SCIENCE CHINA
Information Sciences

• LETTER •

August 2022, Vol. 65 189102:1–189102:3
https://doi.org/10.1007/s11432-019-2731-7

Efficient middlebox scaling for virtualized intrusion prevention systems in software-defined networks

Junchi XING[1], Chunming WU[2,1*], Haifeng ZHOU[2,1], Qiumei CHENG[1], Danrui YU[1] & Mayra MACAS[1]

[1]*College of Computer Science, Zhejiang University, Hangzhou* 310027*, China;*
[2]*Zhejiang Lab, Hangzhou* 311100*, China*

Received 15 June 2019/Revised 2 October 2019/Accepted 17 December 2019/Published online 24 May 2021

Citation Xing J C, Wu C M, Zhou H F, et al. Efficient middlebox scaling for virtualized intrusion prevention systems in software-defined networks. Sci China Inf Sci, 2022, 65(8): 189102, https://doi.org/10.1007/s11432-019-2731-7

Dear editor,

An intrusion prevention system (IPS) [1] is an important security function that detects and filters out the detected anomaly data traffic in networks. Currently, network function virtualization (NFV) [2] is able to virtualize IPSes as virtualized IPS (VIPS) [3] by encapsulating security functions in software-based middleboxes that can be elastically scaled to deal with traffic volume variations [4, 5]. Moreover, an emerging software-defined networking (SDN) paradigm [6] consists of a programmable controller and several belonging switches, where the controller communicates with the switches using southbound protocols to gather network statistics and to program the network. Thus, SDN is allowed to facilitate the central control to scale middleboxes and to redistribute the data traffic to the scaled middleboxes [7].

Furthermore, the problem lies in how to achieve an efficient scaling of middleboxes for VIPS in SDN, which is challenging due to two key factors: (a) the middleboxes must be accurately scaled out/in with high resource usage, to keep up with the pace of dynamic traffic volumes while exactly meeting rigorous IPS performance requirements [8]; and (b) the traffic should be carefully distributed to the scaled middleboxes in order to minimize the overhead of flow detection state sharing [9]. To achieve this, a flow distribution scheme that avoids flow division is needed. A more detailed statement of this problem is presented in Appendix A.

In this study, a novel VIPS framework called ESBox is proposed, which enables efficient scaling of middleboxes for VIPS in SDN. ESBox is composed of two key modules, i.e., middlebox scaler (MS) and flow distributor (FT), which overcome the two challenges for efficient scaling, respectively. The MS module periodically gathers traffic statistics from the SDN switches with the aid of an SDN controller. According to the gathered statistics and IPS performance requirements specified by an administrator, this module computes the minimal number of middleboxes to be scaled out or in. Then, based on the computed result, this module adds or removes the middleboxes using the available computing resources anywhere in the network. The FT module devotes to distributing the flows among the traffic to the scaled middleboxes through installing forwarding rules into the SDN switches. This module uses a flow distribution algorithm for reducing flow division and the group table in SDN switches for load balance.

ESBox architecture. Figure 1 shows the ESBox architecture and its two component modules in the control plane, and the related devices in the data plane. Each module is implemented as an SDN application running atop the SDN controller. Specifically, the MS module is in charge of scaling the middleboxes according to the real-time traffic volumes and IPS performance requirements, and the FT module is used to distribute flows among the traffic to the scaled middlebox. These two modules will be described detailedly later. In addition, the middlebox is a Docker[1)] container that encapsulates a Snort[2)] tool, which is for signature based intrusion prevention. The Snort tool is configured in an inline mode to handle the received traffic. Specifically, it filters the malicious traffic and reports the alerts to the SDN controller via the Unix domain socket[3)]. Otherwise, it sends the legitimate traffic to its original destination. SDN switch is responsible for reporting the current network information to the MS module and redirecting traffic according to the forwarding rules installed by the FT module.

Middlebox scaler. Firstly, this module needs to compute the optimal number of the middlebox to scale according to traffic volumes. The traffic processing of ESBox is modeled as several $M/M/1/K$ queueing systems, the reasons are presented in Appendix B. To formulate the dynamic traffic, we let $\mathrm{TM} = \{\lambda_{ab} | a, b \in U\}$ be the traffic matrix (TM) at the

* Corresponding author (email: wuchunming@zju.edu.cn)
1) Docker. https://www.docker.com/.
2) Snort. https://www.snort.org/.
3) Unix domain socket. https://ryu.readthedocs.io/en/latest/snort_integrate.html.

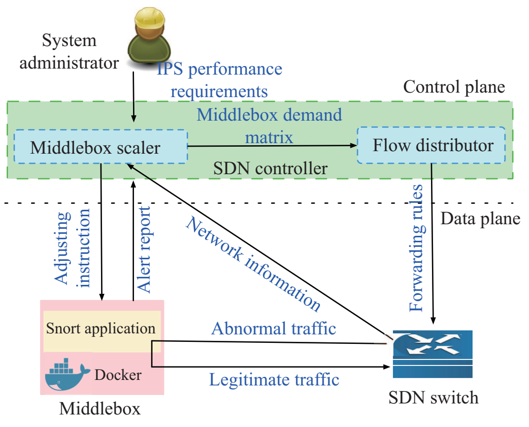

Figure 1 (Color online) ESBox architecture.

present time, where U is the set of the users, and λ_{ab} is the packet arrival rate for the origin–destination pair user a to user b. Two metrics that are widely considered in IPS performance measurement [1] are as follows. (1) Processing time of prevention (PTP) describes the time lapse between the launch of the attack packet and being filtered by the IPS. (2) Prevention rate (PR) refers to the proportion of the packets completely processed by IPS in the total transmitted packets. We let PTP and PR be the IPS performance constraint, which is the maximal threshold specified by an administrator. Moreover, the processing time of a middlebox follows an exponential distribution with parameter μ, and K is the queue capacity of a middlebox. To guarantee high resource utilization, the middleboxes should be invested to a minimum. Thus, with the system model, our goal is to compute the minimal total number of middleboxes n.

Based on the queueing theory, the packet process time and the packet loss rate in our system can be expressed as follows, respectively:

$$\phi(n)=\left\{1-(K+2)\left(\frac{\lambda_t}{\mu n}\right)^{K}+K\left(\frac{\lambda_t}{\mu n}\right)^{K+1}+\cdots\right.$$
$$\left.+(K+1)\left(\frac{\lambda_t}{\mu n}\right)^{2K}-K\left(\frac{\lambda_t}{\mu n}\right)^{2K+1}\right\}$$
$$\Bigg/\left\{\mu\left(\frac{\lambda_t}{\mu n}\right)\times\cdots\times\left(1-\left(\frac{\lambda_t}{\mu n}\right)^{K+1}\right)^2\right\}+d_n, \qquad (1)$$

$$\psi(n)=1-\frac{1-\frac{\lambda_t}{\mu n}}{(\frac{\mu n}{\lambda_t})^K-\frac{\lambda_t}{\mu n}}, \qquad (2)$$

where $\rho=\frac{\lambda}{\mu}$, representing the server utilization of the queueing system, d_n is the network delay, and $\lambda_t=\sum_{a,b\in U}\lambda_{ab}$ represents the total packet arrive rate by accumulating the elements in TM.

Based on Eqs. (1) and (2), the computation of the minimum number of middleboxes can be transformed into an optimization problem, which is formulated as follows:

$$\begin{aligned}\text{mini.}\quad & n && (3a)\\ \text{s.t.}\quad & \phi(n)\leqslant \text{PTP}, && (3b)\\ & \psi(n)\leqslant \text{PR}, && (3c)\\ & n=0,1,\ldots,N_{\max}, && (3d)\end{aligned}$$

where $N_{\max}$ is the maximum number of middleboxes that can be invested in our system. This problem can be easily computed by iterating on the values of n.

Afterwards, let MDM $=\{n_{ab}|a,b\in U\}$ denote the number of middleboxes demanded by the flow from user a to user b. For adapting to the traffic volume, n_{ab} is in proportion to λ_{ab}, thus we have $n_{ab}=\frac{\lambda_{ab}}{\lambda_t}\times n$. And MDM and n will be output to the next module (i.e., flow distributor) for further distribution instruction. In addition, the MS module calls the docker daemon commands to implement the scaling of middlebox.

Flow distributor. This module is devoted to distributing the flows in TM to the n scaled middleboxes based on MDM from the MS module. A bad flow distribution algorithm will lead to unnecessary flow divisions that increase the sharing of flow detection state meaninglessly. Motivated by the examples described in Appendix C.1, we find that the unnecessary flow divisions are avoidable. Thus, we regard Proposition 1 as the principle for avoiding the unnecessary flow divisions, which is also proofed in Appendix C.1.

Proposition 1. The larger flows are prioritized over the smaller flows during the flow distribution. Further, provided that a middlebox can accommodate the flow, we distribute the flow directly to the middlebox rather than divide the flow.

Following Proposition 1, we present the flow distribution algorithm (Algorithm C.1) for reducing the number of flow divisions. We use FDS to store the flow distribution scheme, where each distribution is represented as $[ab,i,c_i]$; this means we will allocate the c_i capacity of M_i to the flow with the origin–destination pair of ab. Lines 1 to 9 conduct the initialization of $\mathbb{F}$ and $\mathbb{C}$, where $\mathbb{F}$ is the set of flows to which the middleboxes have not yet been allocated, and $\mathbb{C}$ is the set of the available capacity of the middleboxes. Lines 10 to 30 are the loops to impose FDS with the constraint of $\mathbb{F}\neq\emptyset$; this means the loop continues until all the flows are middlebox allocated. Lines 11 to 19 handle the flows that cannot be accommodated by any middlebox capacity in $\mathbb{C}$, thus requiring necessary divisions. Each such flow is divided into several parts to reach the middlebox capacity. Lines 21 to 29 handle the flows sharing the middlebox capacity, which may yield unnecessary divisions based on Proposition 1. Line 20 sorts $\mathbb{F}$ in descending order according to the flow volumes to give priority to larger flows (we use merge-sort in this study). Subsequently, lines 22 to 29 search $\mathbb{C}$ to obtain the middlebox capacity that can accommodate the entire flow and allocate it to the flow. In addition, flows that cannot be accommodated by any element in $\mathbb{C}$ remain and will be handled in the next loop.

Moreover, based on the flow distribution scheme FDS, the FT installs group table rules into the SDN switches for traffic redirection while achieving load balance.

Conclusion. To solve the problem of efficient scaling of middleboxes for VIPS, ESBox is proposed. ESBox consists of two modules: a middlebox scaler module to gather network information and IPS performance requirements to compute the optimal number of middleboxes to be scaled with high resource utilization and IPS performance guarantee, and a flow distributor module that redistributes flows to the scaled middleboxes while ensure minimizing flow detection state sharing and load balance. Moreover, ESBox was implemented, and the evaluation results (see Appendix D) revealed that it scaled out/in middleboxes adapted to the traffic volumes, while achieving the goal of high resource utilization (compared to the related studies in Appendix E), IPS performance guarantee, and minimizing flow detection state sharing, with a minor impact on the traffic throughput.

Acknowledgements This work was supported by National Key R&D Program of China (Grant No. 2020YFB1804705), Industrial Internet Innovation and Development Project (2019) — Project of Time Sensitive Network (TSN) Technology based Programmable Logical Controller (PLC) (Grant No. TC190A449), the Key R&D Program of Zhejiang Province (Grant Nos. 2020C01077, 2021C01036), and Major Scientific Project of Zhejiang Lab (Grant No. 2018FD0ZX01).

Supporting information Appendixes A–E. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Scarfone K, Mell P. Guide to intrusion detection and prevention systems (IDPS). NIST Spec Publ, 2007, 800: 94

2 Malathi V, Takehiro S, Molly B, et al. Network function virtualization: a survey. IEICE Trans, 2017, 100: 1978–1991

3 Mishra P, Pilli E S, Varadharajan V, et al. Intrusion detection techniques in cloud environment: a survey. J Netw Comput Appl, 2017, 77: 18–47

4 Xiong W, Hu H P, Xiong N X, et al. Anomaly secure detection methods by analyzing dynamic characteristics of the network traffic in cloud communications. Inf Sci, 2014, 258: 403–415

5 Srikanth K, Sudipta S, Albert G, et al. The nature of data center traffic: measurements analysis. In: Proceedings of the 9th ACM SIGCOMM Internet Measurement Conference, Chicago, 2009. 202–208

6 Kreutz D, Ramos F M V, Esteves Verissimo P, et al. Software-defined networking: a comprehensive survey. Proc IEEE, 2015, 103: 14–76

7 Shin S, Wang H P, Gu G F. A first step toward network security virtualization: from concept to prototype. IEEE Trans Inform Forensic Secur, 2015, 10: 2236–2249

8 Holger D, Anja F, Vern P, et al. Predicting the resource consumption of network intrusion detection systems. In: Proceedings of ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, 2008. 135–154

9 Lorenzo D C, Robin S, Somesh J. Beyond pattern matching: a concurrency model for stateful deep packet inspection. In: Proceedings of ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, 2014. 1378–1390