A Stateful Security Service Chaining for Mobile-Edge Computing

Guanwen Li*, Huachun Zhou, Guanglei Li, Bohao Feng, and Taixin Li
School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, China
{14120079, hchzhou, 15111035, 11111021, 14111040}@bjtu.edu.cn

Abstract

We see an increasing demand for user-defined security service in mobile networks and the trend is to deploy IT based services in a mobile-edge cloud in the future. However, it lacks a flexible and efficient architecture to provide dynamically changing security services for different users in the mobile-edge cloud. To address this problem, we purpose an architecture of mobile-edge stateful security service chaining, which achieves a scalable combination of various required security functions. In addition, we present the stateful service function proxy, aiming to support compatibility to traditional security functions and convert them into stateful functions to shorten the transmission time of packets. Much work has been done to implement the proof-of-concept testbed of the architecture, and experimental results verify its feasibility. Moreover, we compare the performance of the proxy in the stateful mode with that in the stateless mode, and the result proves that the proxy running in the stateful mode can significantly decrease the transmission time of packets and improve the forwarding efficiency.

Keywords: Mobile-Edge Computing, Network Security, Service Function Chaining, Stateful Security Function

1 Introduction

With the exploding growth of mobile devices, the global mobile traffic increases rapidly year by year. Cisco [4] concludes that the mobile data traffic grew 74% in 2015 and forecasts that it will increase nearly eight fold from 2016 to 2020. However, the traditional mobile network infrastructures cannot afford with high demanding requirements from such a large number of mobile users. Recently, many researchers and institutions have focused on addressing the contradiction between the user experiment and the user capacity in the mobile network. The European Telecommunications Standards Institute (ETSI) also proposes a network architecture named Mobile-Edge Computing (MEC) [16], which deploys IT based services at the network edge with cloud computing, to relieve the pressure of traditional mobile networks.

The existing researches on MEC pay attention to the designs of radio access procedure [15, 3, 14, 21, 1]. Unfortunately, there are few considerations in providing flexible and efficient security services in clouds to meet various security requirements for different mobile users.

Therefore, we purpose an architecture of mobile-edge stateful security service chaining in this paper. Referring to the idea of Service Function Chain (SFC) [8], this architecture takes existing network security functions into a service chaining in high-performance mobile-edge clouds in order to deploy these security service functions more efficiently and flexibly. The key challenges are how to steer the traffic in a standard SFC way and be compatible with traditional service function. Thus, we present the stateful security function proxy with dual functions. On one hand, the proxy can translate the Network

*Corresponding author: School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, 100044, china, Tel: +86-13011181332

Service Header (NSH) encapsulation (the standard routing method in SFC) [19] of packets to the IP encapsulation and vice versa. On the other hand, the proxy can achieve the conversion from a stateless security function to a stateful one. Due to the conversion, the transmission time of packets can be reduced apparently to improve the forwarding efficiency of the whole security service chaining. To verify the feasibility of the purposed architecture, we present an implementation of the data plane in our cloud environment, including packet routing with NSH encapsulation and the proxies for security functions in SFC. We also compare its transmission performance in stateful mode with that in stateless mode.

In generally, the advantages of the proposed architecture are as follows.

- Flexibly: with SFC, various security service functions can be combined in more flexible ways for user custom security requirements.
- Scalability: dynamically changing security functions are not easy with traditional network infrastructures, while it comes true in this architecture.
- Compatibility: in terms of stateful security function proxies, traditional security service functions can be applied here without any alterations.
- High-efficiency: achieving stateful security functions can save the transmission time of packets and improve forwarding efficiency.

The Contributions of this paper are as follows. We purpose an architecture in mobile-edge network to provide flexible and scalable security service chains for mobile users. The stateful security function proxy is presented to support traditional security functions in this architecture and converts them into stateful security functions. Much work has been done to implement a proof-of-concept on an Openstack testbed, and in terms of the experimental results, we verify the feasibility of the architecture and prove that the stateful security function proxy can improve the forwarding efficiency by achieving stateful security functions.

The rest of the paper is organized as follows: Section 2 describes the architecture design of mobile-edge stateful security service chaining. Section 3 presents a proof-of-concept implementation on an OpenStack testbed. And we verify the feasibility and evaluate the performance of the proxy in Section 4. Section 5 describes related work, and Section 6 concludes this paper.

2 The Architecture of Mobile-Edge Stateful Security Service Chaining

As shown in Figures 1, the purposed architecture is divided into two layers in a high-level, including the Service Function Layer and the Data Transmission Layer. All of the security functions with their responding proxies are deployed in the Service Function Layer, such as a firewall and an NAT. In this layer, a logical security service function chaining is shown apparently. On the other hand, all kinds of entities on the forwarding path are deployed in the Data Transmission Layer, including a mobile user, an AP, an access gateway, a service classifier, several service forwarders and the destination server. This layer presents a physical forwarding route for the flow. Besides, a control plane is needed but beyond the scope of this paper, which is responsibility to manage the forwarding policies for the classifier and service forwarders.

In the scenario shown in Figures 1, a mobile user wants to access a remote server on the Internet. At first, its wireless access process is finished by an AP in the mobile network. Then, the mobile-edge cloud or datacenter network creates an appropriate security access service chaining for this user. The followings will describe that how each element of the architecture works in detail.

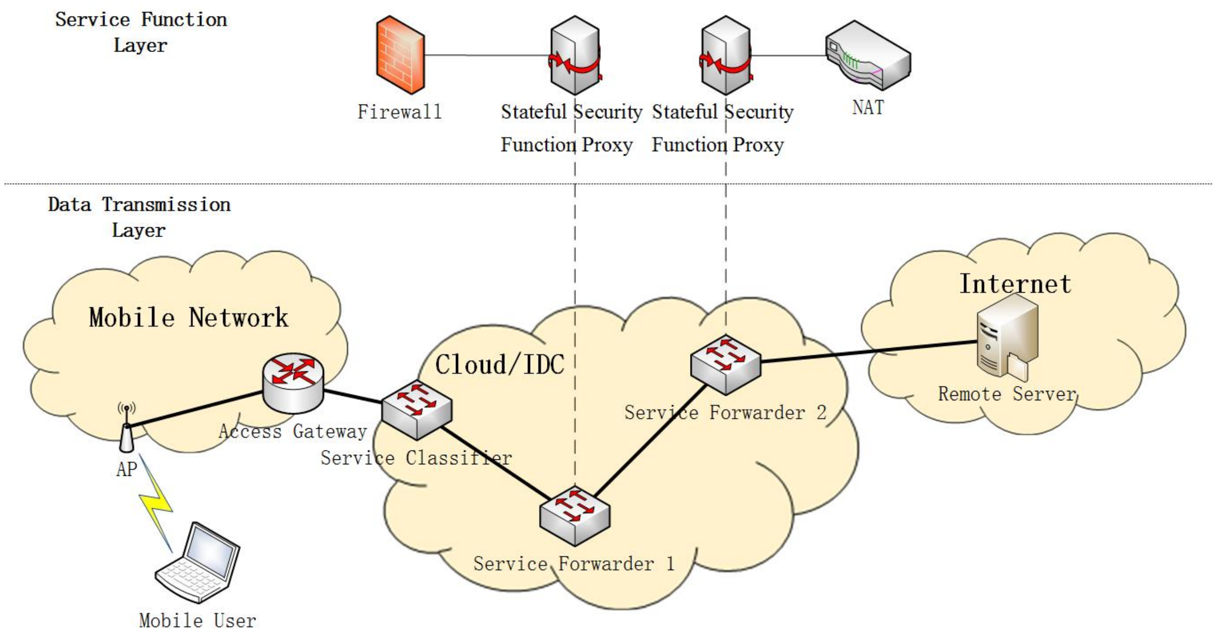


Figure 1: the architecture of mobile-edge stateful security service chaining

Access Gateway. In traditional network architecture, all of the network functions will be done in the mobile network by various equipments. However, in our purposed architecture, the mobile network is only responsibility to finish user access and authentication process. And then, the traffic is steered to an access gateway. The access gateway is a communication bridge between the user access point and the mobile-edge network (such as a cloud environment). It forwards all traffic from a mobile node to the cloud, which provides the security access service chaining on a virtualized platform.

Service Classifier. The traffic sent from the access gateway is received by the service classifier at first in the cloud. The classifier identifies different flows based on their identity/type information (e.g. 5-tuple in TCP/IP) and creates a unique flow identifier for each flow. Then, the classifier uses the corresponding flow identifier as its SPI (service path identifier) to mark a security service chaining for each flow. At the same time, the classifier sends classification results to the controller. Once the security service chaining is assigned, the classifier will forward the flow with its corresponding NSH encapsulation to next service forwarder.

Service Forwarder. Forwarding is the only one mission for a service forwarder, and it is bidirectional. On one hand, when a service forwarder receives a flow from previous service forwarder or the service classifier, it redirects the flow to specific stateful security function proxy according to the NSH encapsulation of the flow. On the other hand, it can also forward a flow back from the proxy to next proxy in the same or different service forwarder. The forwarding path is assigned by the controller.

Stateful Security Function Proxy. The stateful security function proxy plays an important role in this architecture. It has two main functions: first, to support compatibility to an existing traditional (SFC-unaware) security function; second, to convert a stateless security function into a stateful one.

Apparently, a traditional security function cannot recognize a NSH encapsulated flow, because it is designed based on IP network infrastructures. To let the flows received from the service forwarder be processed by a traditional security function, a NSH-IP translator is needed. The stateful security function proxy can be regarded as the translator and be deployed between a service forwarder and its security functions. Moreover, similar to the service forwarder, the stateful security function proxy is a bidirectional translator, which can translate NSH encapsulated flows into IP encapsulated flows and vice versa. Therefore, the proxy we purposed includes two kinds of interfaces: the forwarding interfaces and the service interfaces. The former achieves the function of NSH-IP translator, and the latter is used to communicate with the service function attached to the proxy.

flow ID	matching info	service state
1	(src_ip1, dst_ip1, protocol1, src_port1, dst_port1)	state1
2	(src_ip2, dst_ip2, protocol2, src_port2, dst_port2)	state2
...	...	...

Table 1: Service State List

The other significant function of the stateful security function proxy is to convert a stateless security function into a stateful one. To achieve this function, the proxy maintains a service state list and updates it in time. The service state list seen in Table 1 includes three columns: flow ID, matching info and service state. The flow ID is an identifier of the flow, and we use a 5-tuple of the flow as its matching info. The service state records how the security function processes the flow. When the proxy receives a new flow, it redirects the flow to the specified security function and adds the identification information of the flow to the service state list, including the flow ID and the matching info of the flow. Once the proxy gets the return flow or a message from the security function, it will record the service state of the flow. Furthermore, there is a timer for each flow to count the return time or delivery time of the message. If the threshold time is met, the corresponding service state will be written as “deny” automatically. Then, according to the service state, the proxy can process following packets of the flow directly rather than forward them to the security function.

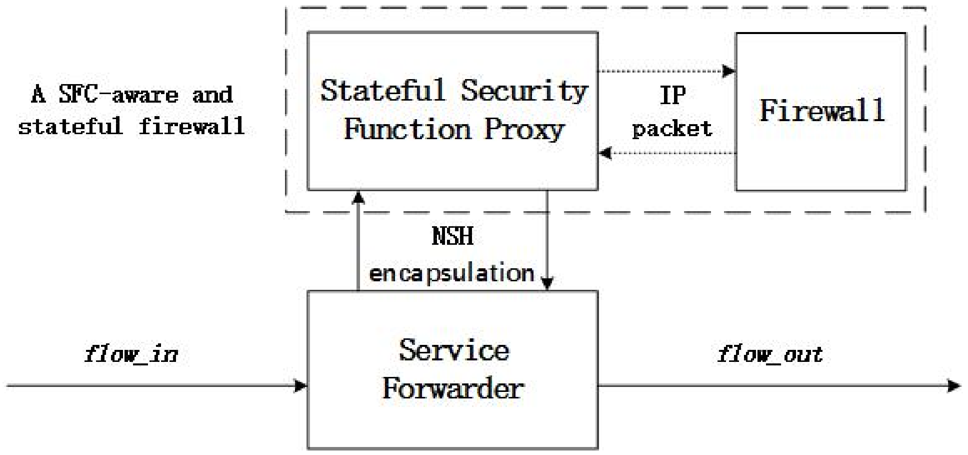


Figure 2: an example for achieving a stateful and SFC-aware firewall

The workflow of the stateful security function proxy is illustrated by an example of achieving a stateful and SFC-aware firewall. Figures 2 shows how this stateless firewall is served as a stateful and SFC-aware firewall with the proxy.

When the head packet of a new flow arrives at the proxy, its NSH header is de-encapsulated at first. After that, the proxy sends it to its attaching firewall immediately. At the same time, the proxy writes the 5-tuple information of the flow to its service state list and launches a timer for this flow. If the firewall allows this flow pass through according to a pre-defined rule, the head packet will return to the proxy in time. And then, the proxy can record the service state of the flow as “allow”. Otherwise, the service state will be written as “deny” when time is over. Once the service state of the flow is filled in, the proxy can process the following packets of the flow independently. In other word, the following packets of the flow are not required to be forwarded to the firewall any more. Finally, if the flow is allowed to be forwarded to next service funcion, the proxy will restore the NSH encapsulation of the flow before sends it out. Therefore, cooperated with the stateful security function proxy, the traditional stateless firewall can be regarded as a stateful and SFC-aware firewall.

Besides, there is a one-to-one correspondence between a security function and its stateful security function proxy. Thus, there may be several proxies connecting to the same service forwarder.

3 The Implementation of Testbed

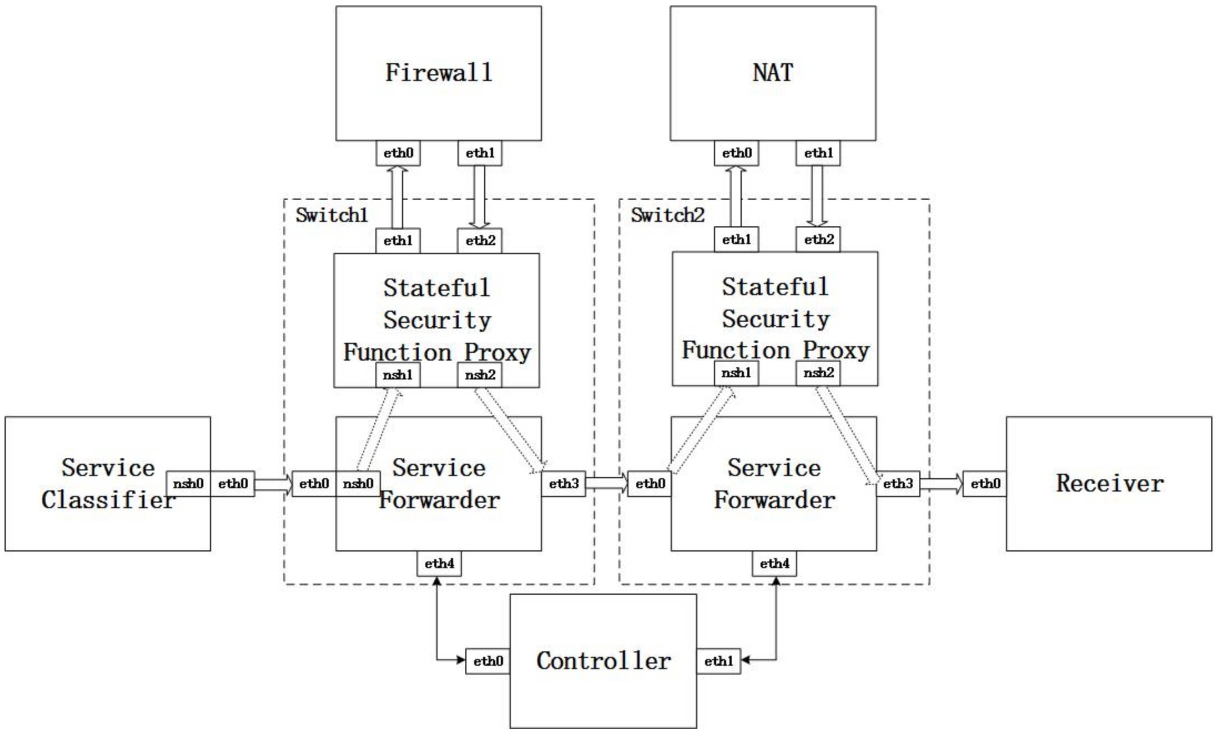


Figure 3: a simplified testbed of mobile-edge stateful security service chaining

To verify the feasibility of the purposed architecture, we simplify the architecture and present a testbed as shown in Figures 3.

The testbed of the implementation consists of a service classifier, a receiver, a controller, a firewall, an NAT and two switches. Each of them runs on a VM with Dual-core CPU, 2GB Memories and several NICs, which is created by KVM (Kernel-based Virtual Machine) in OpenStack.

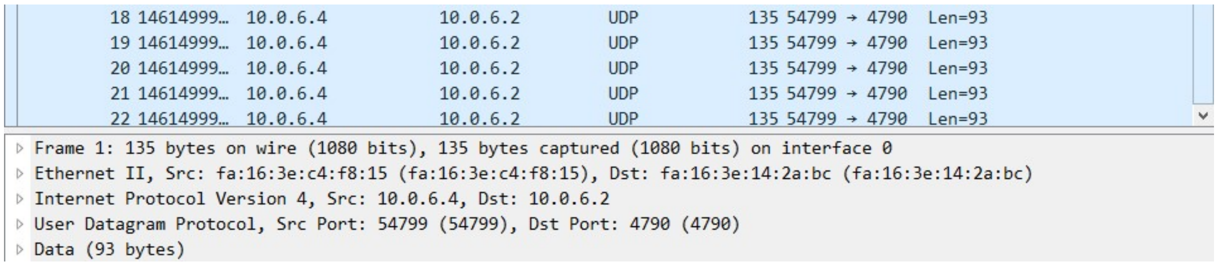
The controller is POX [20], an open source SDN controller. Cooperated with Nshkmod [12], Open-VSwitch [17] is taken as the service forwarder in a switch, which communicates with the controller with standard OpenFlow Protocol. Nshkmod is an open-source Linux kernel module implementation of Network Service Header, which is used to encapsulate and de-encapsulate NSH for service flows. The implementation of the stateful security function proxy is also based on OpenVSwitch and nshkmod, in order to remove/insert NSH encapsulation of packets and redirect them to the next service forwarder or the receiver. Additionally, a service state list is maintained by the proxy to achieve a stateful security function. The firewall and NAT are both implemented by iptables [13], which is integrated into the Linux kernel. The service classifier is responsibility to create a flow and assign an appreciate security service function chaining for it. The flow will arrive at the receiver finally.

To achieve standard SFC routing between two different VMs in Data Transmission Layer, each virtual NSH interface (named nshx), created by Nshkmod, is bound up with a NIC (named ethx) in an ovs net-bridge. The ovs net-bridge has ability to forward packets from a virtual NSH interface to its related NIC in order to encapsulate it by NSH header, and vice versa. Additionally, the data transmission between two virtual NSH interfaces is accomplished by a special tunnel in Linux kernel. In this way, a service forwarder can communicate with its corresponding stateful security function proxy. The stateful security function proxy in a switch can send/receive data to/from its corresponding security function in a traditional routing way (IP routing).

4 The Verification of Proof-of-Concept and Performance Evaluation of the Stateful Proxy

This section is divided into two subsections: one is the feasibility verification of the architecture, and the other is the performance evaluation of the stateful firewall achieved by the stateful security function proxy.

Verification of the feasibility. To verify the feasibility of our architecture in the data plane, we need to prove the flow passes through the established security service chaining. Therefore, we check the correction of the flow encapsulation procedures in a security service chaining. In order to examine the encapsulation, we use Wireshark [5] to analyze the related information of packets on both virtual NSH interfaces and their corresponding NICs. Here, we just capture the packets from eth0 and nsh1 in switch2 to illustrate a typical NSH de-encapsulation procedure, the cases of other interfaces are similar.



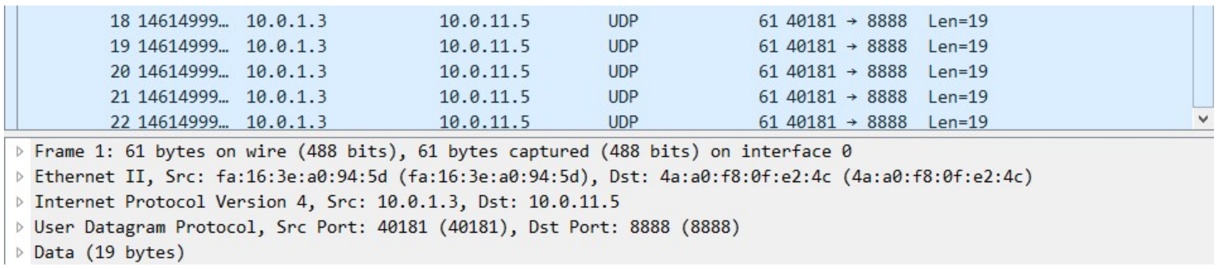
```
18 14614999…  10.0.6.4        10.0.6.2       UDP         135 54799 → 4790  Len=93
19 14614999…  10.0.6.4        10.0.6.2       UDP         135 54799 → 4790  Len=93
20 14614999…  10.0.6.4        10.0.6.2       UDP         135 54799 → 4790  Len=93
21 14614999…  10.0.6.4        10.0.6.2       UDP         135 54799 → 4790  Len=93
22 14614999…  10.0.6.4        10.0.6.2       UDP         135 54799 → 4790  Len=93
```

```
▷ Frame 1: 135 bytes on wire (1080 bits), 135 bytes captured (1080 bits) on interface 0
▷ Ethernet II, Src: fa:16:3e:c4:f8:15 (fa:16:3e:c4:f8:15), Dst: fa:16:3e:14:2a:bc (fa:16:3e:14:2a:bc)
▷ Internet Protocol Version 4, Src: 10.0.6.4, Dst: 10.0.6.2
▷ User Datagram Protocol, Src Port: 54799 (54799), Dst Port: 4790 (4790)
▷ Data (93 bytes)
```

Figure 4: packets captured from eth0 in switch2

Figures 4 shows the information of packets on eth0. These packets are created in previous switch (switch1), and their encapsulations are implemented by the virtual NSH interface nsh2 in the stateful security service proxy of switch1. According to the definition in [19], three kinds of NSH encapsulation are available, including GRE, VxLan-GPE and Ethernet. Here, we use the technique of VxLan-GPE to encapsulate the flow, which adopts 4790 as its UDP port number.

Figures 5 shows the original UDP flow de-encapsulated by the virtual NSH interface nsh1 in switch2. Comparing these packets with the packets on eth0, it is apparent that the NSH encapsulation can be regarded as a UDP tunnel with destination port number 4790. By further analyzing UDP payloads of the packets on eth0, it is easy to find that they are same as the Ethernet frames captured on nsh1. Additionally,



```
18 14614999…  10.0.1.3      10.0.11.5    UDP     61 40181 → 8888  Len=19
19 14614999…  10.0.1.3      10.0.11.5    UDP     61 40181 → 8888  Len=19
20 14614999…  10.0.1.3      10.0.11.5    UDP     61 40181 → 8888  Len=19
21 14614999…  10.0.1.3      10.0.11.5    UDP     61 40181 → 8888  Len=19
22 14614999…  10.0.1.3      10.0.11.5    UDP     61 40181 → 8888  Len=19
▷ Frame 1: 61 bytes on wire (488 bits), 61 bytes captured (488 bits) on interface 0
▷ Ethernet II, Src: fa:16:3e:a0:94:5d (fa:16:3e:a0:94:5d), Dst: 4a:a0:f8:0f:e2:4c (4a:a0:f8:0f:e2:4c)
▷ Internet Protocol Version 4, Src: 10.0.1.3, Dst: 10.0.11.5
▷ User Datagram Protocol, Src Port: 40181 (40181), Dst Port: 8888 (8888)
▷ Data (19 bytes)
```

Figure 5: packets captured from nsh1 in switch2

the above-mentioned NSH de-encapsulation procedure is reversible, and the encapsulation procedure is unnecessary to go into details.

Thus, the results show that the testbed achieves traffic routing in a standard SFC way. In other word, the feasibility of the architecture in the data plane is verified.

Performance evaluation. We compare the performance of the stateful firewall with the stateless firewall by running the stateful security function proxy in different modes. To keep things straight, we choose the transmission time of each packet as the measurement.

flow ID	matching info	service state
1	(10.0.1.3, 10.0.11.5, udp, 40181, 8888)	allow
...	...	...

Table 2: Service State List for Firewall

The stateful security function proxy in switch1 maintains a service state list as seen in Table 2, and a rule is existed. The rule means that the flow can be returned to the service forwarder directly, if the flow matches with source IP of 10.0.1.3, destination IP of 10.0.11.5, type of UDP, source port of 40181 and destination port of 8888. The service state of the flow is pre-defined according to the corresponding firewall rule. In the stateful mode, the proxy can process the flow without the firewall. As a contrast, the proxy running in the stateless mode need to forward all traffic to its corresponding firewall regardless of the list.

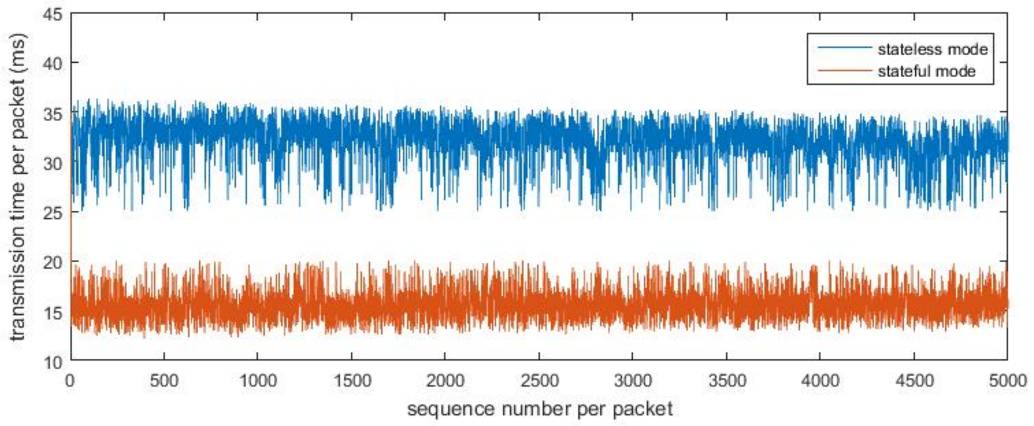


Figure 6: transmission time of each packet in two proxy modes

The comparison of the transmission time of packets in two running modes is shown in Figures 6, where the blue line represents the transmission time in the stateless mode and the yellow line represents that in the stateful mode. Obviously, the transmission time of each packet in the stateful mode is shorter than that in the stateless mode. In other word, running the stateful security function proxy in the stateful mode can significantly decrease the transmission time of each packet. By calculating the average transmission time, we can find that the transmission time in the stateful mode is usually reduced by half. Therefore, the stateful security function proxy improves the forwarding efficiency of nearly fifty percent in the stateful mode.

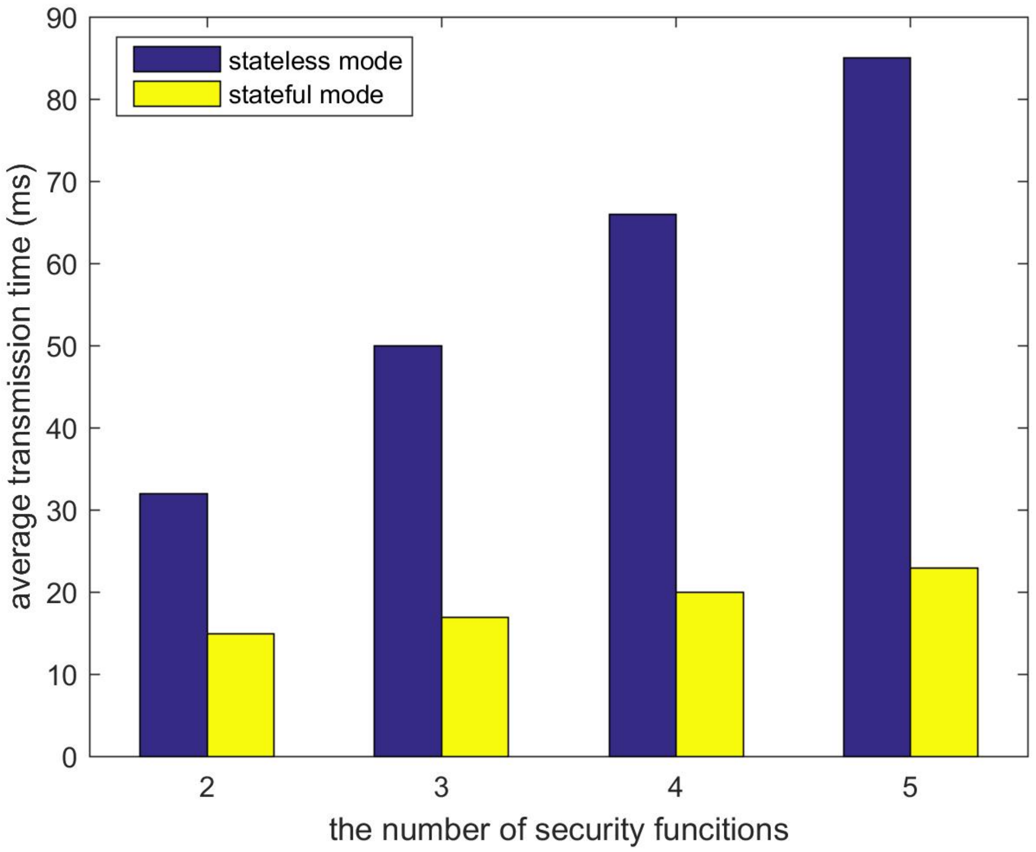


Figure 7: average transmission time with two proxy modes in different cases

In order to prove that the more stateful security function proxies are used, the more efficient it can get, we measure average transmission time in some different cases. In these cases, we insert more security functions with their corresponding stateful proxies into the security service chaining. The result is shown in Figures 7. With stateful security functions adding, there is no significant increasing of the transmission time. However, as a contrast, the transmission time of a security service chaining in stateless mode grows fast. Thus, the stateful proxy can play a more important role in the long security service chaining and significantly improve the forwarding efficiency in a real use-case.

5 Related Work

There are many researches on using service chains to meet diversified user requirements in the cloud, and the security considerations of traditional network infrastructures have been discussed deeply and widely. Unfortunately, we have not seen any work on achieving security access and privacy protection by taking advantages of the security service chaining. Therefore, we purposed the architecture and implement it in the data plane.

Recently, combining different service functions to provide a custom and scalable service chaining for a user is one of the research focuses, and there are some related researches [2, 6, 7, 18] referring to the core idea of SFC. To achieve a service chaining in a cloud environment, [2] purposes a solution. It focuses on the design of the controller, and its implementation is tested on the Mininet [9] emulation platform. Different from [2], [7] purposes a new architecture named Stratos, which achieves dynamically instantiate new middleboxes on demand in Software-Defined Network (SDN). Stratos addresses three main problems in steering traffic to appropriate middleboxes: elastic scaling, middlebox placement and flow distribution. [6] purposes the FlowTags architecture to achieve the integrate middleboxes into SDN network. FlowTags reduces overhead as much as possible compared with traditional SDN mechanism, but needs the FlowTags-enhangced middleboxes. [18] implements dynamic traffic routing without modifying traditional middleboxes. Taking resource constrains into account, SIMPLE purposes an algorithm to generate flow paths and forwarding rules. However, these architectures have no consideration in network security. Thus, in this paper, we concentrate on providing the user-defined security service by creating a security service chaining in cloud.

With the rapid development of the virtualized and “softwarized” network, the security and privacy of the new network architecture attract more and more attentions in academic community. There are some schemes of network security function deployment in SDN, such as [10] and [11]. [10] purposes an OpenFlow-based security framework. It describes how to deploy security policies on the security functions which the flow pass through. The main contributions of [10] are creating security policies in a human-readable language for network security assurance and achieving a low workload controller. [11] also purposes a OpenFlow-based security service. It combines the firewall with an OpenFlow switch into the stateful firewall to improve the forwarding efficiency of the flow. Nevertheless, they cannot achieve a flexible security service management in the data plane. Therefore, we present an architecture design of the data plane for security service chaining.

6 Conclusion

In this paper, an architecture of mobile-edge stateful security service chaining is purposed to meet increasing and various security requirements of mobile users. With the combination of SFC and cloud/IDC networks, the architecture can deploy security functions more flexible and efficient for different mobile users. Moreover, the stateful security function proxy is introduced to support compatibility to traditional security functions and improve forwarding efficiency of the security service chaining. Finally, we did a lot of work to implement a proof-of-concept testbed, and the experiments verify the feasibility of the architecture and prove the stateful function created by the proxy can shorten transmission time of packets to improve the forwarding efficiency.

Acknowledgments

This work is supported by National High Technology of China ("863 program") under Grant No. 2015AA015702, National Basic Research Program of China ("973 program") under Grant No. 2013CB329101, NSFC of China under Grant No. 61271202 and NSAF under Grant No. U153010038.

References

[1] M. T. Beck, S. Feld, A. Fichtner, C. Linnhoff-Popien, and T. Schimper. Me-volte: Network functions for energy-efficient video transcoding at the mobile edge. In *Proc. of the 18th International Conference on Intelligence in Next Generation Networks (ICIN'15), Paris, France*, pages 38–44. IEEE, February 2015.

[2] F. Callegati, W. Cerroni, C. Contoli, and G. Santandrea. Dynamic chaining of virtual network functions in cloud-based edge networks. In *Proc. of the 1st IEEE Conference on Network Softwarization (NETSOFT'15), London, UK*, pages 1–5. IEEE, April 2015.

[3] X. Chen, L. Jiao, W. Li, and X. Fu. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking*, October 2015.

[4] Cisco. Cisco visual networking index: Global mobile data traffic forecast update, 2015–2020. Technical Report 1, CISCO, 2016.

[5] G. Combs and contributors. Wireshark. `http://www.wireshark.org`, 1998–2016.

[6] S. K. Fayazbakhsh, L. Chiang, V. Sekar, M. Yu, and J. C. Mogul. Enforcing network-wide policies in the presence of dynamic middlebox actions using flowtags. In *Proc. of the 11th USENIX Conference on Networked Systems Design and Implementation (NSDI'14), Seattle, WA, USA*, pages 543–546. ACM Press, April 2014.

[7] A. Gember, A. Krishnamurthy, S. S. John, R. Grandl, X. Gao, A. Anand, T. Benson, V. Sekar, and A. Akella. Stratos: A network-aware orchestration layer for virtual middleboxes in clouds. *Computer Science - Networking and Internet Architecture*, pages 1–14, March 2014.

[8] E. J. Halpern and E. C. Pignataro. Service function chaining (sfc) architecture. Technical Report RFC 7665, Internet Engineering Task Force (IETF), 2015.

[9] B. Lantz, B. Heller, and N. McKeown. A network in a laptop: rapid prototyping for software-defined networks. In *Proc. of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets-IX'10), Monterey, California, USA*, pages 1–6. ACM Press, October 2010.

[10] A. Lara and B. Ramamurthy. Opensec: A framework for implementing security policies using openflow. In *Proc. of the IEEE Global Communications Conference (GCC'14), Austin, Texas, USA*, pages 781–786. IEEE, December 2014.

[11] S. Miteff and S. Hazelhurst. Nfshunt: A linux firewall with openflow-enabled hardware bypass. In *Proc. of the IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN'15), San Francisco, CA, USA*, pages 100–106. IEEE, November 2015.

[12] R. Nakamura. Nshkmod. `https://github.com/upa/nshkmod`, 2015–2016.

[13] netfilter core team and contributors. iptables. `http://www.netfilter.org/projects/iptables/index.html`, 1999-2014.

[14] S. Nunna, A. Kousaridas, M. Ibrahim, M. Dillinger, C. Thuemmler, H. Feussner, and A. Schneider. Enabling real-time context-aware collaboration through 5g and mobile edge computing. In *Proc. of the 12th International Conference on Information Technology-New Generations (ITNG'15), Las Vegas, Nevada, USA*, pages 601–605. IEEE, April 2015.

[15] G. Orsini, D. Bade, and W. Lamersdorf. Computing at the mobile edge: Designing elastic android applications for computation offloading. In *Proc. of the 8th IFIP Wireless and Mobile Networking Conference (WMNC'15), Munich, German*, pages 112–119. IEEE, October 2015.

[16] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, A. Neal, et al. Mobile-edge computing – introductory technical white paper. Technical Report 1, The European Telecommunications Standards Institute, 2014.

[17] B. Pfaff, J. Pettit, T. Koponen, E. J. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado. The design and implementation of open vswitch. In *Proc. of the 12th USENIX Conference on Networked Systems Design and Implementation (NSDI'15), Oakland, CA, USA*, pages 117–130. ACM Press, May 2015.

[18] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu. Simple-fying middlebox policy enforcement using sdn. In *Proc. of the Flagship Annual Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'13), Hongkong, China*, pages 27–38. ACM Press, August 2013.

[19] P. Quinn and U. Elzur. Network service header. IETF Internet-draft (work in progress), March 2016. http://tools.ietf.org/html/draft-ietf-sfc-nsh-04.

[20] N. Repo and contributors. The pox controller. `http://www.noxrepo.org/pox/about-pox/`, 2012–2014.

[21] S. Sardellitti, G. Scutari, and S. Barbarossa. Joint optimization of radio and computational resources for multicell mobile-edge computing. *IEEE Transactions on Signal and Information Processing over Networks*, 1(2):89–103, June 2015.

Author Biography

Guanwen Li received his B.S. degree in telecommunications engineering from Beijing Jiaotong University in 2014, and then entered National Engineering Lab for Next Generation Internet Interconnection Devices at BJTU. He is currently working towards the Ph.D. degree in telecommunications and information system at BJTU, China. His main research is directed to the architecture of next generation internet, network service management and network security. His other research interests include satellite network and mobile internet.

Guanglei Li received his B.S. degree from Beijing Jiaotong University in June 2015. He is currently a Ph.D. candidate at the National Engineering Laboratory for Next Generation Internet Interconnection Devices, Beijing Jiaotong University, Beijing, China. His research interests includes next-generation networks and space networks.

Huachun Zhou received the B.S. degree from the People's Police Officer University of China in 1986. He received the M.S. in telecommunication automation and Ph.D. degrees in telecommunications and information system from Beijing Jiaotong University in 1989 and 2008, respectively. In October 1994, he joined Institute of Automation Systems, BJTU, where he is a lecturer. From Apr. 1999 - Sep. 2009, he was a senior engineer at School of Electronic and Information Engineering, BJTU, and at Network Management Research Center, BJTU. From Oct. 2009 to now, he is a professor in National Engineering Lab for Next Generation Internet Interconnection Devices at BJTU. He has authored more than 40 peer-reviewed papers and he is the holder of 17 patents. His main research interests are in the area of mobility management, mobile and secure computing, routing protocols, network management and satellite network.

Bohao Feng received his B.S. degree in telecommunications engineering from Beijing Jiaotong University in 2011, and then entered National Engineering Lab for Next Generation Internet Interconnection Devices at BJTU. He is currently working toward his Ph.D degree in telecommunications and information system at BJTU, China. His research interests are ID/Loc Split network architecture, Software Defined Networking, Information-Centric Networking, network-based caching mechanism, Delay Tolerant Networking, mobile Internet and multicast

Taixin Li received the B.S. degree in telecommunications engineering from Beijing Jiaotong University in 2013, and then entered National Engineering Lab for Next Generation Internet Interconnection Devices at BJTU. Currently, he is pursuing the Ph.D. degree in telecommunications and information system at BJTU, China. His research interests include next generation internet, network services and satellite network