.55
	Not discarded	0.83	0.88	0.89	0.86	0.86
100	Discarded	2.67	9.99	12.87	19.54	25.13
	Not discarded	0.83	0.87	0.88	0.88	0.89
150	Discarded	2.81	10.43	13.55	19.80	26.99
	Not discarded	0.86	0.86	0.87	0.89	0.87
200	Discarded	2.92	10.88	14.16	20.72	28.18
	Not discarded	0.85	0.86	0.86	0.88	0.86

TABLE 7: Hopping frequency (times/s) under different set time intervals in normal period.

Set time interval	OF-RHM [13]	SEHT [19]	Method in [12]	Proposed method
P1	0.200	0.297	0.200	0.213
P4	0.050	0.166	0.050	0.052
P10	0.020	0.119	0.020	0.023

TABLE 8: Hopping frequency (times/s) under different survival rates in attack period.

Survival rate	OF-RHM [13]	SEHT [19]	Method in [12]	Proposed method
0.2	0.033	0.027	0.033	0.026
0.4	0.067	0.054	0.067	0.049
0.6	0.125	0.103	0.125	0.089

6. Conclusions

Faced with scanning attack, in order to improve the effectiveness of IP hopping defense, this paper proposes an adaptive moving target defense system, which uses a designed light-weight CNN to sense changeable attack behavior and guide IP hopping defense. In terms of security, experiment result shows that the proposed method achieves better effectiveness of defense for it performs better under the indexes of survival rate of protected hosts and average lifetime of targets in attack's hit list compared with other methods. In terms of usability, experiment result shows that, when achieving same defense effect, the proposed method only needs lower hopping frequency, thus with lower overhead than other existing methods.

Our following work includes firstly improving the structure of CNN detector to improve the processing efficiency while keeping the accuracy, so as to reduce the rate of data loss and achieve better security of protected network; secondly, applying the proposed method to different scanning strategies such as follow-up scanning; thirdly, exploring the deployment scheme of our proposed method in legacy network.

Data Availability

The data that support the findings of this study are not publicly available due to restrictions as the data contain sensitive information about a real-world enterprise network. Access of the dataset is restricted by the original owner. Data are available upon request to the corresponding author, who

will apply for permission of sharing the data from the original owner.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (2019QY1301 and 2019QY1303), National Natural Science Foundation of China (Grant nos. 61902427 and 61802404), and the Science and Technology Leading Talent Project of Zhengzhou (Grant no. 131PLJRC644).

References

- [1] <https://symantec.broadcom.com/symc-istr-v24-2019-6819>.
- [2] J. H. Jafarian, E. A. Shaer, and Q. Duan, "An effective address mutation approach for disrupting reconnaissance attacks," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 12, pp. 2562–2577, 2015.
- [3] S. Jajodia, A. K. Ghosh, V. Swarup et al., *Moving Target Defense*, Springer, New York, NY, USA, 2011.
- [4] V. Krylov and K. Kravtsov, "DDoS attack and interception resistance IP fast hopping based protocol," 2014, <https://arxiv.org/abs/1403.7371>.
- [5] C. Zhao, C. Jia, and K. Lin, "Technique and application of End-hopping in network defense," in *Proceedings of the 2010 First ACIS International Symposium on Cryptography, and*

- Network Security, Data Mining and Knowledge Discovery, E-Commerce and Its Applications, and Embedded Systems*, pp. 266–270, IEEE, Qinquangdao, China, October 2010.
- [6] A. Clark, K. Sun, and R. Poovendran, “Effectiveness of IP address randomization in decoy-based moving target defense,” in *Proceedings of the 52nd IEEE Conference on Decision and Control*, pp. 678–685, IEEE, Firenze, Italy, December 2013.
 - [7] R. Zhuang, S. A. DeLoach, and X. Ou, “Towards a theory of moving target defense,” in *Proceedings of the First ACM Workshop on Moving Target Defense*, pp. 31–40, Scottsdale, AZ, USA, November 2014.
 - [8] J. Yackoski, P. Xie, H. Bullen et al., “A self-shielding dynamic network architecture,” in *Proceedings of the 2011-MILCOM 2011 Military Communications Conference*, pp. 1381–1386, IEEE, Baltimore, MD, USA, November 2011.
 - [9] J. H. Cho, D. P. Sharma, H. Alavizadeh et al., “Toward proactive, adaptive defense: a survey on moving target defense,” *IEEE Communications Surveys & Tutorials*, vol. 99, 2019.
 - [10] M. Dunlop, S. Groat, W. Urbanski et al., “MT6D: a moving target IPv6 defense,” 2011.
 - [11] C. Morrell, J. S. Ransbottom, R. Marchany et al., “Scaling IPv6 address bindings in support of a moving target defense,” in *Proceedings of the The 9th International Conference for Internet Technology and Secured Transactions (ICITST-2014)*, pp. 440–445, IEEE, London, UK, December 2014.
 - [12] L. Miao, H. Hu, and G. Cheng, “The design and implementation of a dynamic IP defense system accelerated by vector packet processing,” in *Proceedings of the International Conference on Industrial Control Network and System Engineering Research*, Shenyang, China, March 2019.
 - [13] J. H. Jafarian, E. Al-Shaer, and Q. Duan, “Openflow random host mutation: transparent moving target defense using software defined networking,” in *Proceedings of the first workshop on Hot topics in software defined networks*, pp. 127–132, NC, USA, August 2012.
 - [14] J. H. H. Jafarian, E. Al-Shaer, and Q. Duan, “Spatio-temporal address mutation for proactive cyber agility against sophisticated attackers,” in *Proceedings of the First ACM Workshop on Moving Target Defense*, pp. 69–78, Scottsdale, AZ, USA, November 2014.
 - [15] V. Krylov, K. Kravtsov, E. Sokolova et al., “SDI defense against DDoS attacks based on IP Fast hopping method,” in *Proceedings of the 2014 International Science and Technology Conference (Modern Networking Technologies)(MoNeTeC)*, pp. 1–5, IEEE, Moscow, Russia, October 2014.
 - [16] J. H. Jafarian, E. AlShaer, and D. Qi, “Adversary-aware IP address randomization for proactive agility against sophisticated attackers,” in *Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM)*, Hong Kong, China, April 2015.
 - [17] D. C. MacFarland and C. A. Shue, “The SDN shuffle: creating a moving-target defense using host-based software-defined networking,” in *Proceedings of the Second ACM Workshop on Moving Target Defense*, pp. 37–41, Denver, Colorado, USA, October 2015.
 - [18] S. Y. Chang, Y. Park, and B. B. A. Babu, “Fast ip hopping randomization to secure hop-by-hop access in sdn,” *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 308–320, 2018.
 - [19] C. Lei, H.-q. Zhang, and D.-h. Ma, “Network moving target defense technique based on self-adaptive end-point hopping,” *Arabian Journal for Science and Engineering*, vol. 42, no. 8, pp. 3249–3262, 2017.
 - [20] R. J. Smith, H. A. N. Zincir, M. I. Heywood et al., “Initiating a moving target network defense with a real-time neuro-evolutionary detector,” in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, pp. 1095–1102, Denver, CO, USA, July 2016.
 - [21] H. Zhang, C. Lei, D. Chang et al., “Network moving target defense technique based on collaborative mutation,” *Computers and Security*, vol. 70, pp. 51–71, 2017.
 - [22] D. Ma, C. Lei, L. Wang et al., “A self-adaptive hopping approach of moving target defense to thwart scanning attacks,” in *Proceedings of the International Conference on Information and Communications Security*, Springer International Publishing, Singapore, November 2016.
 - [23] J. Li, M. Zhang, and J. Wang, “DDoS attack detection based on RBFNN in SDN,” in *Proceedings of the International Conference on Advanced Hybrid Information Processing*, pp. 204–213, Springer, Yiyang, China, October 2018.
 - [24] A. Wagner, T. Dübendorfer, B. Plattner et al., “Experiences with worm propagation simulations,” in *Proceedings of the 2003 ACM workshop on Rapid Malcode*, pp. 34–41, New York, NY, USA, October 2003.
 - [25] N. McKeown, “Software-defined networking,” *INFOCOM keynote talk*, vol. 17, no. 2, pp. 30–32, 2009.
 - [26] Q. Y. Zuo, M. Chen, G. S. Zhao et al., “Research on OpenFlow-based SDN technologies,” *Journal of Software*, vol. 24, no. 5, pp. 1078–1097, 2013.
 - [27] N. McKeown, T. Anderson, H. Balakrishnan et al., “OpenFlow: enabling innovation in campus networks,” *ACM SIGCOMM - Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.
 - [28] N. Ketkar, *Convolutional Neural networks. Deep Learning with Python*, Apress, Berkeley, CA, USA, 2017.
 - [29] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, 2015.
 - [30] S. Antonatos, P. Akritidis, E. P. Markatos et al., “Defending against hitlist worms using network address space randomization,” *Computer Networks*, vol. 51, no. 12, pp. 3471–3490, 2007.
 - [31] D. Kewley, R. Fink, J. Lowry et al., “Dynamic approaches to thwart adversary intelligence gathering,” vol. 1, pp. 176–185, in *Proceedings of the DARPA Information Survivability Conference and Exposition II. DISCEX’01*, vol. 1, pp. 176–185, IEEE, Anaheim, CA, USA, June 2001.
 - [32] S. Yu, T. Thapngam, J. Liu et al., “Discriminating DDoS flows from flash crowds using information distance,” in *Proceedings of the IEEE 2009 Third International Conference on Network and System Security - Gold Coast*, pp. 351–356, Queensland, Australia, October 2009.
 - [33] C. C. Zou, D. Towsley, and W. Gong, “On the performance of Internet worm scanning strategies,” *Performance Evaluation*, vol. 63, no. 7, pp. 700–723, 2006.
 - [34] D. Teyou, G. Kamdem, and J. Ziazet, “Convolutional neural network for intrusion detection system in cyber physical systems,” 2019, <https://arxiv.org/abs/1905.03168>.
 - [35] N. N. Tran, R. Sarker, and J. Hu, “An approach for host-based intrusion detection system design using convolutional neural network,” in *Proceedings of the International Conference on Mobile Networks and Management*, pp. 116–126, Springer, Melbourne, Australia, December 2017.
 - [36] S. Z. Lin, Y. Shi, and Z. Xue, “Character-level intrusion detection based on convolutional neural networks,” in *Proceedings of the 2018 International Joint Conference on Neural*

- Networks (IJCNN)*, pp. 1–8, IEEE, Rio de Janeiro, Brazil, July 2018.
- [37] P. Wu and H. Guo, “LuNet: a deep neural network for network intrusion detection,” 2019, <https://arxiv.org/abs/1909.10031>.
- [38] S. Santurkar, D. Tsipras, A. Ilyas et al., “How does batch normalization help optimization?” in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 2488–2498, Montreal, Canada, December 2018.
- [39] V. Nair and G. E. Hinton, “Rectified linear units improve restricted Boltzmann machines,” in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814, Haifa, Israel, June 2010.
- [40] B. Lantz, B. Heller, and N. McKeown, “A network in a laptop: rapid prototyping for software-defined networks[C],” in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, pp. 1–6, Haifa, Israel, June 2010.
- [41] M. Abadi, P. Barham, J. Chen et al., “Tensorflow: a system for large-scale machine learning,” in *Proceedings of the 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp. 265–283, Savannah, GA, USA, November 2016.