EVHS - Elastic Virtual Honeypot System for SDNFV-Based Networks

Nguyen Canh Thang[1], Minho Park[*2] and Yang-Ick Joo[3]

[1]Department of Information Communication Convergence Technology, Soongsil University, South Korea
[*2]Corresponding author: School of Electronic Engineering, Soongsil University, South Korea
[3]Division of Electrical and Electronics Engineering, Korea Maritime and Ocean University, South Korea

***Abstract*:** The SDNFV-based network has leveraged the advantages of software-defined networking (SDN) and network-function virtualization (NFV) to become the most prominent network architecture. However, with the advancement of the SDNFV-based network, more attack types have emerged. This research focuses on one of the methods (use of the honeypot system) of preventing these attacks on the SDNFV-based network. We introduce an SDNFV-based elastic virtual honeypot system (EVHS), which not only resolves problems of other current honeypot systems but also employs a new approach to efficiently manage and control honeypots. It uses a network-intrusion-detection system (NIDS) at the border of the network to detect attacks, leverages the advantages of SDN and NFV to flexibly generate honeypots, and connects attackers to these honeypots by using a moving-target defense mechanism. Furthermore, we optimize the system to efficiently reuse the available honeypots after the attacks are handled. Experimental results validate that the proposed system is a flexible and efficient approach to manage and provide virtual honeypots in the SDNFV-based network; the system can also resolve the problems encountered by current honeypot systems.

***Keywords*:** Honeypot system, network intrusion detection system, software-defined networking, network function virtualization.

1. Introduction

In recent years, several changes have been made in com-puter networking, and many opportunities have arisen for researchers and industries, especially with the development of software-defined networking (SDN) and network-function virtualization (NFV). SDN represents a new architecture that decouples the data plane and control plan to centralize mon-itoring and control the network traffic [1-4]. In NFV, network functions are separated from hardware to decrease the cost and time involved in network-service deployment [5][6]. The SDNFV-based network is created using a combination of these two techniques; it offers more advantages in terms of management and operation than those offered by the traditional network [7].

Besides the growth of the network, the number of net-work attacks also increases significantly. Security tools, such as the network intrusion detection system (NIDS), network intrusion prevention system (NIPS), firewall, and honeypot system, should also be deployed to prevent these attacks. However, many problems are encountered when deploying these tools (especially the honeypot system). The honeypot system is a proactive defense mechanism that is built to connect attackers with fabricated hosts (honeypots) instead of real ones [8] – [14]. Once attackers enter the honeypots, their behaviors are monitored, and the administrators can detect any suspicious actions.

Several issues that can affect the deployment of current hon-eypot systems remain to be addressed. The main drawbacks are listed subsequently:

- Costs: Current honeypot systems always need to create several available honeypots, both physical and virtual, to become proactive. These honeypots should be ready at any time to handle any new attackers, which implies that the systems should create many honeypots beforehand. However, it is difficult to scale the definite size of the honeypot system; further, the capability of such a system to realize large-scale deployment is limited owing to its infrastructure and operating costs.
- Adaptability: Computer networking has gained signifi-cant interest over the past few years; this has led to many new types of services and network attacks. Each honeypot should only handle one attack type, and current honeypot systems cannot quickly adapt to the changes in network topologies and attack types.

Thus, the current honeypot systems have several disadvan-tages, and it is difficult to deploy them in future network archi-tectures such as the SDNFV-based networks. This necessitates the development of a new honeypot system that can address all these issues.

Herein, we propose a new elastic virtual honeypot system to resolve the critical issues discussed earlier in the SDNFV-based network. The proposed system leverages the advantages of emerging technologies, SDN (controls the network traffic) and NFV (manages the virtual infrastructure). It enhances the current architecture by generating and managing virtual honeypots; it monitors the incoming traffic in the SDNFV-based network through an NIDS and then decides to generate a new honeypot to handle the network traffic from a suspi-cious address. If honeypots are available, the proposed system automatically redirects the attack traffic to them by applying a moving-target defense mechanism. These algorithms help in handling the issues with the scale of the honeypot system. We solve the problem associated with many attack types by using the database and reusing the honeypots. The information of the honeypot, attacker, and attack type is stored in a database. Each honeypot can handle one attack type from one attacker, and after that attacker gets blocked or disappears, the proposed system reuses the honeypot for the same attack type in the future. Using these algorithms, the issues discussed earlier can be resolved, and the efficiency and performance of the proposed system can be improved.

The major contributions of this research are summarized subsequently:

- The problems of the current honeypot systems (i.e., cost and adaptation) are described.
- A new elastic virtual honeypot system, based on SDN and NFV technologies, which can detect network attacks us-ing an NIDS, generate and manage honeypots elastically, use the moving-target defense mechanism to connect attackers to honeypots, and reuse honeypots efficiently, is proposed. It can improve resource management and resolve the problems discussed earlier.
- Several experiments are conducted to evaluate the perfor-mance and efficiency of the new system.

The remainder of this paper is organized as follows. Section 2 presents brief introductions to the honeypot system, SDNFV-based network, and moving-target defense mechanism. Section 3 discusses research related to this study. The design of the proposed system is detailed in Section 4. Section 5 focuses on the detailed implementation and evaluation of the proposed system. Finally, the conclusion and scope for future research are presented in Section 6.

2. Background Knowledge

2.1 SDNFV-based Network

In the past few years, both academia and the industry have continuously conducted research on SDN and NFV technolo-gies owing to their potentials. SDN enables network program-ming by decoupling the control plane and the data plane. The data plane is a simple forwarding device that receives forwarding rules from the controller in the control plane. The first SDN protocol that allows a single controller to control and manipulate the network infrastructure is OpenFlow. The NFV is an architecture that decouples the network service functions, such as routing, load balancing, and firewalls, from the dedicated hardware. This implies that the network functions are packaged in the form of virtual machines (VMs) on commodity hardware and can easily control and manage the adaptation as per the network requirements. SDN and NFV improve the speed of deployment of network functions and scalability; further, they utilize network resources effectively. Furthermore, Wireless Sensor Network (WSN) can also be considered as similar to SDNFV-based network, where gate-ways or controllers control all network operations and each end device performs as the network function [15].

The SDNFV-based network architecture is detailed in Figure 1, which includes the SDN controller, NFV orchestrator (NFV-O), NFV infrastructure, and forwarding devices [7]. The SDN controller determines the network traffic flows and commu-nicates with the forwarding devices (SDN switches) to apply policies from the control plane to the data plane. The NFV infrastructure includes virtualized network functions (VNFs), which run as VMs on hypervisors and physical servers. The NFV-O is responsible for lifecycle management including the instantiation, scaling, and termination of VNFs. It is integrated with the SDN controller through northbound interfaces or APIs. After checking the policy requirements and generating the network configuration and topology, the NFV-O produces optimal function assignments and assigns the function to certain VMs in the optimized path; this is known as service function chaining.

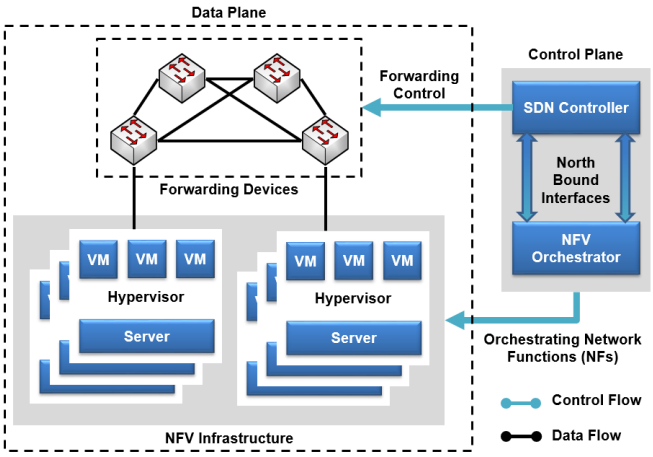

Figure 1. SDNFV-based network architecture.

2.2 Honeypot System

The honeypot system employs a proactive defense mech-anism that is built to connect attackers with fabricated hosts (honeypots) instead of real ones [8] – [14]. Attackers think that they are connecting with real victims, and they begin their attacks. A honeypot can be classified as a low-interaction honeypot (LIH), medium-interaction honeypot (MIH), or high-interaction honeypot (HIH) on the basis of its level of in-teraction [8] – [12]. The LIH only simulates services that are frequently requested by attackers; these include a remote desktop or a secure shell (SSH). The MIH partially or fully implements services to emulate a well-known vendor's imple-mentation. The HIH is most similar to a real system that runs various applications such as web servers and databases. The HIH aims to capture the maximum amount of information of the attackers' techniques. Before the proposal of the virtual honeypot system, there were several honeypot projects, called honeynet (generations I, II, and III) [13] – [14]. The deployment of these generations of honeynet was always based on physical servers and network devices, which implied that one physical server can only support one honeypot. This architecture is neither scalable nor flexible; further, it is too difficult to redeploy after security experts collect all the adversary information. Therefore, the virtual honeypot architecture resolves this problem; it runs multiple virtual honeypots on one physical host and reduces the other infrastructure costs.

A honeypot system can be integrated with an NIDS while it works as the detection sensor. The honeypot system helps security experts detect suspicious activities of network packets that do not adhere to the rules of the NIDS. It can support the NIDS in the detection and prevention of attacks and increase the detection accuracy.

2.3 Moving Target Defense Mechanism

This mechanism was exhibited in the SDN environment, which is detailed in Figure 2. When it is detected that an attacker is attempting to send exploit code to the victim host (1 and 2), the controller commands the switch to redirect the network traffic to the shadow victim (3). The shadow victim then responds to the attacker's request (4 and 5). The attacker is deceived into communicating with the shadow victim instead of the real one.

The moving-target defense mechanism allows security ex-perts to create, analyze, evaluate, and deploy mechanisms and strategies for the victim to continuously shift and change properties over time for increasing the complexity and costs for attackers. It also limits the vulnerability of the host to

attacks, and it increases the system resilience. The recently proposed MTDM is feasible for reducing the percentage of successful attacks by altering or diversifying the attributes or parameters of a protected system. If MTDM is applied to a security system, attackers will encounter more difficulties in launching attacks, which is one of the game-changing themes [16], [17].

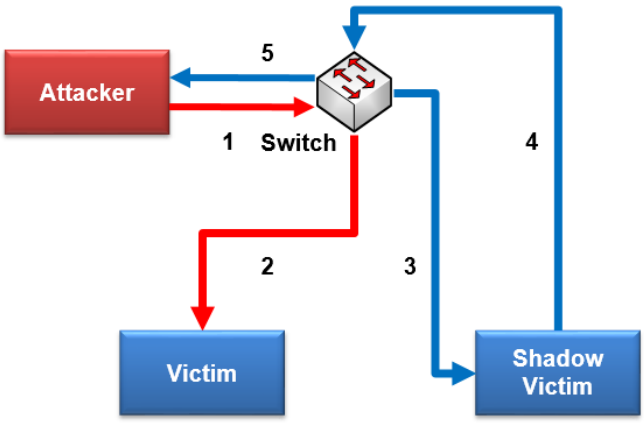

Figure 2. Moving-target defense mechanism.

3. Related Works

3.1 Honeypot System

Several solutions to address the problems of cost and complexity of the virtual honeypot system have been proposed. The hybrid honeypot system was proposed; it includes multiple levels of honeypots, such as the LIH at the front end and HIH at the back end [9], [10]. Once sufficient information is collected from the LIH, the hybrid system redirects the current traffic to the HIH. However, because both honeypot types exist simultaneously in the hybrid system, the resource consumption and complex operations are more. W. Fan [11] proposed a honeypot architecture that can dynamically reconfigure virtual honeypots; both Snort and Honeybrid Gateway were used to make decisions for the packets. The inbound traffic needs to travel through two filter layers; then, the back end can be easily flooded by invalid data. This architecture also requires the preinstallation of all the virtual honeypot instances, which does not resolve the cost problem. T. Lengyel [18] introduced a hybrid honeynet that is based on the clone feature of the Xen virtualization platform; a hybrid honeynet was used to perform routing between the attacker and a combination of LIH and HIH. However, this solution decreases the throughput because each packet is duplicated, which implies that the network performance is also affected.

3.2 Honeypot System In SDNFV-based Network

Many proposals have been made to improve the honeypot system using features and benefits of SDN and NFV. W. Han [19] proposed HoneyMix, which is a hybrid honeypot system running on the SDN-enabled network. When it receives requests from an attacker, HoneyMix redirects these requests to all the available honeypot instances and selects the most appropriate honeypot to respond to the attacker. However, if the number of attacks is larger than the number of honey-pots, HoneyMix cannot continue handling new attackers. This problem also exists in the proposal made by B. Park [20], but this proposal only generates a new honeypot when a new attack is detected. S. Kyung et al. [21] presented a newer version of HoneyMix, which is called HoneyProxy. There is a new proxy module that operates in three modes: transparent mode (T-Mode), multicast mode (M-Mode), and relay mode (R-Mode). Each packet, coming from the outside, needs to be inspected and then forwarded to the network. Because HoneyProxy works as a reverse proxy between honeypots and attackers, this approach also introduces a bottleneck for the SDN controller. W. Fan [22] proposed a method of transparently redirecting traffic from the LIH to HIH. At the 3-way handshake phase, the SYN packet is stored at the front end of the application. Hence, this system is highly susceptible to an SYN flood attack. The packets also need to go through the SDN application and Snort (IDS) to make a decision that creates delays in packet processing.

4. System Design

In this section, we analyze and describe the objectives and design (including the main components) of the proposed system, respectively.

4.1 Objectives

To address the vulnerabilities discussed in Section 2.2, the proposed system needs to achieve the following objectives:

- Adaptability: it should be able to adapt to the changes in network topology and attack types.
- Security: it must ensure that attackers do not reach real hosts.
- High Performance: it needs to generate new honeypots or connect attackers to available honeypots immediately.
- Efficiency: it needs to manage honeypots effectively to
- reduce the costs and handle a new attack at any time.
- Applicability: it should be easily applicable to the real system.

To achieve these objectives, a few assumptions need to be relied upon for designing the system. First, we eliminate the possibility of any compromised components in the network. Furthermore, the connections between network entities need to be reliable. If attackers can access switches or controllers, all the algorithms can be exploited and defeated.

4.2 Methodology

At the beginning, there are no honeypot instances in our system. We use an NIDS at the border of the network to detect attacks. When an attack is detected, a program called Agent sends a message to the EVHS controller to notify it about this attack.

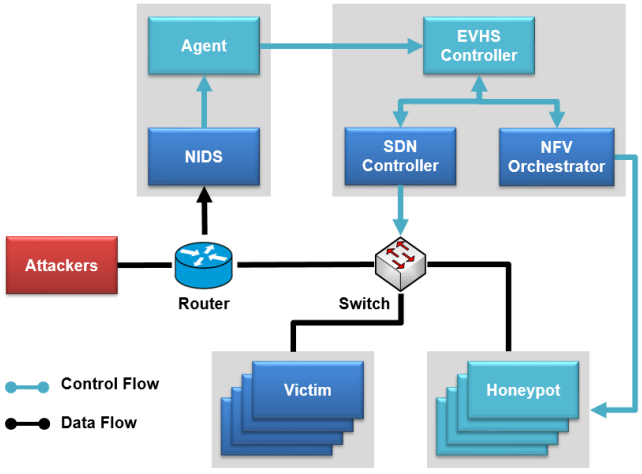

Figure 3. System architecture.

Then, our system generates a new honeypot or uses an available honeypot based on the system state to handle the new attack. The attack traffic is redirected to the honeypot through the moving-target defense mechanism. At this point, we can investigate and monitor all the activities of the attackers and protect the real system. The architecture and procedures are detailed in the next subsections.

4.3 System Architecture

The general system architecture is depicted in Figure3, and has four main components as follows:

4.3.1 Controllers

They work on the control plane of the network and can be categorized into three types, each with their own specific tasks as follows:

- SDN Controller: This type of controller sends the requests to the SDN forwarding devices (gateway and switches) to set up the network topology and also performs the moving target defense mechanism.
- NFV Orchestrator: Based on the requirements of other controllers, this controller manages the creation and op-eration of the NFV infrastructure (which is the honeypot in our scheme).
- EVHS Controller: This is the main controller in our system, and is depicted in Figure 4. The Agent message han-dling module (AMHM) handles the messages from the Agent while the honeypot management module (HMM) interacts with other controllers to set up the topology, and generate or reuse the honeypots to handle attacks. The details of the algorithms are presented in Subsection IV-D.
- Database: The three controllers share the same database that stores the network information regarding the topol-ogy and honeypots.

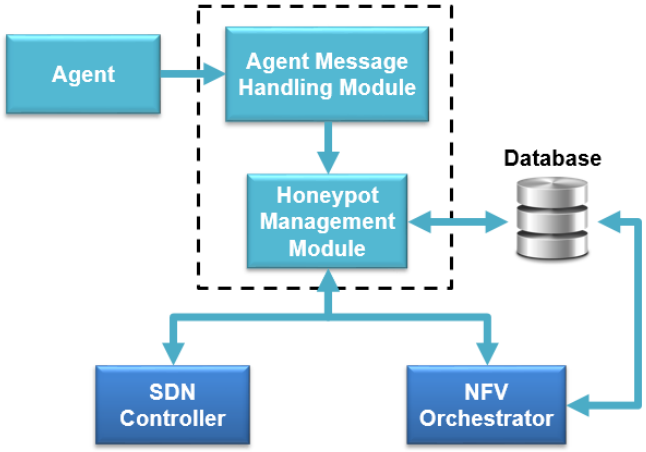

Figure 4. Controller's architecture.

4.3.2 Routers and switches

These are the forwarding devices that follow the command from the SDN controller to create the network topology and forward the network traffic.

4.3.3 Victims and Honeypots

They are controlled by the NFV orchestrator. Honeypots are generated or reused to handle attacks from attackers.

4.3.4 NIDS & Agent

The NDIS is located at the border of the network and the network traffic is mirrored to the NIDS from the router or switch through a span port. The NIDS is configured with rules to detect attacks from the network traffic and logs the alarms. The Agent observes the NIDS logs and retrieves the information on the attackers and victims (e.g, attacker's IP, victim's IP, attack type, etc.) from the alarms, and then sends this information to the EVHS controller.

4.4 Main Algorithms

The main algorithms of our system are depicted in Figure 5 and are divided into two main modules as follows:

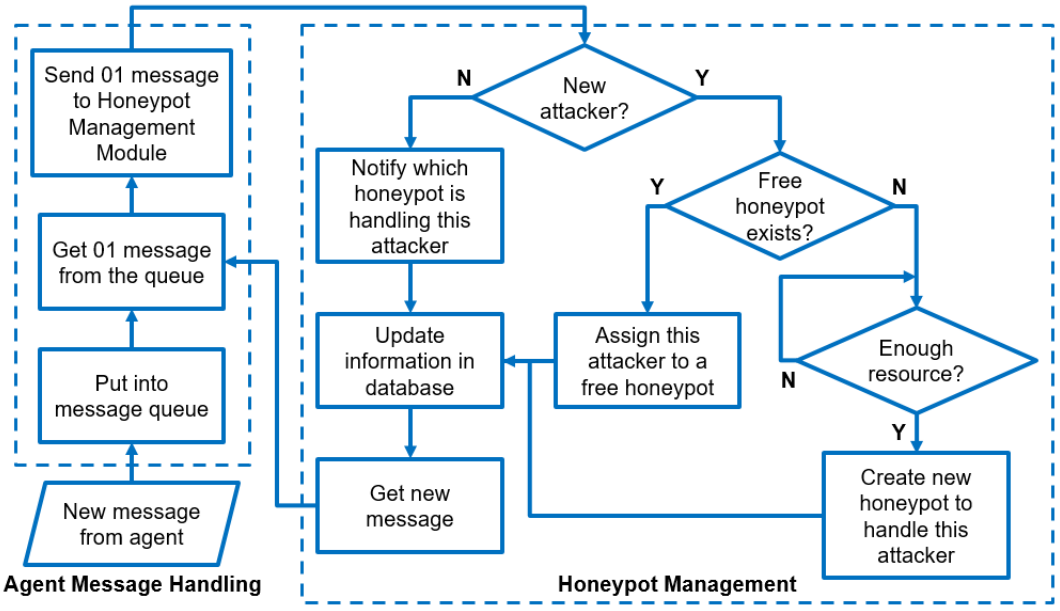

Figure 5. Main algorithms.

4.4.1 Agent Message Handling Module - AMHM

After receiving a message from the Agent, AMHM places the message in a message queue to process later. Owing to the possibility of several attacks occurring simultaneously, at which time our system would take longer to get the honeypots ready, we use the message queue and process them one by one. The HMM triggers the AMHM to send a new message after processing the previous one.

4.4.2 *Honeypot Management Module - HMM*

After receiving information regarding an attack, the HMM checks if the attacker is new by referring to the database. Table 1 presents the parameters of the attackers and honeypots. If the attacker is recognized, HMM notifies the honeypot currently handling this attacker, updates the database with the time at which this attacker appears again, and triggers the AMHM to obtain a new message. If the attacker is new, HMM checks if any honeypot is available for this attack type. If a honeypot is available, the attacker is connected to the available honeypot immediately by using the moving target defense mechanism, which also helps in reducing the operating cost by eliminating the need to create a new honeypot. If no honeypot is available, HMM checks if there are sufficient resources to generate a new honeypot. If sufficient resources are available, it generates a new honeypot based on the attack type, updates the information in the database, and then obtains a new message from AMHM again. In our research, we do not focus on the management of attack types, but instead focus on the effective management of the virtual honeypot.

Table 1. Attackers' Information In Database

Parameters	Descriptions
AtkIP	IP address of attacker
AtkType	Attack type
VicIP	IP address of victim
Timestamp	To calculate free time T_{free} of honeypot

Furthermore, the CPU and memory usage for each honeypot is related to the capacity of the honeypot system. Our system needs to monitor this information to ensure that it has sufficient resources to handle new attacks at any time. Therefore, we adopt another optimization algorithm periodically for HMM, as depicted in Figure 6. For instance, once an attacker is handled (i.e., the attack is complete), and there is no new attacker to connect to, a timestamp is updated in the database to indicate to the system that a particular honeypot is available. The time from when the honeypot is free until when the optimization algorithm starts is referred to as T_{free}. We set two thresholds for this interval, which are described as follows:

- If $T_{free} > T_{free\text{-}min}$, HMM sets the honeypot to be idle. At this time, the honeypot sleeps and waits for the NFV orchestrator to wake it up, which reduces the energy and operating cost of the system.
- If $T_{free} > T_{free\text{-}max}$, HMM removes the honeypot from the network, which helps decrease the operating costs and enables more resources to be available for other system tasks.

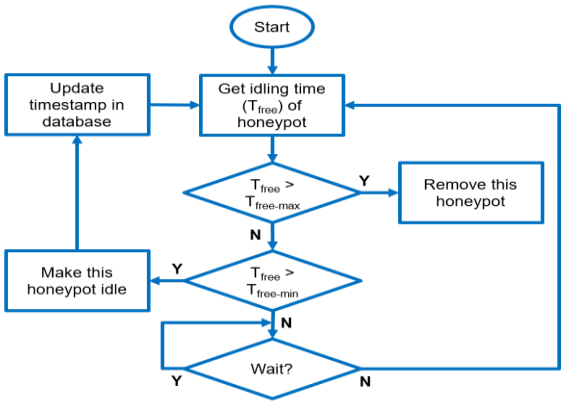

Figure 6. Honeypot management optimization algorithm.

5. Implementation and Evaluation

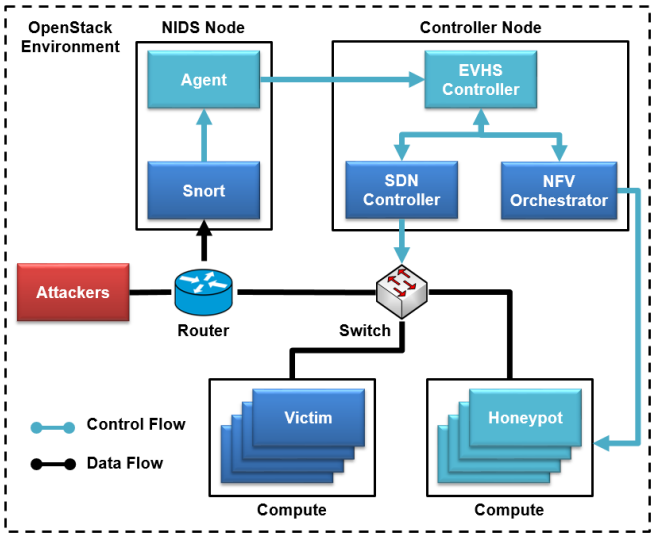

Figure 7. Implementation model.

5.1 Implementation

To evaluate the performance and efficiency of our system, we set up the experiment topology as depicted in Figure 7. The system is implemented in the OpenStack environment (Stein version) to take advantage of the SDN and NFV features. We implement our prototype on a computer with 12 CPUs Intel i7-8700 3.4 GHz and 48 GB memory. The prototype consists of five nodes, which can be described as follows:

- 01 Attacker node (01 CPU & 01 GB memory): This node runs a program to perform attacks from many fake IPs.
- 02 Compute nodes (03 CPUs & 08 GB memory): These nodes store the victim and honeypots, which run as VMs.
- 01 NIDS node (01 CPU & 01 GB memory): This node is installed at the border of the network, and connects to the router through a span port. All the network traffic is also forwarded to the NIDS by this port. We write a small program called Agent, which works on this node to observe the NIDS logs and send messages to the EVHS controller when attackers are detected. We use Snort as the NIDS with preconfigured rules to detect attacks [24], [25].
- 01 Controller node (04 CPUs & 30 GB memory): This node contains the EVHS controller and other entities. All programs need only more than 700 lines of C code and can be easily modified and integrated into other real systems.

5.2 Evaluation

In this section, we evaluate the performance, efficiency, and applicability of our system through three experiments. The first experiment focuses on the response time that the system requires to handle attackers. In the second experiment, we evaluate the efficiency of the system by observing the number of honeypots and attackers based on time. In the last experiment, we evaluate the CPU and memory consumption of our programs (agent, EVHS controller) on the NIDS node and controller node, respectively.

5.2.1 *Response Time Evaluation*

The time from when the attacker starts attacking until when the attacker receives a response from the victim is referred to

as response time. We calculated the response time and compared it with the results of most related works (DVNH) [20], and the results are depicted in Figure 8. DVNH needs 19 seconds to generate a new honeypot and connect it to the attacker, whereas our system needs only 13.551 seconds. Furthermore, because we use algorithms to reuse the available honeypots, the average response time of our system is only 7.099 seconds. Based on these results, we can conclude that our system performs attack handling instantly.

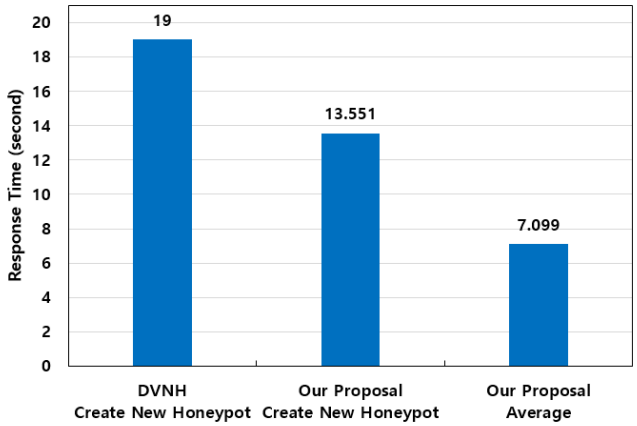

Figure 8. Response time of our system to attacks in comparison with DVNH [20]

5.2.2 Efficiency Evaluation

In this experiment, we set up conditions as presented in Table 2. A new attacker is created randomly from 1 to 5 minutes, and one attacker is deleted randomly after 5 to 10 minutes. After 15 minutes without being attached to any new attacker, honeypots are set to be idle and are removed after 60 minutes if they continue to remain idle. After 6 hours, we obtain the result depicted in Figure 9.

Table 2. Experiment Conditions

Name	Action	Time (minute)
Attacker	Create 01 new after	Random 1~5
	Delete 01 after	Random 5~10
Honeypot	Go to idle if T_{free} >	15
	Remove if T_{free} >	60

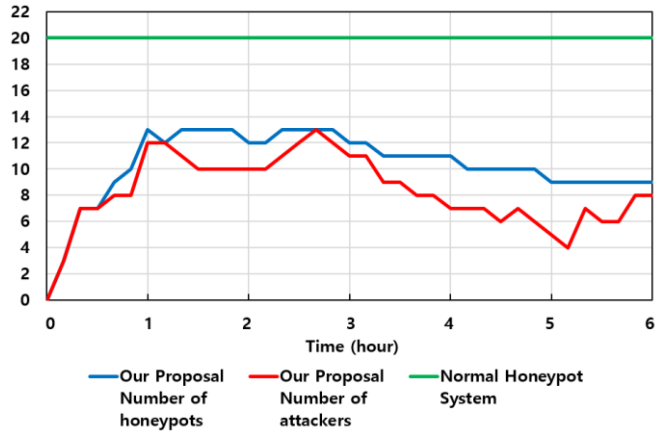

Figure 9. Efficiency on managing honeypots.

The baseline (green line) is the normal honeypot system, which is set at 20. This implies that the normal honeypot system needs to create 20 honeypots before handling any new attackers. In our results, the number of honeypots (blue line) always follows the number of attackers (red line). With the optimization algorithm, our system can adapt to the change in the attack number and attack type, by reusing the available honeypot without creating a new one. When the attackers gradually disappear, the number of honeypots also steadily decreases. Particularly, in our experiments, there is no idling honeypot because all honeypots are reused efficiently.

5.2.3 Resources Consumption Evaluation

To evaluate the performance and applicability of our system, we continue to evaluate the CPU and memory utilization of our programs on the NIDS node and controller node. For all our programs (Agent and EVHS controllers), the CPU and memory consumption is only under 0.1% on the NIDS node (1 CPU and 1 GB memory) and controller node (4 CPUs and 30 GB memory), respectively. These results indicate that our programs work well and cause a significant overhead to the operation of the system.

6. Conclusion & Future Works

In this research, we introduced an EVHS for an SDNFV-based network. Additionally, we addressed the complexity and issues of current honeypot systems. This proposal not only resolves these problems but also brings a new approach to manage and control the honeypot system efficiently. The system uses NIDS at the border of the network to detect attacks, and takes advantage of the SDN and NFV technologies to flexibly generate honeypots and connect attackers to them by using the moving target defense mechanism. Furthermore, we optimize the system to efficiently reuse the available honeypots after handling attacks. Through the experimental results, we can validate that our system is an elastic and efficient approach to manage and provide virtual honeypots in the SDNFV-based network and can resolve the problems of current honeypot systems.

In the future, we intend to enhance our system by forcing multiple attackers of the same attack type to attack only one virtual honeypot to reduce the maximum resource consumption of virtual honeypots, and apply more malicious detection mechanisms to cover a wider range of attacks and improve detection accuracy.

7. Acknowledgement

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No 2018-0-00254, SDN security technology development).

Reference

[1] S. Sezer et al., “Are We Ready for SDN? Implementation Challenges for Software-Defined Networks,” in IEEE Communications Magazine, vol. 51, no. 7, pp. 36-43, July 2013.

[2] B. A. A. Nunes, M. Mendonca, X. Nguyen, K. Obraczka, and T. Turletti, “A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks,” in IEEE Communications Surveys & Tutorials, vol. 16, no. 3, pp. 1617-1634, Third Quarter 2014.

[3] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, “A Survey on Software-Defined Networking,” in IEEE Communications Surveys & Tutorials, vol. 17, no. 1, pp. 27-51, First quarter 2015.

[4] D. Kreutz, F. M. V. Ramos, P. E. Ver´ıssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig,

"Software-Defined Networking: A Com-prehensive Survey," in Proceedings of the IEEE, vol. 103, no. 1, pp. 14-76, Jan. 2015.

[5] F. Reynaud, F. Aguessy, O. Bettan, M. Bouet, and V. Conan, "Attacks Against Network Functions Virtualization and Software-Defined Networking: State-of-the-Art," 2016 IEEE NetSoft Conference and Work-shops (NetSoft), Seoul, 2016, pp. 471-476.

[6] M. Pattaranantakul, R. He, Q. Song, Z. Zhang, and A. Meddahi, "NFV Security Survey: From Use Case Driven Threat Analysis to State-of-the-Art Countermeasures," in IEEE Communications Surveys & Tutorials, vol. 20, no. 4, pp. 3330-3368, Fourth quarter 2018.

[7] Y. Li and M. Chen, "Software-Defined Network Function Virtualization: A Survey," in IEEE Access, vol. 3, pp. 2542-2553, 2015.

[8] M. Shukla, P. Verma, "Honeypot: Concepts, Types and Working," International Journal of Engineering Development and Research (IJEDR), ISSN:2321-9939, vol. 3, no. 4, pp. 596-598, December 2015.

[9] M. Bailey, E. Cooke, D. Watson, F. Jahanian, and N. Provos, "A Hybrid Honeypot Architecture for Scalable Network Monitoring," Technical Report CSE-TR-499-04, University of Michigan, 2004.

[10] H. Artail, H. Safa, M. Sraj, I. Kuwatly, and Z. Al-Masri, "A Hybrid Honeypot Framework for Improving Intrusion Detection Systems in Protecting Organizational Networks," Comput. Secur., vol. 25, no. 4, pp. 274288, Jun. 2006.

[11] W. Fan, D. Fernandez, and Z. Du, "Adaptive and Flexible Virtual Honeynet," in Proc. Int. Conf. Mobile, Secure, Program. Netw., Paris, France, vol. 9395, pp. 1-17, Jun. 2015.

[12] L. Spitzner, "The Honeynet Project: Trapping the Hackers," IEEE Security Privacy, vol. 1, no. 2, pp. 15-23, Mar. 2003.

[13] L. K. Yan, "Virtual Honeynets Revisited," in Proc. 6th Annu. IEEE SMC Inf. Assurance Workshop (IAW), pp. 232-239, Jun. 2005.

[14] F. Abbasi and R. Harris, "Experiences with a Generation III Virtual Honeynet," in Proc. Australas. Telecommun. Netw. Appl. Conf. (ATNAC), pp. 1-6, Nov. 2009.

[15] S. Godala and R.P.V. Vaddella, "A study on intrusion detection system in wireless sensor networks", International Journal of Communication Networks and Information Security 2020, vol. 12, pp. 127–141.

[16] C. Lei, H. Zhang, J. Tan, Y. Zhang, and X. Liu, "Moving Target Defense Techniques: A Survey," Security and Communication Networks, vol. 2018, pp. 1-25, Jul. 2018.

[17] L. Wang and D. Wu, "Moving Target Defense Against Network Reconnaissance with Software Defined Networking," In Bishop M., Nascimento A. (eds) Information Security. ISC 2016. Lecture Notes in Computer Science, vol 9866. Springer, Cham, 2016.

[18] T. Lengyel, J. Neumann, S. Maresca, and A. Kiayias, "Towards Hybrid Honeynets via Virtual Machine Introspection and Cloning," in Network and System Security (Lecture Notes in Computer Science), vol. 7873, J. Lopez, X. Huang, and R. Sandhu, Eds. Berlin, Germany: Springer, pp. 164-177, 2013.

[19] W. Han, Z. Zhao, A. Doupe, and G.-J. Ahn, "HoneyMix: Toward SDN-Based Intelligent Honeynet," in Proc. ACM Int. Workshop Secur. Softw. Defined Netw. Netw. Function Virtualization, New York, NY, USA, pp. 1-6, 2016.

[20] B. Park, S. P. Dang, S. Noh, J. Yi, and M. Park, "Dynamic Virtual Network Honeypot," 2019 International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Korea (South), pp. 375-377, 2019.

[21] S. Kyung et al., "HoneyProxy: Design and Implementation of Next-Generation Honeynet via SDN," in Proc. IEEE Conf. Commun. Netw. Secur. (CNS), pp. 1-9, Oct. 2017.

[22] W. Fan and D. Fernandez, "A Novel SDN-Based Stealthy TCP Connection Handover Mechanism for Hybrid Honeypot Systems," 2017 IEEE Conference on Network Softwarization (NetSoft), Bologna, pp. 1-9, 2017.

[23] https://www.openstack.org

[24] https://https://www.snort.org

[25] A. Gupta and L. Sharma, "Mitigation of DoS and Port Scan Attacks Using Snort," International Journal of Computer Sciences and Engineering, vol. 7, pp. 248-258, 2010.