Security Service Function Chain based on Software-Defined Security

Sheng Zhu¹, Cheng Zhou¹, Cheng Wang¹

¹Global Energy Interconnection Research Institute, Nanjing 210003, China
*Corresponding author’s e-mail: 86503739@qq.com

Abstract. Software-defined networks face not only traditional network security threats, but also new network security threats brought by centralized and programmable control planes. To address the above problems, this paper proposes a business chain design mechanism for security protection functions based on software-defined security, which constructs a security service function chain according to user requirements and also realizes security resource scheduling according to the load of hosts, and the security service function chain rules classify and redirect network flows to corresponding security devices sequentially in the form of OpenFlow flow tables to realize network traffic Dynamic control of network traffic. The test shows that this mechanism can effectively realize security service virtualization and provide flexible and dynamic security protection mechanism according to security service requirements.

1. Introduction

With the advent of the mobile Internet era, the rapid landing of cloud computing applications and the upcoming Internet of Things era, the Internet has long penetrated into various industry sectors and lives such as people's livelihood, transportation, healthcare, finance and education. The traditional network security service function chain has revealed more and more drawbacks[1], mainly in two aspects: on the one hand, it is coupled with the network topology, which makes deployment difficult and management control not easy; on the other hand, when the service function chain needs to be expanded or changed, the physical topology of the network needs to be changed and the configuration of the network equipment needs to be updated, which increases the investment of human and material resources and is not easy to operate and manage. Traditional security service function provisioning is increasingly difficult to adapt to the increasingly complex needs of virtualized network security protection. In order to solve the drawbacks of the traditional network security function service approach, it is urgent to study a new dynamic and flexible security service function chain mechanism with scalability.

The primary task is to address the redesign and deployment of network devices and security devices in the cloud computing environment, so that the new security service function chain has dynamic and flexible customizability[2]. The OpenFlow-based SDN technology emerged, which is based on Ethan[3] as a prototype of the network architecture and refers to the SANE[4] and 4D[5] architectures, separating the control logic layer from the data forwarding layer in the traditional network architecture, and programming the switching devices with standard programming interfaces through controllers with logical control at the control logic layer In the control logic layer, the controller with logical control programs the switching equipment with standard programming interface to realize the control of the data forwarding layer. The applications written on the controller according
to the security business logic requirements can flexibly and dynamically redirect the network data flow to each security function service node in turn to realize the forwarding behavior of the network, so as to achieve the purpose of network data flow detection and protection.

The SDN architecture, with its separation of the logical and physical planes, open API interfaces, and software-based programming, received widespread attention and was subsequently extended to other domains, resulting in a series of concepts such as software-defined storage and software-defined systems, and the application of these features of network evolution to the security domain, thus giving rise to the concept of software-defined security [6]. 2012 Gartner proposed the SDS architecture [7], in which the security logical control function is extracted from the traditional security architecture to form a security control plane and a security data plane, decoupling network security devices from their access and deployment methods, virtualizing security devices through NFV technology to form a security resource pool on the security data plane, and automating management on the security control plane through a software-defined The security data plane is virtualized to form a security resource pool through NFV technology, and the security control plane is managed automatically through a software-defined approach, ultimately realizing flexible and efficient development and deployment of security business requirements. In the virtual network environment, the realization of dynamic, flexible and on-demand security protection mechanisms requires consideration of the following points.

(1) How to achieve flexible deployment of resources without changing the existing network underlying physical connectivity and existing service configuration.

(2) Make full use of the virtualization characteristics of the network to achieve dynamic scheduling of security services.

(3) Build a logical security service function chain design and realize the mapping of logical security service function chain to physical forwarding path.

(4) Improving the efficiency of virtual security resource scheduling for security service function chains and improving network resource utilization. In summary, this paper will study the dynamic scheduling and deployment of security service function chains in virtual network environments based on the service function chain framework of SDS, mainly discussing the construction of security service function chains, scheduling of virtual security devices and traffic scheduling aspects to achieve a dynamic, on-demand security protection mechanism.

2. Background

2.1. Software Defined Networking
SDN architecture is proposed in a white paper [8] provided by ONF (Open Networking Foundation), which is a network architecture that separates the logical control plane from the data forwarding plane in the traditional network architecture. The logical control function of the control plane is realized by the controller, and the SDN switch realizes the data forwarding function of the data plane. This network architecture reduces the burden on the data plane by means of a logically centralized SDN controller and an open OpenFlow standard interface, and the controller realizes flexible and fast forwarding behaviour in a software-defined manner according to the business logic requirements, thus improving. The diagram of SDN network architecture is shown in Figure 1, which shows that the architecture of SDN can be divided into three layers from bottom to top: infrastructure layer, control layer, and application layer [9]. Infrastructure layer: it is at the bottom of the whole architecture and consists of physical or virtual switches, routers, and other network components, which can be managed and controlled by the SDN controller to realize the network forwarding and other functions in the data plane, and the network components can communicate with the controller through SSL secure connection channels. The OpenFlow protocol is the most common and standard protocol for communication and interaction between the controller and data plane devices. Control layer: the control layer consists of software-based SDN controllers, which provide network control management and monitor network forwarding functions, and provide programming interfaces for communication between controllers and network devices and between controllers and controllers. The application
layer consists of one or more end-user applications (security applications, visualization applications, etc.). The application layer interacts with the control layer through the application programming interface (i.e., the northbound interface), and users can realize customized management of application requirements through the northbound interface. This enables automation and intelligence of network control and management.

2.2. Software-Defined Security
The software-defined security architecture system decouples the control plane of security from the data plane, and the architecture is shown in Figure 2 and can be divided into three parts: a security resource pool to implement security functions, a software-defined security control platform, and a security application.

Security resource pool: through security capability abstraction and resource pooling, security devices are abstracted into a resource pool with security capabilities to provide basic security protection capabilities;
Security control layer: the security control platform provides programmable APIs for the security application layer in the north direction, provides pooled management of security device resources in the south direction, and adapts to different business management platforms in the east-west direction, such as infrastructure management platforms;

Security application Interface: located at the top layer of the architecture, pushing user-defined security requirements to the security control layer and changing the traditional offline transaction model.

3. Overall framework
Before you begin to format your paper, first write and save the content as a separate text file. Complete all content and organizational editing before formatting. Please note sections A-D below for more information on proofreading, spelling and grammar.

3.1. SDS-based security service function chain architecture
The SDS-based security service function chain architecture can be divided into four parts: a security application interface, a security resource pool that implements security functions, a software-defined security controller, and a software-defined SDN controller, as shown in Figure 3.

Security application interface: located at the top layer of the architecture, pushing user-defined security requirements to the security controller.

Security resource pool: consists of security protection devices and virtualized security protection devices, which are abstracted into a resource pool with security capabilities through security capability abstraction and resource pooling to provide basic security protection capabilities.

Security controller: the security controller interacts with security applications in the north direction for data and security requirements; the south direction provides management of registration and scheduling of basic security protection components; the west direction is docked with the SDN controller to generate the required logical topology and scheduling instructions for data flow; and

SDN Controller: Maintains the network-wide view, monitors the network-wide topology, and implements the function of network traffic redirection according to the flow instructions delivered by the security controller.

3.2. Security Resource Scheduler
Security resource pooling is provided by Security Fabric Manager Scheduler (SFMS), a security resource pooling management scheduler, to provide the virtual security appliance instance lifecycle operation interface, including: start instance, stop instance, restart instance, delete instance, and other operations. In terms of the business requirements of the security resource pool, the most important
business is to start the virtual security appliance for users. In order to make full use of the resources of the hosts in the security resource pool and improve load balancing, three resource metrics (memory, CPU and disk space) are considered in a comprehensive scheduling algorithm when starting the virtual security appliance.

Scheduling is divided into two steps: first, the nodes that do not meet the requirements are filtered out; then the remaining nodes are calculated and the optimal scheduling node is selected.

(1) Filter nodes: compare whether the free memory, CPU, and disk space of each host node is greater than the memory, CPU, and disk space occupied by the virtual security device to be started, and filter out the host node if one of the metrics does not meet the requirements.

(2) Calculation phase: Calculate the normalized utility value of the remaining host resource metrics.

1) $MEM_{max}$ denotes the maximum free memory value among the remaining hosts, $MEM_{min}$ denotes the minimum free memory value among the remaining hosts, and the memory normalized utility value $U_m$ is calculated for a host with free memory $X$ as shown in Equation (1).

$$U_m = \frac{(X - MEM_{min})}{(MEM_{max} - MEM_{min})}$$  \hspace{1cm} (1)

2) $CPU_{max}$ denotes the maximum number of idle CPUs in the remaining hosts, $CPU_{min}$ denotes the minimum number of idle CPUs in the remaining hosts, and the CPU normalized utility value $U_c$ for a host with $Y$ idle CPUs is calculated as shown in Equation (2).

$$U_c = \frac{(Y - CPU_{min})}{(CPU_{max} - CPU_{min})}$$  \hspace{1cm} (2)

3) $DISK_{max}$ denotes the maximum free disk space value among the remaining hosts, and $DISK_{min}$ denotes the minimum free disk space value among the remaining hosts. The normalized utility value $U_d$ of the disk space of a host with free disk space $Z$ is calculated by Equation (3).

$$U_d = \frac{(Z - DISK_{min})}{(DISK_{max} - DISK_{min})}$$  \hspace{1cm} (3)

4) The utility value of the combined three resource metrics, the combined utility value $U$ of the host node is calculated as shown in Equation (4).

$$U = U_m W_1 + U_c W_2 + U_d W_3$$  \hspace{1cm} (4)

Where $W_1$, $W_2$, and $W_3$ represent the corresponding weights of the three resource metrics (memory, CPU and disk space) and $W_1 + W_2 + W_3 = 1$. The node with the largest $U$ value is the optimal scheduling node to be selected.

4. Experiment and Conclusion

In order to verify the flexibility and effectiveness of the software-defined security-based service function chain for security protection in the virtual environment proposed in this paper, a topology was built on the physical server with SSFC1, with the SDN controller (OpenDayLight) and security controller activated in the topmost virtual machine in the figure, Classifier, SFF1, and SFF2 as OpenvSwitch switches, VSD1 and VSD2 as virtual security appliances Firewall and IPS respectively. Classifier, SFF1, and SFF2 are OpenvSwitch switches, and VSD1 and VSD2 are virtual security devices Firewall and IPS respectively. H1 communicates with H2, requiring the communication process to pass through the virtual security device VSD1 (Firewall), VSD2 (IPS) in turn, the IP address to be configured for the virtual security device VSD1 (Firewall) is 192.168.1.30, and the IP address to be configured for the virtual security device VSD2 (IPS) is 192.168.1.30. VSD2 (IPS) will be configured with an IP address of 192.168.1.40.
The running environment and configuration of the experiment are shown in Table 4. There are five virtual machines, two of which are virtual security appliances VSDs, one running OpenDayLight (ODL) controller and security controller, and the ODL controller is connected to three virtual switches (OpenvSwitch) of Classifier/SFF1/SFF2; the other two are simulated as virtual machines for two users.

After enabling the controller and requesting a security service function chain, it was verified by (1) that the virtual security devices VSD1 (Firewall) and VSD2 (IPS) were up. Then, the communication from the H1 end of the network to the H2 end was tested to be connected using the ping command; then, the flow tables in the virtual switches Classifier, SFF1, and SFF2 were viewed to analyze the flow of data and to determine the change of data in the messages by capturing packets.

The flow direction of the data stream from H1 to H2 is shown in the steps 1 to 6 in Figure 4, and the experimental data are as follows.

Step 1: The packet from H1 arrives at Classifier, the packet is encapsulated with NSH and exits from port 2. The flow table is as follows.

Step 2: The data stream reaches SFF1 and matches the following flow table, nsp=44, nsi=255, destination address=192.168.1.30, out from port 1.

Step 3: VSD1 processes the packet, nsp=44, nsi=254, and then forwards the data packet to SFF1.

Step 4: The data packet matches the following flow table, nsp=44, nsi=254, out of port 1.

Step 5: The data packet matches the following flow table, nsp=44, nsi=254, destination address=192.168.1.40, out from port 1.

Step 6: VSD2 processes the packet, nsp=44, nsi=253, and then forwards the data packet to SFF2.
The design of the service function chain based on software-defined security is proven feasible through experiments. In the virtual network environment, the data flow is made to pass through each virtual security device in an orderly sequence according to the user's security requirements, and the dynamic control of network traffic is realized.

Acknowledgments
Supported by the science and technology project of State Grid Corporation of China: "Research on The Security Protection Technology for Internal and External Boundary of State Grid information network Based on Software Defined Security" (No. 5700-202058191A-0-0-00);

References
[1] A. M. Medhat, T. Taleb, A. Elmangoush, G. et al. Service Function Chaining in Next Generation Networks: State of the Art and Research Challenges. IEEE Communications Magazine, vol. 55, no. 2, pp. 216-223, February 2017
[2] L. Yang, Y. Song, S. Gao, et al. Griffin: An Ensemble of AutoEncoders for Anomaly Traffic Detection in SDN. GLOBECOM 2020 - 2020 IEEE Global Communications Conference, 2020, pp. 1-6.
[3] Mingyuan Z, Wei W U, Yubo S, et al. Security level evaluation system for wireless local area network access devices. Journal of Tsinghua University (Science and Technology), 2020, 60(5): 371-379.
[4] Casado M, Garfinkel T, Akella A, et al. SANE: A Protection Architecture for Enterprise Networks. USENIX Security Symposium. 2006, 49: 50.
[5] Greenberg A, Hjalmtysson G, Maltz D A, et al. A clean slate 4D approach to network control and management. ACM SIGCOMM Computer Communication Review, 2005, 35(5): 41-54.
[6] Yubo S, Ming F A N, Junjie Y, et al. Multipath Solution and Blocking Method of Network Attack Traffic Based on Topology Analysis. Netinfo Security, 2020, 20(3): 9.
[7] Al-Ayyoub M, Jararweh Y, Benkhelifa E, et al. Sdsecurity: A software defined security experimental framework. 2015 IEEE international conference on communication workshop (ICCW). IEEE, 2015: 1871-1876.
[8] Kulkarni S G, Zhang W, Hwang J, et al. Nfvinice: Dynamic backpressure and scheduling for nfv service chains. IEEE/ACM Transactions on Networking, 2020, 28(2): 639-652.
[9] Wang S, Cao H, Yang L. A Survey of Service Function Chains Orchestration in Data Center Networks. 2020 IEEE Globecom Workshops (GC Wkshps. IEEE, 2020: 1-6.