Dynamic Defense Provision via Network Functions Virtualization

Younghee Park[†1], Pritesh Chandaliya[†2], Akshaya Muralidharan[†3], Nikash Kumar[†4], Hongxin Hu[§5]
[†] San Jose State University, [§] Clemson University
[1]younghee.park@sjsu.edu, {[2]priteshchandaliya,[3]aksh3001,[4]nikashedu}@gmail.com, [5]hongxih@clemson.edu

ABSTRACT
Network Function Virtualization (NFV) is a critical part of a new defense paradigm providing high flexibility at a lower cost through software-based virtual instances. Despite the promise of the NFV, the original Intrusion Detection System (IDS) designed for NFV still draws heavily on processing power and requires significant CPU resources. In this paper, we provide a framework for dynamic defense provision by building in light intrusion detection network functions (NFs) over NFV. Without using the existing IDSes, our system constructs a light intrusion detection system by using a chain of network functions in NFV. The entire IDS is broken down into separate light network functions according to different protocols. The intrusion detection NFs cover various protocol stacks from the link layer to the application layer protocols. They also include different deep packet inspection NFs for different application layer protocols. The experimental results show the proposed system reduces resource consumption while performing valid intrusion detection functions.

Keywords
Network Functions Virtualization; Software-Defined Networks; Security; Network attacks

1. INTRODUCTION
Since middleboxes share expensive and proprietary intrusion detection systems at fixed locations across the network, their limitations include inscalabiltiy, inflexibility, and high cost. Network Function Virtualization (NFV) has decoupled network functions (NFs), such as the firewall, load balancer, NAT, and web proxy, from dedicated hardware and has implemented them as pure software instances on industrial standard high-volume servers and storage [2, 8, 9, 18].

Some research has focused on NFV performance issues [2, 4, 5, 9], while other research has proposed various defense systems using NFV and Software-Defined Networks (SDN) [1, 3, 6, 12, 13, 14, 15]. Anat Bremler-Barr, et. al. have proposed a framework using NFV to provide a deep packet inspection service [17]. Bohatei [3] and VNGuard [1] have proposed a defense mechanism utilizing traditional intrusion detection systems like Bro and firewalls based on NFV and SDN for a highly scalable and flexible defense. However, no effort has been made to construct built-in light intrusion detection NFs to leverage the virtualization benefits of NFV.

*Dr.Park, the first author, is a corresponding author.

SDN-NFV Sec'17, March 22-24, 2017, Scottsdale, AZ, USA

DOI: http://dx.doi.org/10.1145/3040992.3041005

In this paper, we propose a dynamic defense provision framework based on NFV and SDN. It largely consists of a dynamic network function virtualization system to provide various intrusion detection network functions with an SDN controller to monitor and deploy NFV in the network. There are different components to provide dynamic defense methods using NFV: a traffic classifier, a pool of network functions, a service chaining module, and a virtual machine manager. Instead of instantiating the existing bulky IDS like Bro for NFV, our system implements various small intrusion detection NFs and instantiates only a light chain of necessary network functions to detect and investigate traffic according to traffic types. These network functions cover various actions for simple header information checking as well as for deep packet inspection for each protocol. The SDN controller orchestrates intrusion detection network services by managing network functions while monitoring the entire network topology.

This paper contributes to building on fine-grained network functions for intrusion detection on top of virtualization for the dissemination of a practical lightweight IDS. To achieve our ultimate goal, we first implement a number of small intrusion detection NFs by breaking down an entire complicated IDS into a small set of network functions. Second, the proposed system enables us to dynamically invoke a chain of the light intrusion detection NFs depending on the traffic characteristics. Lastly, we implement our proposed system based on ClickOS [32]. In our experiments, we demonstrate that our system can detect and examine malicious packets by using a chain of light network functions.

The rest of the paper is organized as follows. Section 2 describes our proposed system architecture. Section 3 explains our system implementation and discusses experimental results. Section 3 addresses future work, and we present our conclusion in Section 4.

2. System Architecture
This section describes our proposed system architecture for a dynamic defense provision method based on NFV and SDN. The SDN monitors an entire network topology and determines the routing path for each flow that can be used to deploy necessary NFs from the source to the destination. Based on the routing path, a set of network functions is placed on the switches in order to defend against network attacks. We address the deployment model and our key idea to introduce the dynamic defense provision system.

Figure 1 shows our system architecture based on NFV and SDN controller. The SDN controller plays a role in orchestrating network intrusion detection services and determining the locations of services across the network while monitoring the entire topology. The NFV provides dynamic defense network services according to traffic types by instantiating a set of network functions in the switches.

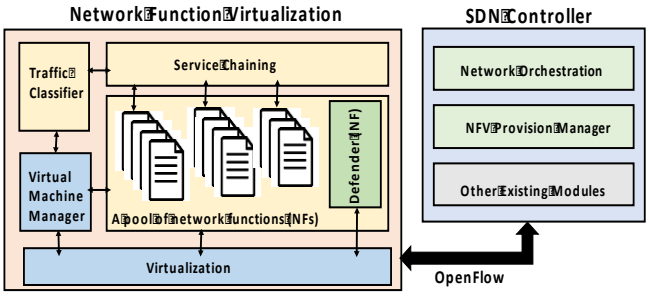

Figure 1. The proposed system architecture

2.1 Dynamic NFV Deployment and Orchestration

The SDN controller in our system orchestrates network services by using a routing algorithm to steer traffic. It determines where to deploy NFV and manages it remotely. It is a challenge to determine the optimal place and to deploy the NFV in order to defend against network attacks [1, 3]; however, the rule of thumb is that traffic must be examined in the routing path to detect malicious traffic.

We use the Dijkstra algorithm to determine the placement of NFV services in the network, since it is currently used in OpenDaylight, an industry-sponsored SDN platform [7, 16, 19, 10]. Based on this shortest path algorithm, the SDN controller decides the routing path for each flow and monitors the entire network topology in the proposed system. Based on the results from the shortest path, the SDN controller pushes flow rules into switches based on the results of the routing algorithm. Accordingly, network functions are virtualized on the designated paths to examine traffic that traverses the network. Our system flexibly changes the placement of NFV according to another routing algorithm that users can define themselves to send their traffic efficiently. However, our NFV deployment principle is based on one routing algorithm.

NFV services are determined by traffic characteristics in the traffic classifier shown in Figure 1. The SDN controller can easily identify traffic characteristics by using existing modules in the SDN controller: all initial flows are diverted to the SDN controller, which decides the best routing path by pushing flow rules. The traffic classifier also determines a set of network functions to examine flow on the designated routing path for network orchestration services. We utilize protocols (P) and port numbers (N) to determine a set of network functions in the switches. The result of the traffic classification allows NFV to dynamically instantiate network functions virtually, as explained in the next section.

2.2 Dynamic Defense Provision in NFV

We present a dynamic defense provision method based on NFV, as shown in Figure 1. Our method consists of a traffic classifier, a virtual machine manager, and a service chaining module to instantiate network functions. The basic components can be defined as:

- $P = \{tcp, udp, icmp, arp, http, \dots\}$ where P is a finite set of protocol types in the traffic.
- $N = \{n_1, n_2, \dots, n_k\}$ where N is a finite set of port numbers that the traffic uses. Note that $k = \{1, 2, \dots, n\}$.
- $V = \{v_1, v_2, \dots, v_k\}$ where V is a finite set of network functions (NF). Note that $k = \{1, 2, \dots, n\}$.
- $S = < \overline{V}, p, t >$ where S is a service chain with three components. Note that $\overline{V}$ is a subset of V ($\overline{V} \subseteq V$) with $p \in P$ and $t \in T$.

The traffic classifier identifies traffic characteristics based on the protocol and port number used to determine a set of network functions for a service chain. Based on these two important criteria, the traffic classifier determines applications to characterize traffic. For example, traffic related to HTTP uses TCP and port number 80.

The service pool has two different subsets of network functions: packet header inspection and deep packet inspection. First, it includes various network functions to inspect packet header information depending on protocols for TCP, UDP, ICMP and ARP from the link layer to the transport layer. Second, it selects different types of deep packet inspection (DPI) NFs depending on application layer protocols including MQTT [10, 24], ZMTP [25], HTTP. These network functions have different DPI functions in order to check packet payloads according to the standard application protocol specification. This aims to determine the maliciousness of traffic based on the application protocol specification since malware is often hidden inside payloads. Note that current network functions for different protocols include basic and essential functions to defend against simple network attacks, such as spoofing attacks and denial-of-service attacks for each protocol level and simple deep packet inspection against malware by checking packet formats. Our goal is to present an NFV-based dynamic defense provision system with a lot of small light network functions without the support of any existing IDS.

Service chaining instantiates a set of network functions through a virtual machine manager. The traffic classifier determines a used protocol P and a used port number N and an application. Service chaining instantiates a set of network functions where, for example, $S = <\overline{V}, p, t > where\ \overline{V} = \{v_1, v_2, v_4, v_6, v_8\}$. For example, for $P = \{tcp\}$ and $N=\{80\}$, and $V = \{v_1, v_2, v_4, v_6, v_8\}$, v_1 can be the traffic classifier, v_2 can be the IP header analyzer, v_4 can be the TCP header analyzer, and v_6 can be the HTTP-based DPI function. v_6 is a network function for packet drops, called a defender.

The virtual machine manager generates this service chain to examine the traffic from the source to the destination while the network functions for the traffic classifier and the defender must be always run. This set of network functions is dynamically called upon depending on traffic characteristics and types. This method reduces cost to dynamically generate virtualized network functions, since we customize the traditional IDS system by breaking it down into specific small intrusion detection network functions. In addition, when malicious packets are detected through the set of the customized intrusion detection network functions, the defender shown in Figure 1 simply drops the uncovered malicious traffic. The network function for the packet drop that should always be instantiated by the virtual machine manager as a part of $\overline{V}$.

3. IMPLEMENTATION AND EVALUATION

We designed and implemented our proposed system based on ClickOS [18], an open source operating system for virtualizing

middleboxes and instantiating network functions, using hypervisors with functions running on separate VMs. We used Click version 2.0.1 and Open vSwitch version 2.3.90 based on Xen hypervisor version 4.4.1.

We used three machines for our experiments: one Intel core i7 2.8 GHz with 16GB RAM and two Intel core i5 with 4 GB RAM. The first machine includes our system framework to perform NFV according to traffic characteristics. The second two machines were used to generate source and destination traffic for our experiments. The Open vSwitch ran on each host. All network traffic passed through the testbed setup, where Click's primary domain was running along with the service-chaining instance. All packets were diverted to Click's instance, which was one of the DOMs in Xen hypervisor. The spawned instance was able to include different network functions, such as a traffic classifier, a DPI, and a defender. The service chaining module invoked a Click instance using a common Xen configuration file.

We discuss the experimental results for the proposed system in terms of start-up times to instantiate network functions and memory usage to run them, comparing them to the required resources for Snort, a signature-based open source IDS.

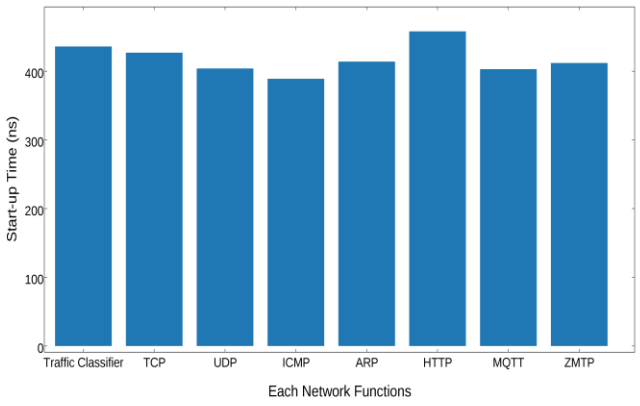

Figure 2. An average start-up time to spawn each network function

Start-up Time for Network Functions: Figure 2 shows the time to instantiate each network function for the traffic classifier, different protocol header analyzers, and different DPI functions for different protocols. Click instances represent the specific set of network functions, which were booted over Xen hypervisor. Time to spawn an instance includes the creation of the Xen DOM and the starting time of the set of network functions. As a result, we show that the time to spawn one instance with the network functions in ClickOS requires a very short time, less than 0.1 second, which is negligible value compared to the typical DOM-0 booting time in Xen, which is several seconds. VNGuard states that a virtual network function instance created in ClickOS can be booted within 30 milliseconds [1] on their system specification. In our system, to make a chain of network functions for a service, we expect that an instance can be invoked within less than a millisecond even though it might be different from the number of network functions instantiated.

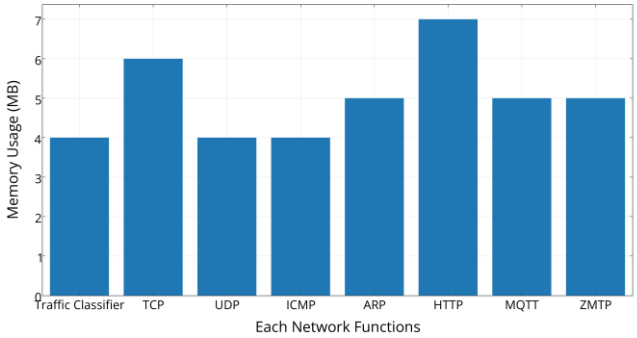

Figure 3. A maximum memory allocated to a new instance

Memory Usage for Network Functions: Figure 3 shows the maximum memory allocated to a new instance for each network function service. Memory allocation depends on the initial configuration (i.e. Xen configuration file) and the Click elements used for implementing a network function. We instantiated a Click instance using ClickOS where each instance was allocated around 1GB-2GB of RAM and 1 CPU. We excluded memory usage for both the privileged domain and the unprivileged domain. As shown in Figure 3, memory consumption for each network functions was generally less than 7 MB. This constitutes very small memory usage compared to the typical guest OS on Xen hypervisor. In our experiment, we consumed 128MB for ClickOS on Xen.

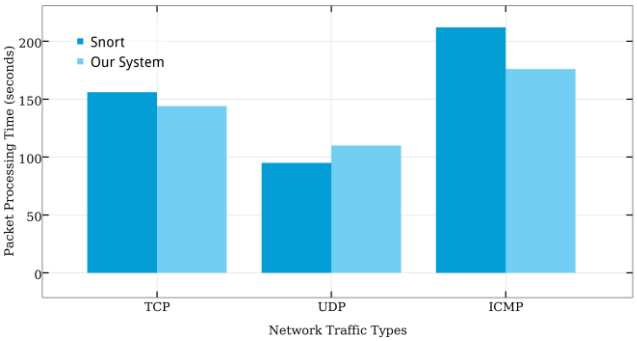

Figure 4. An average start-up time between Snort and our proposed system

Traffic Processing Time: Figure 4 compares processing time, and we evaluated memory usage for our proposed system architecture and Snort version 2.9.6.0 for different traffic types. We generated invalid or malicious TCP, UDP, and ICMP packets from a particular source and destination address pair using Scapy, an interactive packet manipulation tool [26]. For our proposed system, we included the traffic classifier, the defender, and various NFs for different traffic types and application protocols. For Snort, we used Snort itself without Xen in our testbed to perform intrusion detection actions. If malicious packets were detected in our system or Snort, we dropped the packets. We showed the time in order to process 100 incoming packets. Both our system and Snort detected 100%, and our system dropped 100% packets successfully. Our results showed that the processing time to check 100 packets for our proposed system is similar to Snort processing time but that our system used significantly less memory than Snort. Snort used 112 MB

memory, but our light IDS in NFV used less than around 30 MB that might be different from the number of network functions used. Our system requires approximately 30% less memory than Snort. The results demonstrated that our proposed system can efficiently achieve the same IDS functions using fewer resources in terms of processing time and required memory usage.

4. DISCUSSION AND FUTURE WORK

We developed a framework to dynamically provide network functions appropriate to different traffic. One noteworthy advantage of the proposed system architecture is that it includes customized processing functionalities relying only on existing elements of ClickOS virtualized middleboxes, without a resource intensive IDS like Bro or Snort. Previous research has utilized existing IDS systems like Bro in Bohatei [3]. However, our proposed system makes full use of all available elements in *Click* to develop a lightweight IDS system for each traffic type instead of using the existing IDSes. This is beneficial in spawning only the necessary NF services instead of continuously running Bro in NFV. We thus realize resource savings while dynamically providing each service.

The proposed system demonstrates the feasibility of building an internal IDS with built-in elements in Click. The proposed system covers a core set of protocols to investigate traffic from the link layer to the application layer. The current system in this paper includes only a few of the network functions for each protocol; however, we can easily extend it to more complicated network functions, especially DPI network functions for different protocols. We can also build more new NFs for different protocols, such as DNS, FTP, SMTP, and other IoT protocols in order to cover diverse protocol stacks. Our goal in this paper is to provide a framework for a dynamic NFV-based defense with multiple independent light network functions for intrusion detection. In future work, we will develop a complete system by including the full array of network functions for intrusion detection in order to cover well-known attacks and as yet unknown attacks without the support of the existing IDSes.

5. CONCLUSION

This paper has proposed a dynamic defense provision framework utilizing NFV with SDN. Without relying on existing intrusion detection systems, this paper has proposed a lightweight intrusion detection system based on separately available or new network functions based on Click. Depending on traffic characteristics, the proposed system dynamically creates a chain of network functions to investigate traffic that passes through the routing path in the network. Thus, our method does not require conventional approaches to redirect traffic to designated NFV placement. Our method includes different deep packet inspection NFs for different protocols in order to achieve lightweight DPI NFs by breaking down the entire DPI system into a small set of DPI functions according to different protocols. This separation of network functions for intrusion detection achieves significant resource savings while also avoiding a single point of failure.

6. REFERENCES

[1] Juan Deng et al., "VNGuard: An NFV/SDN combination framework for provisioning and managing virtual firewalls," Network Function Virtualization and Software Defined Network (NFV-SDN), 2015 IEEE Conference on, San Francisco, CA, 2015, pp. 107-114.

[2] Battula, L.R., "Network Security Function Virtualization(NSFV) towards Cloud computing with NFV Over Openflow infrastructure: Challenges and novel approaches," in Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on , vol., no., pp.1622-1628, 24-27 Sept. 2014.

[3] Seyed K. Fayaz and Yoshiaki Tobioka and Vyas Sekar and Michael Bailey, “Bohatei: Flexible and Elastic DDoS Defense”, 2015 24th USENIX Security Symposium (USENIX Security 15).

[4] Taekhee Kim, Taehwan Koo, and Eunkyoung Paik, “SDN and NFV Benchmarking for Performance and Reliability”, 2015 Asia-Pacific Network Operations and Management Symposium (APNOMS).

[5] T. Wood, K. K. Ramakrishnan, Jinho Hwang, G. Liu and Wei Zhang, "Toward a software-based network: integrating software defined networking and network function virtualization," in IEEE Network, vol. 29, no. 3, pp. 36-41, May-June 2015.

[6] H. Hu, W. Han, G. Ahn, and Z. Zhao, “FlowGuard: building robust firewalls for software-defined network,” in HotSDN’14, Chicago, IL,USA, 2014.

[7] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. De Turck and R. Boutaba, "Network Function Virtualization: State-of-the-Art and Research Challenges," in IEEE Communications Surveys & Tutorials, vol. 18, no. 1, pp. 236-262, Firstquarter 2016.

[8] Attila Csoma, Balázs Sonkoly, Levente Csikor, Felicián Németh, Andràs Gulyas, Wouter Tavernier, and Sahel Sahhaf. 2014. ESCAPE: extensible service chain prototyping environment using mininet, click, NETCONF and POX. In Proceedings of the 2014 ACM conference on SIGCOMM (SIGCOMM '14). ACM, New York, NY, USA.

[9] Yong Li and Min Chen, "Software-Defined Network Function Virtualization: A Survey," in IEEE Access, vol. 3, no. , pp. 2542-2553, 2015.

[10] Konglong Tang, Yong Wang, Hao Liu, Yanxiu Sheng, Xi Wang, Zhiqiang Wei, “Design and Implementation of Push Notification System Based on the MQTT Protocol,” International Conference on Information Science and Computer Applications, 2013.

[11] Taekhee Kim, Taehwan Koo, and Eunkyoung Paik, “SDN and NFV Benchmarking for Performance and Reliability”, 2015 Asia-Pacific Network Operations and Management Symposium (APNOMS).

[12] Y. Ben-Itzhak, K. Barabash, R. Cohen, A. Levin and E. Raichstein, "EnforSDN: Network policies enforcement with SDN," 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, ON, 2015, pp. 80-88.

[13] M. Vijayalakshmi, S. Mercy Shalinie and A. Arun Pragash, "IP traceback system for network and application layer attacks," Recent Trends In Information Technology (ICRTIT), 2012 International Conference on, Chennai, Tamil Nadu, 2012, pp. 439-444.

[14] W. Kinney, "Protecting against application DDoS attacks with BIG-IP ASM: A Three-Step solution," 2012.

[15] . Ranjan, R. Swaminathan, M. Uysal, A. Nucci and E. Knightly, "DDoS-Shield: DDoS-Resilient Scheduling to Counter Application Layer Attacks," in IEEE/ACM Transactions on Networking, vol. 17, no. 1, pp. 26-39, Feb. 2009.

[16] A. Gember, R. Grandl, A. Anand, T. Benson, and A. Akella. Stratos: Virtual Middleboxes as First-Class Entities. Technical Report TR1771, University of Wisconsin-Madison, June 2012.

[17] Anat Bremler-Barr, Yotam Harchol, David Hay, and Yaron Koral, “Deep Packet Inspection as a Service,” In Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies (CoNEXT), NY, USA, 271-282, 2014.

[18] Joao Martins, Mohamed Ahmed, Costin Raiciu, Vladimir Olteanu, Michio Honda, Roberto Bifulco, and Felipe Huici, “ClickOS and the art of network function virtualization.” In Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation (NSDI). USENIX Association, Berkeley, CA, USA, 459-473, 2014.

[19] OpenDaylight project. http://www.opendaylight.org