Dynamic scheduling mechanism for software-defined security resources based on multi-mode load balancing

Tao Zhang*, Zhipeng Shao

1 Global Energy Interconnection Research Institute, Nanjing 210003, China
*Corresponding author’s e-mail: songyubo@seu.edu.cn

Abstract. Rather than the traditional rigid and decentralized security mechanism, software-defined security provides a flexible and centralized solution by extracting security mechanisms from the security hardware layer to the software layer, but the existing software-defined security mechanism is prone to the problem of high load on physical devices performing security functions leading to performance limitations. To address these problems, this paper proposes a dynamic scheduling mechanism for security resources based on load balancing, while designing the southbound interface of software-defined networks to be scheduled by the control plane of software-defined security for flows according to established policies and logical topologies. The mechanism can match the load resources required by the security components, thus avoiding the problem of overall performance degradation due to excessive load and long queuing time of individual devices.

1. Introduction
Network security service is an indispensable function in the current network. The traditional security protection strategy is to inform the security device of the address of the entity object to be protected, and the security device establishes a connection with it through the traditional network to protect it, while in the existing network, due to the virtualization of various network facilities and network functions, the security service is no longer simply to protect a certain entity of a certain device, but a complex internal cloud tenant, the virtual machines and other objects. The current security architecture is difficult to meet the development requirements of the network, cannot make sufficient progress in the current open, virtualization and other characteristics of the network protection needs, and most importantly, because most of the current security management function modules are deployed in the east-west direction, so that the security management function cannot be organically integrated with the whole system. Although the development of mimetic technology to a certain extent to improve the degree of automation of security, but real-time, the diversity of applications and dynamic characteristics of these needs are still not met, the security protection capacity is not much improved, or not only weak and high cost.

Service awareness, flexible asset configuration, on-demand deployment and other security capabilities represent the development direction of the future development of network security mechanisms, although now in the global vision, security automation and other aspects have made some progress, but to achieve process automation and security operations still have a long way to go. At present, only the United States to promote the SCAP (Security Content Automation Protocol) [1]. Although there is currently some progress in the sharing and interaction of vulnerability information and in the opening of security information [2]. Security as a service is weak and rigid in innovation compared to other services, and the reuse and inheritance of security capabilities are poor and only stay
at the product update level. The current security architecture is difficult to meet the development requirements of the network. The diversity of applications and dynamic characteristics of these needs are still not met, the security protection capacity is not much improved, or not only weak and high cost.

Software-defined network architecture will bring new opportunities for network security protection [3-5]. The architecture decouples the control plane and data plane of network security devices, and the bottom layer virtualizes traditional hardware security devices and pools resources through Network Function Virtualization (NFV) technology to form a unified security resource pool, and the top layer defines and describes network data flow in a software-defined manner with fine granularity, and redirects the target network data flow to a sequence of multiple virtual security devices for security protection. Although building a security service chain based on SDN and NFV technology can effectively improve the efficiency of security protection and reduce the cost and complexity of network management [6], the existing security service chain mechanism still has major defects, mainly as how to effectively balance the load of various types of virtual security devices when scheduling security resources, improve the efficiency of security service chain security resource scheduling, and then improve the resource utilization rate of the physical servers it hosts.

This paper proposes a dynamic scheduling mechanism for software-defined security resources based on multi-mode load balancing. The prototype system of the resource scheduling mechanism is implemented through the design of a unified southbound API, the virtualization of the underlying security resources, and the design of the scheduling algorithm.

2. Related work
Shinetal proposed a controller framework design that can protect the control-plane against unexpected operational failures and loss of network control [7]. However, studies on SDS do not adequately consider security resource capabilities and scheduling. The purpose of this research work is to match the appropriate security resources for task processing and to achieve resource load balancing to guarantee a high efficiency of the service. Mahapatra [8] proposed that the resource capabilities separated from the resource orchestration and control layers at the application layer are integrated in a certain process and encapsulated into a series of common and reusable atomic services. Qiu [9] proposed a SDS (Software Defined Security) architecture, which is similar to the three-layer architecture of SDN. The security controller in the SDS architecture is specifically implemented in the literature [10], and the feasibility of this SDS architecture is verified with experimental scenarios such as anti-DDoS scenarios, port scan detection, and high traffic alerting, and the necessary performance comparative evaluations are performed. The experimental results demonstrate the feasibility of this architecture and the superiority of this architecture in detecting and protecting against various attacks effectively at the flow level and packet level.

However, all solutions did not implement the software definition of security resources in the physical layer of the three-layer architecture of software-defined security SDS, and does not allow effective management and scheduling of physical layer security resources, so this paper adds a resource scheduling mechanism to this foundation, and then schedules security resources in the physical layer through a unified southbound API to achieve integrated and comprehensive security services.

3. Resource scheduling mechanism
The flow of the resource scheduling mechanism in this paper can be described as follows: when a user requests a service from the security controller at the application layer to the security controller at the control layer, the security controller resolves the relevant parameters of this service, then reads the relevant parameters of the security resources from the physical layer, then inputs the relevant parameters of this service and the relevant parameters of the security resources into the resource scheduling algorithm, and finally the resource scheduling algorithm calculates the most suitable security resources to provide the service for the user. Finally, the resource scheduling algorithm calculates the most suitable security resources to provide services for users.

This resource scheduling mechanism consists of the following components.
The security controller and physical layer security device resources form a unified southbound API interface.

The pooling of resources for security equipment, the hardware security equipment or software security functions abstracted, the need for vendors to open a unified interface to form a pool of security resources available for security controller calls.

Scheduling algorithms that achieve certain goals (such as reducing energy consumption, improving resource utilization, or multiple goals, etc.).

Resource scheduling is performed logically, not physically, by mapping the physical topology. To the logical topology, the control plane of the SDN network schedules the flows according to the established policy and logical topology, and lets the flows go through the corresponding security devices for processing (e.g., monitoring, detection, cleaning, etc.).

3.1. Resource scheduling mechanism process

When all the above 4 points have been implemented, the resource scheduling mechanism flow can be specifically described as shown in Figure 1.

- The user places an order at the application layer and sends the order to the security controller at the control layer via the northbound interface.
- After receiving the order, the controller parses out the order parameters and sends them to the resource scheduling algorithm.
- The resource scheduling algorithm in the D security controller obtains all device related parameters (e.g. CPU memory usage, etc.) through the southbound interface.
- The resource scheduling algorithm calculates the optimal solution based on the parameters of the order and the parameters of the device according to the goal of the resource scheduling algorithm, and then sends the task down to the device through the unified southbound interface for the device to execute the task.

3.2. Resource scheduling algorithm

In the SDS architecture through a unified security controller and the southbound interface between the physical layer security devices, the pooling of security device resources, this paper through flexible resource scheduling in business needs, operations and maintenance automation than the traditional security model, we choose to achieve the underlying security resource pool load balancing distributed
scheduling algorithm. The goal of this scheduling algorithm is to select the least loaded security device to perform security tasks, such as selecting the least loaded scanner to scan in this task, which can achieve load balancing while also providing faster scanning speed.

The scheduling algorithm input is the task parameters and resource pool engine information filled by the upper layer. The output is a list of security resources suitable for processing the task. The selected security resource is required to complete the task request completely and efficiently. The design and implementation of the scheduling algorithm takes into account the scalability and performance requirements of the long-term evolution of the system as much as possible, but the security resource interface of the first phase of the project is not uniform, the interface function is not perfect, the foreground interface provides limited parameters, and the system lacks actual operational data. Focus on ensuring the scalability and scalability of the overall architecture design, functional priority to achieve the most urgent needs of the current.

Northbound received capability requests for services may be for multiple assets, each of which may contain multiple security service requests. The capability requests are split into task requests by other modules in terms of a single security maintenance for one asset. For capability requests with heterogeneous engine requirements, order management splits two scans into two task requests, and if the capability request does not specify an equipment vendor, order management specifies a different equipment vendor for each of the two scans. All task requests are tagged with a unique task id, and the real-time service requests are sent to the resource scheduling algorithm. The device ids obtained from resource management by the resource scheduling algorithm are all available in real time, from which the algorithm selects the three best devices in order of suitability and outputs the task id and a set of device ids.

After the scheduling algorithm receives the task, the scheduling algorithm sub-module obtains the current CPU utilization, memory utilization and hard disk utilization of all resources in the resource pool through the southbound API and then calculates the load according to equation (1), and then directly selects the resource (scanner) with the lowest load (i.e., the minimum value of equation (1)) to execute the task.

Prototype system In this paper, a load-balanced scheduling algorithm is chosen, i.e., the optimal policy in this paper is to select the least loaded security resource to execute the task when it is issued. The load is a function about CPU usage, memory usage and hard disk usage, which can be adjusted according to the actual situation. At present, it is simply a weighted average of CPU usage, memory usage and hard disk usage, and in fact, a more accurate load expression can also be obtained by a neural network learning algorithm. The load of the j-th scanner is expressed as

\[
load_j = \text{cpuusage}_j \times \text{cpumusage}_j + \text{memoryusage}_j \times \text{memoryusage}_j + \text{diskusage}_j \times \text{diskusage}_j
\]

After the resource scheduling algorithm receives the task, it obtains the current CPU usage, memory usage and hard disk usage of all resources in the resource pool through the southbound API and then calculates the load according to equation (1), and then directly selects the resource (scanner) with the lowest load (i.e., the minimum value) to execute the task.

4. Results & Discussion

4.1. Experimental design

This prototype system takes the provision of URL (Uniform Resource Locator) scanning service as an example to demonstrate that this SDS architecture is capable of flexible scheduling of physical layer security resources through physical layer security resource abstraction and a unified southbound API interface, and achieves the separation of physical layer security device control functions from security devices, and the security. The controller informs the physical layer security devices what to do, while the physical layer security devices simply execute the commands from the control layer. Our next step is to add a variety of security devices in the physical layer to enrich the security service capability, and further store the execution results of security devices in the knowledge base for data mining, and
combine it with the app store security application scheduling being done to provide comprehensive security services for users in a more comprehensive and intelligent automated manner.

The application layer in this paper is two web pages integrated into the app store, one of which allows the user to fill in the URL to be scanned and the related parameters, and to see the scan results if the task is completed. To demonstrate load balancing, which is one of the resource scheduling objectives of this paper, the web page has a dynamic graph that shows the real-time load of the physical layer security devices in real time. Another web page shows the orders that have been placed by users and the scan results if the orders have been completed.

Security controller: Security control is deployed on the server and written in java language. Security controller (depends on two services, one for rabbitMQ and the other for zookeeper, both for distributed deployment. rabbitMQ is a message queue service to distribute events among multiple nodes and zookeeper monitors the operational status of each node and centrally manages cluster information. In this paper, we add a resource scheduling module to the security controller.

Physical layer scanners: after the research of open source security scanners choose nikto this paper open source security scanners, in the scanner installed in the virtual machine, at the same time for open source security scanners open a unified interface, currently supports query open source security scanners virtual machine CPU utilization, memory utilization, hard disk utilization and other interfaces, scan results and logs are reported after the scan is completed. This paper designs a scan task per second when 3 scanners are idle, then the user sends a scan task per second, and then designs a scan task per second when the initial load of 3 scanners is different, and finally this paper gets two dynamic graphs of the actual load of 3 scanners, and the end user gets the scan results.

4.2. Experimental results
From Figure 2 we can see that the loads of the three scanners are different at the beginning, with the highest load on Scanner 3, the second highest load on Scanner 2, and the lowest load on Scanner 1. The load of Scanner 3 and Scanner 2 does not change, while the load of Scanner 1 rises, so it can be concluded that the task that arrives at the first moment is indeed given to Scanner 1 with the lowest load. From the figure, we can see that the initial load of scanner 3 is high, and no task is sent to scanner 3 from moment 1 to moment 7, until moment 8 when the load of scanner 3 is lowest, then the task is sent to scanner 3. The graph also clearly shows that the load of the three scanners rises alternately, so that one scanner does not perform a lot of tasks while the other scanner does not perform many tasks.

![Figure 2. Real-time load per scanner.](image)
Figure 3. Share of load per scanner.

Since this figure is not very obvious to show the load balancing, the data is normalized and drawn as follows: the horizontal coordinate in Figure 3 corresponds to the time of the above figure, and the vertical coordinate is the proportion of each scanner’s load after the normalization process, and it is easy to see from the figure that the load of each scanner is comparable as time goes on. It can be seen from the graph that the initial scanner 3 has a higher load, and from moment 1 when the task is issued to moment 7 - straight there is no task issued to scanner 3 until moment 8 when scanner 3 has the lowest load, and only then the task is issued to scanner 3. It is also obvious from the graph that the load of the 3 scanners rises alternately, and there will not be a situation where one scanner performs a lot of tasks and the other scanner. This effectively improves the utilization of the device. And in another web page you can get the scan results of the finished scan, you can get the OSVDB number, and you can check the corresponding vulnerability in the official website.

5. Conclusion

This paper proposes a resource scheduling mechanism, which not only realizes load balancing in the physical layer, but also realizes software definition in the process in a real sense, opens a unified interface to the underlying security devices, and then freely operates the underlying security devices through a unified southbound API. This mechanism forms the resource scheduling module of the security controller in the software-defined security architecture. Through the unified southbound API interface, the physical layer security resource control and data separation, the security functions such as scanning and other functions are abstracted from the security devices, and the security controller can flexibly schedule the physical layer security devices to serve the users. This prototype system with scanning as an example is system validated, and the experimental results show that the target load balancing of the resource scheduling mechanism is achieved.

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).\n
References

[1] Van der Geer, J., Hanraads, J.A.J., Lupton, R.A. (2010) The art of writing a scientific article. J. Sci. Commun., 163: 51–59. D. Walternie, S. Quiim, K.Scarfone, and A. Halbardier. "The technical specification for the Security Content Automation Protocol (SCAP). SCA Pversion1.2, NIST Special Publication, vol. 800, p. 126, 2011.
[2] A. Rutkowski, Y. KadobayaA, I. Furey, D. Rajnovic, R. Martin. Rajnovic, R. Martin, T. Takahashi, et al. "CYBEX: the cyber security information exchange framework(x . 1500),"
ACMSIGCOMM Computer Communication Review, vol. 40, pp. 59-64, 2010.

[3] Yang L, Song Y, Gao S, et al. Griffin: An Ensemble of AutoEncoders for Anomaly Traffic Detection in SDN. GLOBECOM 2020-2020 IEEE Global Communications Conference. IEEE, 2020: 1-6.

[4] Gao S, Peng Z, Xiao B, et al. Detection and Mitigation of DoS Attacks in Software Defined Networks. IEEE/ACM Transactions on Networking, 2020, 28(3): 1419-1433.

[5] SONG Yubo, YANG Huiwen, WU Wei, HU Aiqun, GAO Shang. Joint DDoS detection system based on software-defined networking. Journal of Tsinghua University (Science and Technology), 2019, 59(1): 28-35.

[6] Shirmarz A, Ghaffari A. Performance issues and solutions in SDN-based data center: a survey. The Journal of Supercomputing, 2020, 76(10): 7545-7593.

[7] S. Shin et al., "Rosemary: A robust, secure, and high-performance secure, and high performance network operating system," inProc. 21th ACMConf. Comput. Commun. Secur (CCS), 2014, pp. 78-89. 2014, pp. 78-89.

[8] Mahapatra C, Moharana A K, Leung V. Energy management in smart cities based on internet of things: Peak demand reduction and energy savings. Sensors, 2017, 17(12): 2812.

[9] QIU Xiao-feng, ZHAO Liang2, GAO Teng. VSA and SDS: Two Security Architectures in SDN. Journal of Chinese Computer Systems, 2013, 34(10). 2298-2303.

[10] Shan-Shan J, Ya-Bin X. The APT detection method in SDN. 2017 3rd IEEE International Conference on Computer and Communications (ICCC). IEEE, 2017: 1240-1245.