Journal of Computer and Communications, 2015, 3, 203-211
Published Online May 2015 in SciRes. http://www.scirp.org/journal/jcc
http://dx.doi.org/10.4236/jcc.2015.35026

The Design and Research for Network Address Space Randomization in OpenFlow Network

Ziyu Zhao, Yuanbo Guo, Wei Liu

State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, China
Email: 1323308566@qq.com

Received April 2015

Abstract

By allocating IP address and changing IP address in source and destination hosts, network address space randomization is committed to construct a dynamic and heterogeneous network to decrease the attacking possibility and predictability. The research mainly deploys the features of OpenFlow network including data plane and control plane decoupling, centralized control of the network and dynamic updating of forwarding rules, combines the advantages of the network address space randomization technology with the features of the OpenFlow network, and designs a novel resolution towards IP conversion in Floodlight controller. The research can help improve the unpredictability and decrease the possibility of worm attacking and IP sniffing by IP allocation.

Keywords

IP Conversion, OpenFlow, Moving Target Defense, Floodlight, Hitlist Worms

1. Introduction

Network address space randomization is a kind of effective moving target defense (MTD) [1] technology. For moving target defense, it is one of the revolutionary technologies to "change the game rules in the network space field" proposed by the United States in recent years. It not only relies on the complexity of the security system itself to protect the target, but also makes full use of time, space, and the complexity of the physical environment, so as to greatly increase the attack difficulties. In addition to IP address variability, moving target defense also includes communication channel variability, routing and IP security protocol (IPSec) channel variability, network and host identity randomization, code randomization, address space randomization, instruction set randomization and data randomization etc.

This paper concentrates on the IP variability field. In this field, several efforts have been made. APOD [2] (Applications That Participate in Their Own Defense) model is based on address and port randomization hopping scheme to cover up terminal identity to avoid sniffing, but this scheme needs the cooperation of client and the server host in the IP changing period. DyNAT [3] scheme propose IP hopping strategy, changing IP addresses before the packets coming into the nuclear or public network, covering up IP addresses to prevent IP

How to cite this paper: Zhao, Z.Y., Guo, Y.B. and Liu, W. (2015) The Design and Research for Network Address Space Randomization in OpenFlow Network. *Journal of Computer and Communications*, **3**, 203-211.
http://dx.doi.org/10.4236/jcc.2015.35026

sniffing attack in the middle link. Although the IP hopping strategy can prevent sniffing attack, but for the scanners largely relying on the terminal detection response to scan the terminals, it doesn't work. NASR [4] (Network Address Space Randomization) is an address randomization strategy on the level of local area network (LAN), but its deployment is limited. And in the process of IP changing, its link will get interrupted. Besides, it needs to change the terminal operating system which is of high cost, so that it is difficult to realize. RHM (Random Host Mutation) strategy [5] have higher IP conversion frequency, and it is able to withstand malicious scanning and worm attacks; however, it is relatively complex to realize, and is not able to manage in a global view and achieve the effectiveness of dynamic updating. OF-RHM model [6] is first deployed in SDN architecture, but it's not a modular design and is not a complete solution especially in the authentication and IP conversion aspects.

This paper studies the IP address variability through assigning virtual IP to hosts in the use of OpenFlow. We combine IP variability in moving target defense technology with the OpenFlow network architecture, increase the vIP (virtual IP)-rIP (real IP) conversion module, authentication module and virtual IP generating module in its Floodlight controller, so that form a new scheme of IP conversion. It is a novel modular solution for IP variability in moving target defense field. Results show that it can obtain obvious effectiveness for preventing IP sniffing and worm attacks [7] [8] especially hit list worms to achieve effective protection for the target host by randomly assigning a virtual IP address to the source/destination host which will largely enhance the unpre- dictability thus increasing attack difficulty.

2. The Introduction of OpenFlow and Floodlight Controller

OpenFlow [9]-[11] is a novel network architecture proposed by N. McKeown et al in Stanford University. Devices in the traditional network forward data by the distributed management approach and the forwarding way of data from the source device to the destination device is determined by the router itself. In OpenFlow, the controlling module is detached from the devices and added to the outside web servers. The web server runs a controlling program to send instructions to switches and designs the forwarding rules. The controlling program can also provide a programmable interface for managers to take charge of the switches and reduce manual configuration process. Due to the characteristics such as centralized control of the network and dynamic updating of forwarding rules of OpenFlow, the OpenFlow network can be used in the moving target defense technology field with obvious advantages. For the controller can make decisions in a global view and directly deploy the resource effectively. Its programmability also plays a more important practical part, the programmers can directly develop the application through the API in the upper level.

OpenFlow Specification [12] defines the basic components and the functions. An OpenFlow switch consists of a flow table, which performs packet lookup and forwarding and a secure channel to an external control. The controller manages the switch over the switch channel using the OpenFlow protocol. The flow table contains a set of flow table entries, and each flow table entry contains header fields to match against packets, counters to update for matching packets and a set of zero or more actions to apply to matching packets. On receipt of a packet, an OpenFlow switch [13] performs the function shown in **Figure 1**.

The OpenFlow protocol supports three message types, Controller-to-Switch messages (including Features, Configuration, Modify-State, Read-State and Send-Packet), Asynchronous messages (containing Packet-in, Flow Expiration, Post-Status and Error) and symmetric messages (such as Hello, Echo and Vendor).

The OpenFlow controller is the control center for the entire OpenFlow network architecture. It owns a global view for the whole network. Recently, the most widely-used controllers include NOX, SNAC, Beacon, Floodlight, Bigswitch, Maestro etc. The controller is responsible for designing logical rules for data flow, and adding/removing flow table entries to transmit data flow on the designated path. Based on detailed analysis, we choose the functional (able to achieve multi-thread operation and cooperate with multiple controllers), well-portable

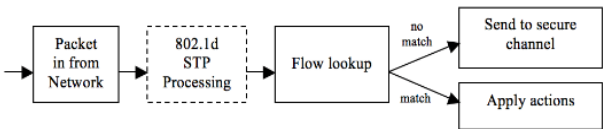

Figure 1. The function performed on a packet as it moves through an OpenFlow switch.

controller, Floodlight, to take the experiment. The main structure of Floodlight is shown in **Figure 2**.

The Floodlight Open SDN Controller [14] is an enterprise-class, Apache-licensed, Java-based OpenFlow Controller.

Floodlight adopts a modular architecture to implement its controller features and some applications. The main modules are FloodlightProvider, DeviceManagerImpl, LinkDiscoveryManager, TopologyService, RestAPI-Server, ThreadPool, MemoryStorageSource, FlowCache, PacketStreamer, OFSwitchManager. Among all the modules, FloodlightProvider provides two main pieces of functionality. It handles the connections to switches and turns OpenFlow messages into events that other modules can listen for. The second big function that it provides is decides the order in which specific OpenFlow messages are dispatched to the modules that listen for the messages. Modules can then decide to allow the processing of the message to go onto the next listener or to stop processing the message. And the functions of other main modules are shown in **Figure 3**.

3. The System Framework of IP Address Space Randomization in OpenFlow Network

3.1. The Design of System Framework

We add three main modules in Floodlight controller in order to generate IP address randomly, achieve IP variability and authenticate the identity of the unknown hosts in the communication process.

As shown in **Figure 4**, it is a part of the overall structure of the Floodlight controller. In the figure, Forwarding module, PacketStreamer module are already set in the controller, and the class OFMessageFilterManager in the PacketStreamer module can filter and classify incoming packet-in message. In contrast, the virtual IP (vIP) generating module, vIP (virtual IP)-rIP (real IP) conversion module and authentication module are novel-designed modules. OpenFlow switch (OF switch) are connected to hosts or server such as Web Server, and DNS server is linked to the controller through DeviceManager module.

3.2. The Communication between Source/Destination Hosts

1) Communication Based on Names

We define the source host as host 1 and the destination host as host 2. Assuming already knowing the name of host 2, host 1 sends a request message containing the name of host 2 to get connected. If no match is found in the OF switch (Open Flow switch), the OF switch will send the packet-in message to the controller. As shown in **Figure 5**, the Forwarding module in Floodlight is responsible to forward messages, then forwards requests of OF switch to the local DNS server. After conversion, DNS sends rIP (real IP) of host 2 to the vIP-rIP conversion module. The vIP-rIP conversion module receives vIP from the virtual IP generating module. So far, the vIP-rIP conversion module has restored all the vIP-rIP mapping relationships. Meanwhile, the Forwarding module sends the rIP of host 1 to the vIP-rIP conversion module and the vIP-rIP conversion module also receives vIP from the

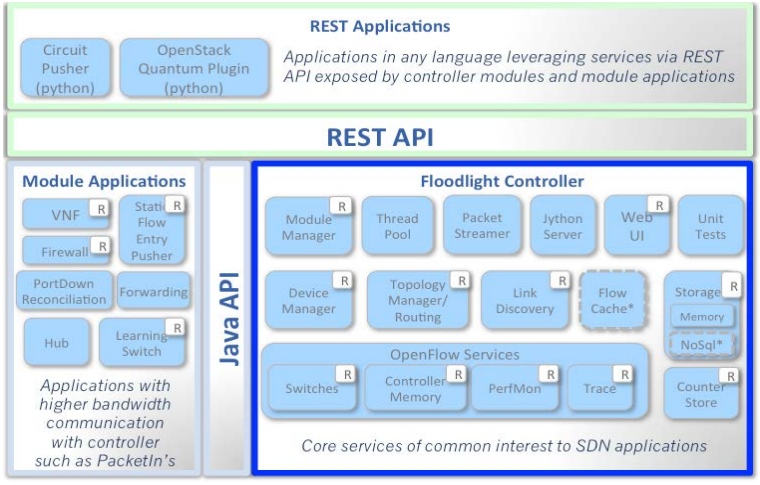

Figure 2. The structure chart of Floodlight.

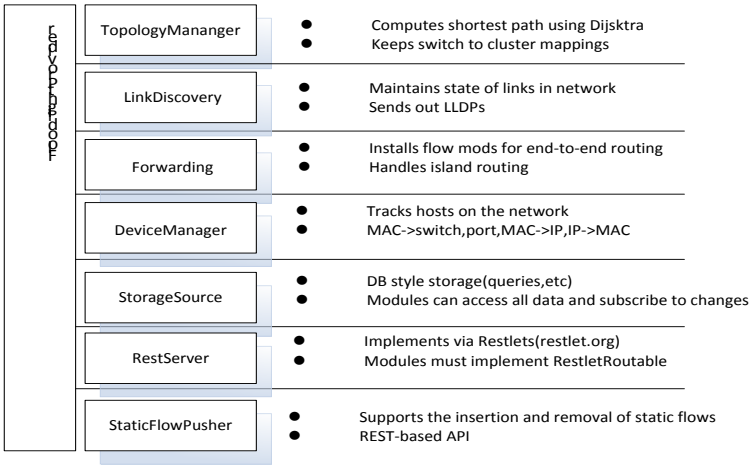

Figure 3. The functions of each main module in Floodlight.

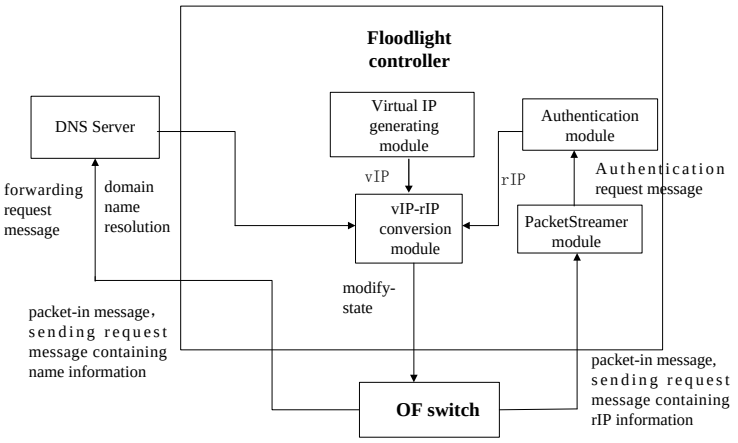

Figure 4. The overall frame chart of this system.

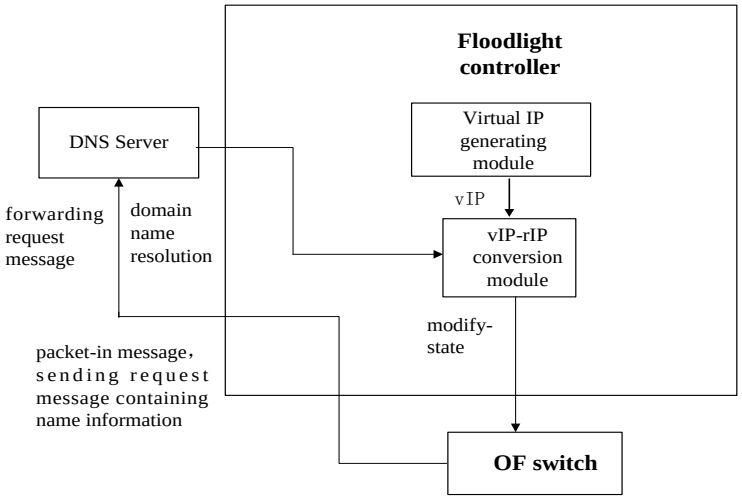

Figure 5. The module calling chart based on names.

vIP generating module, thus restoring the vIP-rIP mapping relationships of host 1 in vIP-rIP conversion module. vIP-rIP conversion module sends packet-out messages containing vIP of host 1 and host 2 to the OF switch and

the OF switch adds flow table entries.

2) Communication Based on Real IP Addresses

Knowing the rIP of host 2, host 1 sends messages containing the rIP of host 2 to get connected. If no match is found, OF switch will send packet-in request messages to the controller.

In **Figure 6**, PacketStreamer module in the controller will filter the IP addresses first and then forward to authentication module. Authentication module authenticates the identity of host1, and upon successful authentication, it forwards the rIP of both hosts to the vIP-rIP conversion module. vIP-rIP conversion module receives vIP from IP generating module and maintains the vIP-rIP mapping table of both hosts, then sends packet-out message containing vIP of both hosts to OF switch and adds flow table entries to the switch.

Besides, the communication between a host and a server is similar to the communication between the hosts. They can also communicate based on names and real IP addresses respectively.

4. The Design and Realization of Each Module in Floodlight Controller

4.1. The Design and Realization of vIP-rIP Conversion Module

vIP-rIP conversion module is the core module of this research. The controller is in charge of all the OF switches in certain domain and each switch is connected to the controller through the secure channel. The controller restores the vIP-rIP conversion mapping table.

The vIP-rIP conversion mapping restores the vIP and rIP for the source and destination hosts respectively. The rIP of host 1 (source host) is sent by Forwarding module, while the rIP of host 2 (destination host) is sent by DNS or authentication module. vIP of both hosts are sent by vIP generating module and meanwhile they can be regarded as the input parameters table in the controller.

We set two timers in this module. Timer 1 is responsible for updating the mapping table in the controller while Timer 2 takes charge of deleting the entries of the mapping table at regular intervals. Timer 1 is set to satisfy the most appropriate frequency according to the sensitivity of different hosts. Due to the exponential growth of the entries in the mapping table and OF switch, Timer 2 is set to consecutively remove the entries through flow-removed message.

4.2. The Design and Realization of Authentication Module

The design for this module is based on the access control strategy.

In Floodlight, we use an application to verify the registration information for each device when it accesses to the internet. The devices supporting the OpenFlow protocol are only needed to match the source MAC address.

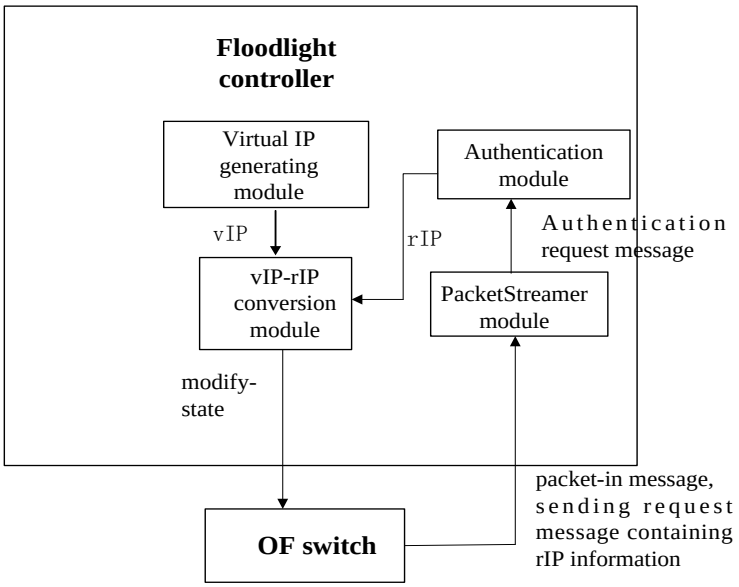

Figure 6. The module calling chart based on real IP addresses.

Besides, the OF switch must send the packets matching certain rules to the controller or adjust the flow to VLAN. The source host sends its first message which contains the real IP address information of the destination host. If no match is found, OF switch will forward the message to the controller. PacketStreamer module filters the packets and then sends the messages to the authentication module. Authentication module designs the rules for the accessing OF switches and designates access ports. For improving the efficiency of the authentication module, we set a cache to restore the authenticated data flow. If a rule is in the cache, the controller will send the rule directly to the OF switch in packet-out messages. Through this revised solution, we realize the authentication for the source host and finally achieve the goal of communication between hosts based on the real IP addresses.

4.3. Virtual IP Generating Algorithm

According to different sensitivity, each host has a peculiar IP conversion interval. To maximize the unpredictability, we must increase the randomness of IP address assignment. We can separate the IP time interval into two parts (T_1 and T_2). We need to assign available IP segments for the subnet S_k. Define the available IP segment $A_1,..,A_n$. Generate a IP address segment V by random (). In T_1, the segment V remains unchangeable. When h_1 (host 1) gets connected with h_2 (host 2), they will get certain virtual IP P_j. We use the pre-built hash function H and shared key K to compute the virtual IP of hosts in segment V. Thus, in T_2 (T_2 < T_1), h_1 and h_2 can have m virtual IP in segment V: $P_1,...,P_m$; in T (time interval), virtual IP P_j of host h_i is:

$$a_{(H(K\square j)\ mod\ m)+1}. \tag{1}$$

In this scheme, virtual IP generating module is responsible for generating random IP address. We adopt multi-level optimizing IP conversion technology to increase the randomness of IP selection thus enhancing attack difficulty. vIP-rIP conversion module is for maintaining vIP-rIP mapping table and updating DNS as well as setting timers to delete and update flow table entries. Authentication module is to authenticate the identity of hosts via communication.

Through adding three modules in Floodlight, we realize the IP assignment for source and destination hosts in the subnet. We utilize multi-level optimizing approach (separating different time intervals) to increase uncertainty by all available IP address space. We also adopt the advantages of OpenFlow network, for its centralized control and dynamic updating of transportation rules, to effectively add and remove flow table entries and achieve the goal of preventing vicious scanning and worm attacks.

5. Implementation and Evaluation

We use Mininet to simulate an OpenFlow network environment. We connect the external controller Floodlight. The OVS provides OF switch which provides the support for OpenFlow protocol in the software level, and it is connected by SSH. We define a network topology class MyTopo () by python API. Through Wireshark, we can directly observe the communication between Floodlight and OF switch. The OpenFlow protocol between the controller and OF switch is applied above TCP transport layer. The controller and Mininet communicate by OpenFlow 1.0 (shown in **Figure 7**). If OF switch finds no match in the header field, then sends packet-in messages to the controller and the controller sends out packet-out messages to add/modify entries, thus enabling the communication between two hosts (in **Figure 8**).

We use Nmap to scan the available network segment A1,...An (in **Figure 9**).

The following figures show the results of this scheme. It can be concluded that we decrease the worm propagation speed (**Figure 10**) and reduce the infectious hosts by minimal time intervals.

This scheme can prevent worm attacks by making their hit list soon out of date. Based on the reports of the scan, we conclude that less than 5% virtual IP are found by Nmap scanner (**Figure 11**).

Since the number of accessed hosts is exponential growing, we set two timers and the effectiveness of these timers is shown in **Figure 12**.

6. Conclusions

This paper provides a novel resolution for the moving target defense field. We design a scheme and realize it in OpenFlow network by adding three main modules, namely, the vIP (virtual IP)-rIP (real IP) conversion module, authentication module and virtual IP generating module, and study a IP generating algorithm.

511 55.204681	192.168.5.204	192.168.5.203	OpenFlow	64	Type: OFPT_HELLO
513 55.206109	192.168.5.203	192.168.5.204	OpenFlow	64	Type: OFPT_HELLO
514 55.210867	192.168.5.203	192.168.5.204	OpenFlow	64	Type: OFPT_FEATURES_REQUEST
516 55.211610	192.168.5.204	192.168.5.203	OpenFlow	232	Type: OFPT_FEATURES_REPLY
517 55.216100	192.168.5.203	192.168.5.204	OpenFlow	68	Type: OFPT_SET_CONFIG
518 55.216317	192.168.5.203	192.168.5.204	OpenFlow	64	Type: OFPT_GET_CONFIG_REQUEST
520 55.219443	192.168.5.204	192.168.5.203	OpenFlow	68	Type: OFPT_GET_CONFIG_REPLY
524 55.248766	192.168.5.204	192.168.5.203	OpenFlow	64	Type: OFPT_HELLO
526 55.253516	192.168.5.203	192.168.5.204	OpenFlow	64	Type: OFPT_HELLO
527 55.255844	192.168.5.203	192.168.5.204	OpenFlow	64	Type: OFPT_FEATURES_REQUEST
529 55.256420	192.168.5.204	192.168.5.203	OpenFlow	232	Type: OFPT_FEATURES_REPLY
530 55.257832	192.168.5.203	192.168.5.204	OpenFlow	68	Type: OFPT_SET_CONFIG
531 55.257969	192.168.5.203	192.168.5.204	OpenFlow	64	Type: OFPT_GET_CONFIG_REQUEST
533 55.258148	192.168.5.203	192.168.5.204	OpenFlow	128	Type: OFPT_FLOW_MOD
534 55.258352	192.168.5.204	192.168.5.203	OpenFlow	68	Type: OFPT_GET_CONFIG_REPLY
535 55.289854	192.168.5.203	192.168.5.204	OpenFlow	128	Type: OFPT_FLOW_MOD

Figure 7. Communication between the controller and OF switch.

193.666562	10.0.0.1	10.0.0.2	OpenFlow	172	Type: OFPT_PACKET_IN
193.674867	1e:f2:14:3b:f2:36	7e:82:d8:ca:7c:fe	OpenFlow	122	Type: OFPT_PACKET_OUT
193.676395	1e:f2:14:3b:f2:36	7e:82:d8:ca:7c:fe	OpenFlow	122	Type: OFPT_PACKET_OUT
193.676563	1e:f2:14:3b:f2:36	7e:82:d8:ca:7c:fe	OpenFlow	176	Type: OFPT_PACKET_IN
193.679148	192.168.5.203	192.168.5.204	OpenFlow	136	Type: OFPT_FLOW_MOD
193.679308	192.168.5.203	192.168.5.204	OpenFlow	64	Type: OFPT_BARRIER_REQUEST
193.679775	192.168.5.204	192.168.5.203	OpenFlow	64	Type: OFPT_BARRIER_REPLY
193.688289	192.168.5.203	192.168.5.204	OpenFlow	152	Type: OFPT_FLOW_MOD
193.688443	192.168.5.203	192.168.5.204	OpenFlow	64	Type: OFPT_BARRIER_REQUEST
193.688912	192.168.5.204	192.168.5.203	OpenFlow	64	Type: OFPT_BARRIER_REPLY
193.692702	10.0.0.1	10.0.0.2	OpenFlow	178	Type: OFPT_PACKET_OUT
193.703841	10.0.0.1	10.0.0.2	OpenFlow	178	Type: OFPT_PACKET_OUT
193.704395	10.0.0.1	10.0.0.2	OpenFlow	172	Type: OFPT_PACKET_IN
193.712097	10.0.0.1	10.0.0.2	OpenFlow	172	Type: OFPT_PACKET_IN

Figure 8. The packet-in and packet-out messages.

```
文件(F)  编辑(E)  查看(V)  搜索(S)  终端(T)  帮助(H)
root@kali:~# nmap 192.168.0.2-254

Starting Nmap 6.40 ( http://nmap.org ) at 2014-03-19 19:14 CST
Nmap scan report for 192.168.0.102
Host is up (0.0043s latency).
Not shown: 987 filtered ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
443/tcp   open  https
445/tcp   open  microsoft-ds
554/tcp   open  rtsp
902/tcp   open  iss-realsecure
912/tcp   open  apex-mesh
1935/tcp  open  rtmp
3389/tcp  open  ms-wbt-server
5357/tcp  open  wsdapi
49155/tcp open  unknown
49156/tcp open  unknown
49161/tcp open  unknown
MAC Address: 4C:0F:6E:FB:4D:55 (Hon Hai Precision Ind. Co.)
```

Figure 9. Nmap scanning for available network segment.

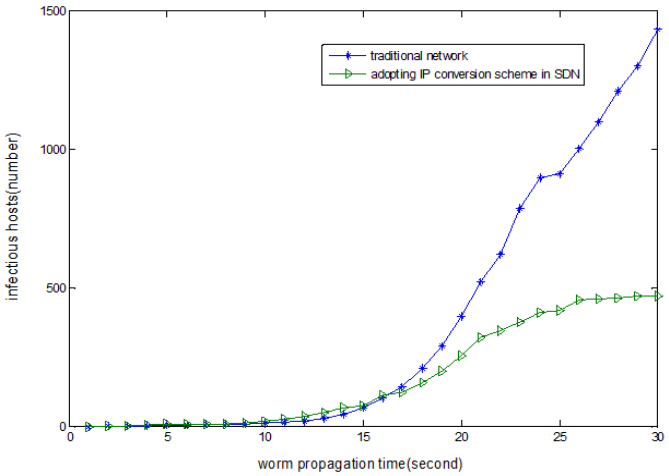

Figure 10. Worm propagation time and the infectious hosts.

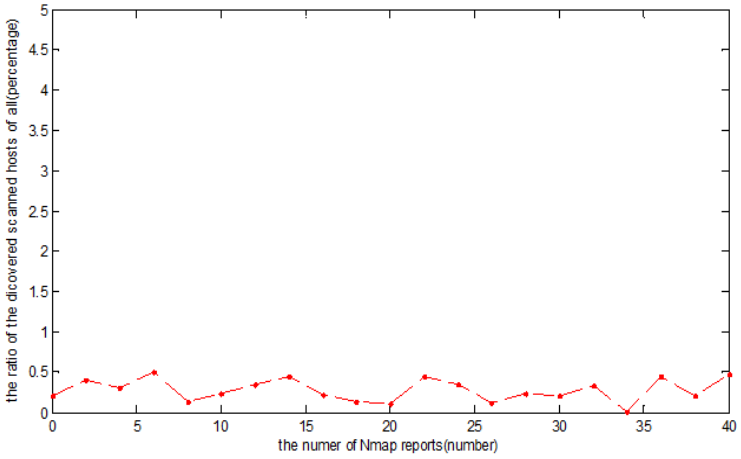

Figure 11. The virtual IP scanned by Nmap.

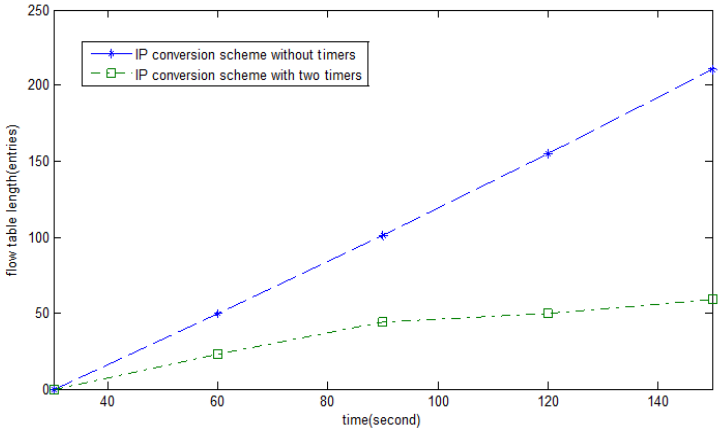

Figure 12. The length of flow table entries with time.

We are now studying random routing strategy to better defense worm attacks. For future, we hope to study access control and cryptography in OpenFlow network to increase security and decrease network attacking.

References

[1] Sushil, J., Anup, K.G., Vipin, S., *et al.* (2011) Moving Target Defense—Creating Asymmetric Uncertainty for Cyber Threats. Springer Press, 1.

[2] Atighetchi, M., Pal, P., Webber, F. and Jones, C. (2003) Adaptive Use of Network-Centric Mechanisms in Cyber-Defense. In *ISORC*'03, IEEE Computer Society, 183.

[3] Kewley, D., Fink, R., Lowry, J. and Dean, M. (2001) Dynamic Approaches to Thwart Adversary Intelligence Gathering. *Proceedings of DARPA Information Survivability Conference and Exposition II*. *DISCEX*'01, **1**, 176-185. http://dx.doi.org/10.1109/discex.2001.932214

[4] Antonatos, S., Akritidis, P., Markatos, E.P. and Anagnostakis, K.G. (2007) Defending against Hitlist Worms Using Network Address-Space-Randomization. *Computer Networks*, **51**, 3471-3490. http://dx.doi.org/10.1016/j.comnet.2007.02.006

[5] Al-Shaer, E. and Duan, Q. (2011) Random Host. IP Mutation for Moving Target Defense. Technical Report UNCCCYBERDN A-0728, University of North Carolina at Charlotte, NC, July.

[6] Jafar, H.J., Ehab, A. and Duan, Q. (2012) OpenFlow Random Host Mutation: Transparent Moving Target Defense Using Software Defined Network. *HotSDN*, **12**, 127-132.

[7] Zou, C.C., Towsley, D. and Gong, W. (2003) On the Performance of Internet Worm Scanning Strategies. *Elsevier Journal of Performance Evaluation*, **63**, 700-723.

[8] Moore, D., Shanning, C. and Claffy, K. (2002) Code-Red: A Case Study on the Spread and Victims of an Internet worm. In: *Proceedings of the* 2*nd Internet Measurement Workshop* (*IMW*), ACM, New York, 273-284. http://dx.doi.org/10.1145/637201.637244

[9] Benton, K., Camp, L.J. and Small, C. (2013) OpenFlow Vulnera-bility Assessment. In: *Proceedings of the* 2*nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, ACM, New York, 151-152. http://dx.doi.org/10.1145/2491185.2491222

[10] Nadeau, T.D. and Pan, P. (2011) Software Driven Networks Problem Statement. IETF Internet-Draft (Work-in-Progress) Draft-Nadeau-SDN-Problem-Statement-01, Oct. 2011.

[11] Kreutz, D., Ramos, F. and Verissimo, P. (2013) Towards Secure and Dependable Software-Defined Networks. In *Proceedings of the* 2*nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, ACM, New York, 55-60. http://dx.doi.org/10.1145/2491185.2491199

[12] Yadav, N. (2011) SDNs, OpenFlow 1.x, OpenFlow 2.0..., December.

[13] The OpenFlow Switch Consortium. OpenFlow SwitchSpecification Version 1.0.0, December 2009.

[14] Erickson, D. (2012) Floodlight Java Based OpenFlowController. Last Accessed, Ago.