Scheduling of Security Resources in Software Defined Security Architecture

Gang Zhang, Xiaofeng Qiu, Wei Chang
Beijing Key Laboratory of Network System Architecture and Convergence, Beijing University of Posts and Telecommunications, Beijing 100876, China
zhanggang1408@bupt.edu.cn, qiuxiaofeng@bupt.edu.cn, changweitju@163.com

***Abstract*—With the development of Software Defined Networking, its software programmability and openness brings new idea for network security. Therefore, many Software Defined Security Architectures emerged at the right moment. Software Defined Security decouples security control plane and security data plane. In Software Defined Security Architectures, underlying security devices are abstracted as security resources in resource pool, intellectualized and automated security business management and orchestration can be realized through software programming in security control plane. However, network management has been becoming extremely complicated due to expansible network scale, varying network devices, lack of abstraction and heterogeneity of network especially. Therefore, new-type open security devices are needed in SDS Architecture for unified management so that they can be conveniently abstracted as security resources in resource pool. This paper firstly analyses why open security devices are needed in SDS architecture and proposes a method of opening security devices. Considering this new architecture requires a new security scheduling mechanism, this paper proposes a security resource scheduling algorithm which is used for managing and scheduling security resources in resource pool according to user's security demand. The security resource scheduling algorithm aims to allocate a security protection task to a suitable security resource in resource pool so that improving security protection efficiency. In the algorithm, we use BP neural network to predict the execution time of security tasks to improve the performance of the algorithm. The simulation result shows that the algorithm has ideal performance. Finally, a usage scenario is given to illustrate the role of security resource scheduling in software defined security architecture.**

Keywords—open security devices; security resources scheduling; software defined security; security task abstracting; security resource capability abstracting

I. INTRODUCTION

Traditional network security is facing enormous challenges mainly due to the revolution of IT system and the challenge of security threats. The existing security system bases on the assumption that network perimeter is fixed, and it depends on physical methods a lot such as the deployment of hardware security devices and the control of physical topology. However, the emergence of network virtualization, cloud computing and software defined architecture brings new challenges for network security.

The challenges include the following three main aspects:

1) The existing security devices are not open. Therefore, security devices are difficult to be managed as its heterogeneity. They are also difficult to be expanded later. So, it is difficult to meet rapid change of security business demands.

2) Security operation depends on network environment. The deployment of security devices needs according to network topology and practical environment. Therefore, high-efficiency migration and reuse of security devices is difficult to realize.

3) Network security strategy still needs manual configuration. This leads the validity and rationality of security strategy deployment depends on operation staffs a lot. This also leads the high cost of network operation and maintenance.

4) There is no effective orchestration mechanism between security services, which limits the threats detection efficiency.

Based on above analysis, network security operation should keep pace with network evolution. Recent years, the evolution of network presents the trend of virtualization, openness and software defined. In order to keep up with the pace of network evolution and the speed of business innovation, network security should develop towards to software-oriented security mechanism, open interface, easy extension and separation of functional architecture. Applying these network evolution characteristic in network security field, the concept of software defined security (SDS) is emerged. Different enterprises and organizations have

978-1-5386-2209-4/17 $31.00 © 2017 IEEE
DOI 10.1109/CyberC.2017.46

different viewpoints for software defined security. The following are some typical representations:

1) Fortinet[1]

Fortinet's software defined security architecture emphasizes the combination with data center. They aim to apply 'software defined' into network security so that security operation can be flexible as other parts in data center. Fortinet adds three network security planes, which are data plane, control plane and management plane, in network architecture. The function of data plane is realizing device abstraction and service abstraction. The control plane is used for realizing cooperation and automation of the platform. The function of management plane is providing management interface.

2) Embrane[2]

The core of it is replacing hardware devices with virtual devices, and centralizing virtual devices' control in one platform. Therefore, the security operation system can provide security protection service more rapidly and improve operation efficiency.

3) Catbird[3]

Catbird divides security system into two layers. The first layer groups all assets and composes 'catbird TrustZones', and the other layer receives assets' status and the relationship between them. The system platform can find security threats and deal with them in time, and the assets grouping guarantees the flexibility of security strategy.

4) Software defined security architecture that we proposed[4][5]

With SDN for reference, we also put forward a kind of software defined security architecture. Considering SDN's most basic principle, which is the decoupling of control plane and data plane, the software defined security architecture that we put forward has three layers (including application plane, control plane and infrastructure plane). Application plane includes many kinds of security applications, and users can select their needed security protection service. Control plane is the core system of this architecture. For realizing specific security protection function, control plane needs collect network resources, collect user storage and computing resources, manage security resources and schedule security resources, etc. Therefore, we develop a security controller in control plane to finish above-mentioned tasks. Infrastructure plane, which is also called data plane, is composed of physical or virtualized security devices. It communicates with control plane through southbound interface. Security strategies and network data are issued from control plane to data plane through southbound interface. Then, security devices in data plane execute the security protection according to the command from southbound interface.

As can be seen from the four typical software defined security architectures, all of them realize the decoupling of security control plane and data plane. Underlying security devices, which are physical or virtualized, are centralized deployment and form a resource pool. Furthermore, the control of these security resources is logically centralized in one control plane. Therefore, new-type open security devices are needed in SDS Architecture due to following reasons:

- Network management has been becoming extremely complicated due to varying network devices and heterogeneity of network. With open security devices, the centralized management and scheduling of security devices becomes more easier so that they can be conveniently abstracted as security resources in resource pool
- As is well-known, a very important reason for the emergence of SDN is that it breaks the lock of network devices vendors, so that users can freely choose network devices, develop or purchase network applications according to their business needs. The same problem exists in the field of network security.

With software defined security architecture and security resource pool, resource utilization of network security devices can be improved and security task execution efficiency can be improved. Therefore, network security can be improved. Considering that the new situation needs a new resource scheduling mechanism, we propose a security resource scheduling algorithm which is applied in software defined security architecture with above-mentioned situation.

When dealing with security resources scheduling problem in software defined security architecture, the following two main difficulties are existing. The first is there are a wide range of existing security devices and each type of security devices has a different vendor. However, these existing security devices are not open enough. This means existing security devices' interfaces may limit the effect of resource scheduling algorithm because software defined security architecture emphasizes the openness of underlying security devices. The second is how to ensure load balance among security devices so that improving overall performance of security resource pool. In this way, security task execution efficiency and security protection efficiency can be improved so that network security is improved. In order to realize high performance load balance among security resources, the load variance of security resources over future period of time should be considered. So, it is important to predict when a security task will finish executing and release occupancy of a resource. This is difficult to deal with as different security task has different complexity degree so that its execution time is difficult to measure.

In our proposed algorithm, we first abstract a security protection task as several its determined parameters

according to task type. Then, we develop a method which is used for forecasting execution time of security protection tasks according to their abstracted parameters. And then, we describe security resources' task execution ability abstractly based on above work. The security resources scheduling algorithm will select the security resources with suitable type and maximum task execution ability. Therefore, load balancing among security resources can be realized and security protection efficiency can be improved, and then network security is improved.

In this paper, we firstly analyze the reasons of putting forward software defined security and introduce three typical software defined security architecture as well as our own software defined security architecture. Then, we expound the reason why open security devices are needed in SDS architecture and propose a method to open security devices. Next, we introduce the security resource scheduling algorithm we have designed detailedly and show the simulation result of the algorithm. At last, this paper shows an application scenario of the resource scheduling algorithm in our proposed SDS architecture and gives a conclusion.

II. Related Work

Almost all of the existing research on software defined security is to study the implementation architecture of SDS or its northbound applications, such as paper [4][6][7][8]. The most basic feature of these SDS architectures is the decoupling of control plane and data plane. To the best of our knowledge, there are no research on open security devices and security resources scheduling issue in software defined security architecture.

Based on the analysis of the difference of the definition of network resource between SDN and traditional IP network, the idea of the integrated allocation of link bandwidth and flow table for multiple control applications in SDN is proposed in paper [9]. Paper [10] proposes a middle box management architecture with SDN – OpenMiddlebox, by extending OpenFlow to support middle boxes with ClickOS virtual machines (VM), so that programmable middleboxes could be deployed and managed in switches with fast booted ClickOS VMs flexibly. Different from the management and scheduling of network resources in software defined network, management and scheduling of security resources encounter new problems because there are many types of security devices, and their functional features, deployment patterns and working methods are also different. Therefore, it is not possible to abstract out control protocols such as OpenFlow, which is also one of the reason that we develop security controller [4] [5]. Therefore, security resource scheduling requires new mechanism in SDS architecture.

Paper [11] proposes an Application-aware Resource Allocation (App-RA) scheme to predict resource requirements and allocate an appropriate number of virtual machines (VMs) for each application in SDN-based cloud datacenters. The proposed App-RA is an application-aware resource allocation scheme that adapts to all types of applications. The proposed App-RA adopts the neural network based predictor to forecast the requirements of resources (CPU, Memory, GPU, Disk I/O and bandwidth) for an application. Paper [12] presents a load balancing and scheduling algorithm that is stock optimal, without assumed that job durations, which are modeled as random variables, are known or are upper bounded. In our work, the security resources scheduling mechanism is security task type-aware. For different type of security devices, the parameters used in the scheduling algorithm are different. Inspired by App-RA's work, we use BP neural network to forecast the execution time of a security task. With predicted execution time of a security task, the scheduling of security resources can be more efficiency than scheduling tasks with unknown duration.

III. Analysing And Design

A. Open method of security devices

In software defined security architecture, physical or virtualized security devices are centrally controlled in control plane. Therefore, it has significant meaningful to open security devices and formulate unified interface for the same type of security device. This makes centralized management and scheduling of security resources become feasible and convenient.

There are various types of security devices, and their functional features, deployment modes and operating mode are also different. Therefore, it is not possible to abstract a control protocol like 'OpenFlow' between security control plane and data plane. However, the following two works can be done at least.

The first is summarizing the common features of functional requirements of different security devices, listing their common application interfaces. No matter what kind of security device, the following four types of interfaces are needed:

1) Basic information interface: accesses to basic information about the device, such as version number, host name, CPU usage, memory usage, executing security protection tasks and their execution progress, etc.

2) Configuration interface: gets or sets some configuration values for the device, such as network interface, operating mode, etc.

3) Policy interface: some security policies that can be used by the device, such as the five tuple of firewall, WAF protection rules, etc.

4) Log interface: stores and retrieves uploaded log and alarm messages of security device

Second, for a specific type of security device, a unified protocol standard can be developed. Take the firewall as an example, the delivery interface of access control strategy

five tuple can be specified so that the same application interfaces can be used by different vendors' firewall.

Through above-mentioned steps, centralized management and scheduling of security resources become feasible and convenient. With centralized scheduling of security resources in resource pool, the utilization of security resources can be increased and the security task execution efficiency can be improved. Furthermore, network security is improved. Therefore, a high performance security resource scheduling algorithm is important.

B. Overview of security resource scheduling algorithm

Traditional security resources scheduling methods are generally based on polling mode or random fashion. Some systems adopt weighted calculation according to security resources' capability to decide which security resource a security protection task is allocated to. However, these security resources scheduling methods do not consider dynamic factors such as execution time of a task and task progress of an executing task. This results in scheduling of security resources not efficiency enough.

For dealing with above-mentioned problems, this paper proposes a novel security resources scheduling algorithm. In the algorithm, we first abstract a security protection task as several its determined parameters according to type of task. Then, we develop a method which is used for forecasting execution time of security protection tasks according to their abstracted parameters. The reason for using the BP neural network will be explained in section '*A. Solution 1*' of '*IV. Evaluation*'. Then, we describe security resources' task execution ability abstractly based on above work. The security resources scheduling algorithm will calculate the task execution ability of each security resource synthetically and allocate a security protection task to the security resource with suitable type and maximum task execution ability. Therefore, security resource utilization and security task execution efficiency can be improved in software defined security architecture.

Realization process of the proposed security resources scheduling algorithm in software defined security architecture is shown in Figure 1. The operation flow is as follows:

1) The security control plane receives security protection request and generates security protection tasks.
2) For each security protection task, security task abstract description module firstly abstracts it into several its determined parameters. Output of this module is a security task abstract description data structure, which is sent to security task amount forecast module and security resources preliminary screening module. The abstract description method of a security protection task will be discussed in detail in '*C. Abstract Description Method of Security Task*'.

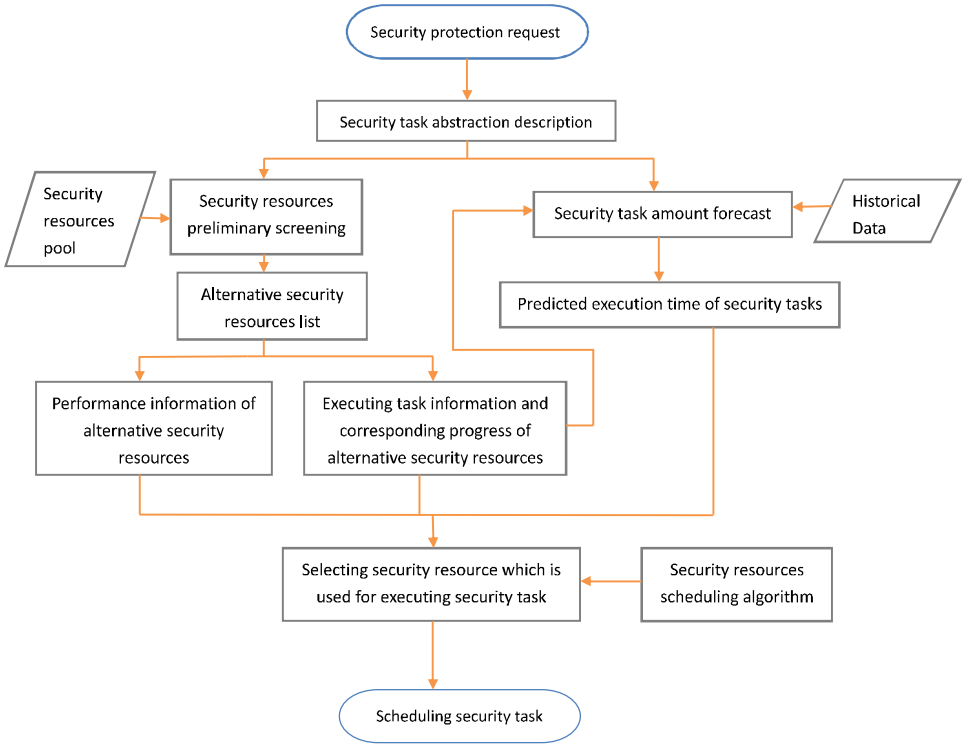

Figure 1. Realization process of security resources scheduling algorithm

3) After extracting the data from security task abstract description data structure, security resources primary screening module selects suitable security resources from security resource pool according to needed security resource type and vendor of security devices. Output is the selected alternative security resources list.
4) Security task amount forecast model forecasts execution time of a security protection task according to values of task type and task execution parameters, which are defined in security task abstract description data structure. The BP neural network, which is used for forecasting task execution time, in this module is trained by historical task execution data in advance. Output of this module is the predicted task execution time of a security protection task, its value will be stored in module 'Predicted execution time of security tasks', and the data in this module will be used in security resource scheduling algorithm later to measure the performance of security devices. The execution time forecast method will be discussed in detail in '*D. Execution Time Forecast Method of Security Task*'.
5) Security resource scheduling module acquires CPU usage, memory usage and each on-going task's execution process of each security resource in the selected alternative security resource list. Combining with the on-going security tasks' predicted execution time, which are forecasted by module 'security task amount forecast', of each security device in the selected alternative security resources list, this module describes each security resource's task execution ability abstractly and calculates their equivalent load. The abstract description method of security resource and calculation method of security resource's equivalent load will be discussed in detail in '*E. Abstract Description Method of Security Resource*'.
6) Selecting the security resource with minimum equivalent load and allocating the corresponding security protection task to this security resource.

C. Abstract Description Method of Security Task

For measuring the complexity degree of a security protection task, it should be described abstractly. In the algorithm, a security protection task is abstracted as several its determined parameters according to task type. These parameters are digitized description of a security task and related to their task amount. Therefore, these parameters can be used for measuring the execution time of security tasks. Following are two examples of the abstract description of two kinds of security business. The first example is web scan security business. Its three determined parameters, which are digitized description of a web scan security task and related to its task amount, are maximum number of webpage that needed to be scanned, scan depth and number of concurrency. The second example is Anti_DDOS security business. Its three determined parameters, which are digitized description of a Anti_DDOS security task and related to its task amount, are protection strategy, protection level and rate-limiting level.

D. Execution Time Forecast Method of Security Task

For measuring task execution capability of a security resource synthetically, we need forecast the execution time of on-going security protection tasks in that security resource. We establish relationship between abstract description parameters of a security task and execution time of that security task through BP neural network. For different type of security business, the system will train BP neural network independently and generate different groups of neural network weights.

The specific steps of task execution time forecast method are as following:

1) Establishing BP neural network module and randomizing node's weights in the map space.
2) Training neural network with historical security business execution data. Each type of security business establishes BP neural network module independently. When training BP neural network, input parameters include abstract description parameters of security tasks and their practical execution time. The relationship is established between abstract description parameters of a security task and its practical execution time.
3) Repeat procedure 2 until weight vector of neural network has no significant change.
4) After training BP neural network, inputting the abstract description parameters of a new security protection task into the trained BP neural network model, output is the predicted execution time of the corresponding task.
5) Each time a task has been executed, its abstract description parameters and practical execution time will be used for incremental training of BP neural network.

E. Execution Time Forecast Method of Security Task

For describing task execution capability of a security resource intuitively, this paper proposes an abstract description method for security resources. Capability of a security resource is determined by two main factors.

The first factor is the real-time physical load, which is calculated by formula (1), of a security resource.

$$\begin{aligned} load = {} & cpu\_usageWeight \times cpu\_usage + \\ & memory\_usageWeight \times memory\_usage \\ & + disk\_usageWeight \times disk\_usage \end{aligned} \quad (1)$$

In formula (1), cpu_usage, memory_usage and disk_usage are real-time utilization of CPU, memory and disk of a security resource respectively. Their values are

acquired from security resource. Cpu_usageWeight, memory_usageWeight and disk_usageWeight are corresponding weight of each parameters, weight values represent the important degree of each parameter when measuring how busy a security resource is. Different security business has different preference for these three parameters.

The other factor is defined as 'time factor', which is related to the remainder execution time of all on-going tasks in a security resource. The remainder execution time of a security task is calculated by task progress and its corresponding predicted execution time.

Capability of security resource is determined by above two factors, and we use 'equivalent_load', which is calculated by formula (2), to describe security resources' capability. In formula (2), factor 'load' and 'time factor' are combined with a simple multiplication for the following three reasons: (i) 'equivalent_load' is proportional to factor 'load' and 'time factor'; (ii) for considering the two factors synthetically; (iii) for reducing computation complexity so that improving computation efficiency.

$$equivalent\_load = [\sum_{k=1}^{n}(1-task\_progress_k)\times Predicted\_Time_k]\times load \quad (2)$$

In formula (2), parameter 'task_progress$_k$', which is acquired from security resource, represents execution progress of task 'k' in a security resource. Parameter 'predicted_time$_k$' represents the predicted execution time, which is forecasted by BP neural network module, of task 'k'.

Lower equivalent load means higher task execution capability, so the algorithm will select the security resource with minimum equivalent load to deal with the corresponding task. In this way, resource utilization of network security resources in the security resource pool can be improved and security task execution efficiency can be improved. Therefore, network security can be improved.

IV. EVALUATIONV

Due to the lack of open security devices and real network environment, we made a simulation to evaluate the performance of the security resource scheduling algorithm.

In order to make the simulation results easy to observe, this paper uses a simple case to verify the proposed algorithm. We take web scan task as an example in this paper, and the simulation environment is set as following:

(1) Number of security task is set as 500. And the arrival of tasks subject to Poisson distribution. Therefore, time interval between adjacent coming tasks is subject to exponential distribution. Task parameters are assumed as Table 1.

(2) For each security resource pool. Number of security resources in it is set as 5. And their initial CPU usage and memory usage is generated as Table 2.

In Table 1, 'i' represents task sequence number. And TaskNew(i,9) = TaskNew(i,6) - TaskNew(i,5), in simulation, practical execution time of task is assumed in proportional to 'TaskNew(i,2)' and introducing randomness to compensate the differences (scan depth, amount of concurrency and complexity of different websites) with practical application.

TABLE 1. TASK PARAMETERS OF SIMULATION

No.	Parameters	Description	Distribution type
1	TaskNew(i,1)	Task ID	——
2	TaskNew(i,2)	Maximum number of webpages	Discrete Uniform Distribution a=1000, b=10000
3	TaskNew(i,3)	Scan depth	Discrete Uniform Distribution a=0, b=10
4	TaskNew(i,4)	Amount of concurrency	Discrete Uniform Distribution a=1,b=10
5	TaskNew(i,5)	Arrival time of task	Exponential Distribution
6	TaskNew(i,6)	End time of task	——
7	TaskNew(i,9)	Practical execution time of task	——
8	TaskNew(i,10)	Predicted execution time of task	——

TABLE 2. GENERATION OF INITIAL CPU USAGE AND MEMORY USAGE

No.	Parameters	Description	Distribution type
1	cpu_usage	Percentage of CPU usage	Continuous Uniform Distribution a=0.2, b=0.3
2	memory_usage	Percentage of memory usage	Continuous Uniform Distribution a=0.2, b=0.4

The CPU occupancy ('TaskNew(i,7)') and memory occupancy ('TaskNew(i,8)') of a task are set as constant value according to experience.

In simulation, 'load' in formula (1) is calculated by parameters in Table 2; 'Predicted_time' in formula (2) is forecasted by BP neural network with parameter TaskNew(i,2), TaskNew(i,3) and TaskNew(i,4) in Table 1 as input; 'Task_progress' in formula (2) is calculated by [Current time - TaskNew(i,5)]/TaskNew(i,10). And the difference of TaskNew(i,9) and TaskNew(i,10) in Table 1 is the back error of BP neural network. The BP neural network is set as three layers: input layer includes three nerve cells, hidden layer includes five nerve cells and output layer includes one nerve cell. Number of hidden layers is set according to experience. Number of iterations is set as 15000. Learning rate is set as 0.1. Target error is set as 0.01.

As shown in Figure 2, the simulation result shows that the complete proposed resource allocation algorithm can realize load balance among security resources. In Figure 2, each different color line represents a security resource's load condition.

Figure 2 shows that with proposed security resources scheduling algorithm, a security protection task will not be allocated to security resource with minimum load but minimum equivalent load. Therefore, we can see from Figure 2, as the increasing of security tasks with different complexity degree, the five security resources' load are relative balanced all the time with proposed high performance algorithm.

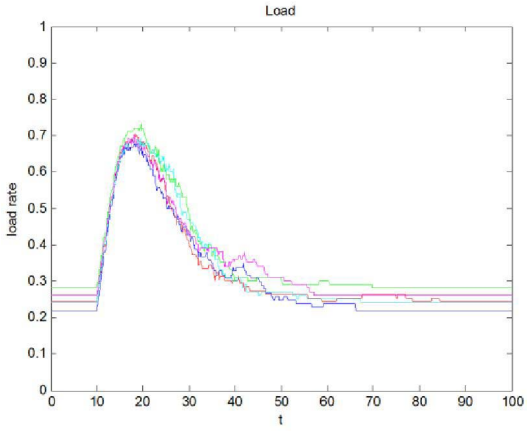

Figure 2. Equivalent load of five security resources

As shown in Figure 3, as the increasing of security tasks with different complexity degree, the five security resources' CPU usages are relative balanced all the time with proposed high performance algorithm.

As shown in Figure 4, the memory usage of each security resource is also relative balanced, but its effect is not as ideal as CPU usage. This is because memory has smaller weight than CPU when considering the composite load of a resource.

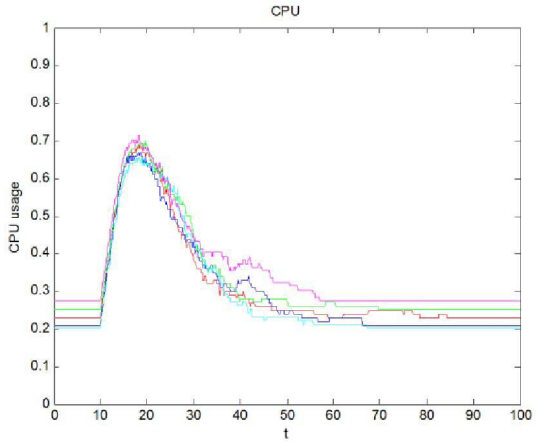

Figure 3. CPU usage of five security resources

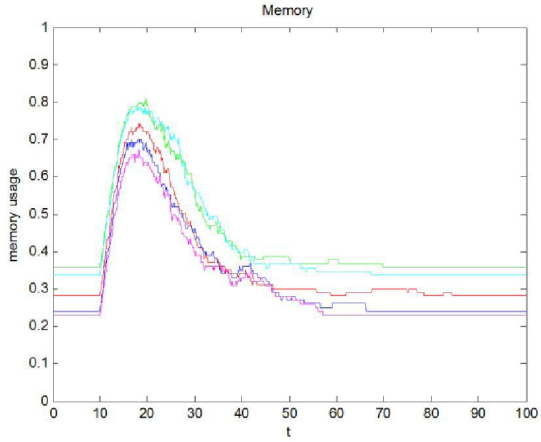

Figure 4. Memory CPU usage of five security resources

As the simulation result shows an ideal performance, we developed the security resources scheduling algorithm in our proposed SDS architecture. The following is an application scenario of security resource scheduling in our SDS architecture.

The software defined security architecture that we proposed has been introduced in 'Introduction'. Based on this architecture, we developed a very important northbound application, which is security business orchestration application. The whole system is shown in Figure 5. Orchestration engine combines independent and different types of security resources in the resource pool effectively through utilizing centralized management and scheduling capability of security controller to security resources. Therefore, security protection efficiency can be improved through alliance of several security resources. For realizing automated and intelligentized security orchestration, uniform scheduling of security resources and analysis of some security threats are needed.

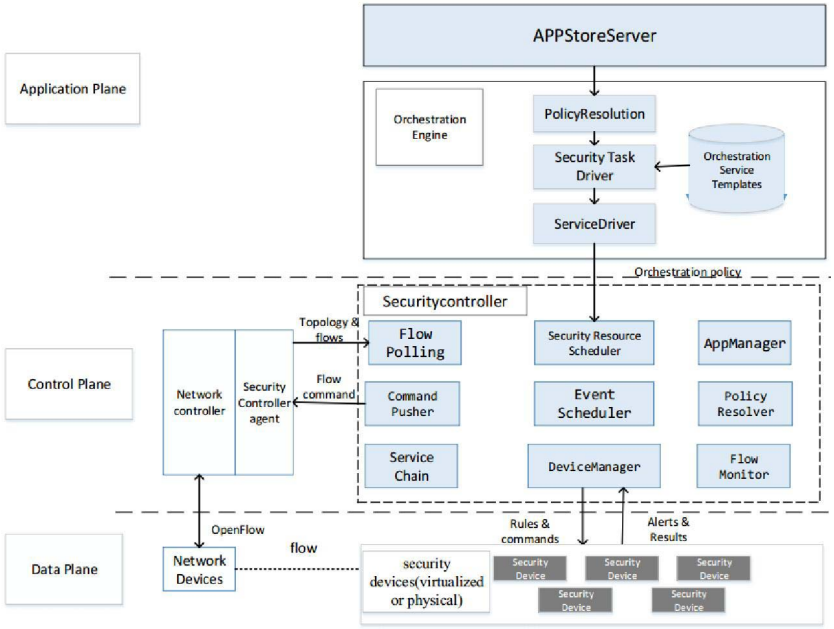

Figure 5. Software defined security architecture

The security resources scheduling algorithm proposed in this paper is deployed in module 'Security Resource Scheduler' in security controller. Users can input the IP address of the host that needed to be protected in APPStore, which is a user interface between users and APPStoreServer. Then, after policy resolution, the vulnerabilities of the host will be detected. According to these detected security vulnerabilities, the orchestration engine will create a series of security orchestration tasks which are sent to security controller by service driver. The security resources scheduling algorithm in it will allocate each task to a suitable security resource according to task type and task execution capability of security resources. The Service Chain Module interacts with SDN controller to make sure network flows are led into security resources sequentially.

TABLE 3. SUPPORTED PARAMETERS OF 'SKIPFISH'

Supported Parameters	Remarks
Authentication and access options	Setting parameters related to http, host&ip, cookie, etc
Crawl scope options	Including max_depth (maximum crawl tree depth)
Reporting options	Setting the format of scan report
Dictionary management options	Setting parameters that related to dictionary
Performance settings	Setting max global simultaneous TCP connections, max simultaneous connections per target IP, etc
Other settings	Setting max_req, duration

However, existing security devices cannot meet the need of high performance resources sharing and scheduling in software defined security architecture because they are not open enough. At here, we list the supported parameters of some typical security devices.

TABLE 4. SUPPORTED PARAMETERS OF 'NIKTO'

Supported Parameters	Remarks
Cgidirs	Catalog of CGI that needed to be scanned
config	Replacing local 'config.txt' with specific config file
dbcheck	Selecting scan database with wrong grammer
evasion	Using avoiding technology for IDS inLibWhisker
findonly	Only finding port of HTTP and HTTPS but not execute detection rule
Format	Appointing file format of output check report
host	Target host (name, IP, host list file)
id	Authentication for competent HTTP by ID and password
maxtime	Maximum execution time per host, in seconds
Tuning	Controlling Nikto to scan target with different mode
port	Assigning scan port
Display	Controlling output display of Nikto

Another security protection application is 'Paros proxy', its supported parameters includes Connection, Local proxy, Authentication, Certificate, View, Trap, Spider and Scanner. Obviously, these options can not reflect the real-time load as well as task execution capability of security resources.

Above material shows that existing security devices (applications) are not open enough for software defined security architectures because business demands are different between traditional network security and SDS. Traditional security devices are not deployed through cloud, so there is no need for considering security resources sharing and scheduling. But SDS emphasizes the openness of security devices and need to consider security resources sharing and scheduling. Therefore, there are always differences when investigating resources scheduling in software defined security architectures with traditional security devices.

For realizing the complete function of the software defined security system, this paper proposes following two solutions.

A. *Solution 1*

In that application scenario, 'Nikto' is adopted as web scanner. For Nikto, just the parameter 'maxtime' can reflect the task amount of a task. Parameter 'maxtime' is used for setting parameter 'amount of concurrency' in our proposed algorithm. For acquiring CPU usage and memory usage of security devices, we installed 'Nikto' in different Ubuntu systems and compute each security resource's CPU usage and memory usage through return value of 'top' command. Therefore, for realizing the complete function of the system, we applied a customized version of the proposed security resource scheduling algorithm according to the parameters that can be acquired from the security devices. In the customized version algorithm, 'time factor' defined in formula (2) will not be considered. This means we make 'equivalent_load' just equal to 'load' for compatible with existing security devices. With the compatible version security resource scheduling algorithm, a task can be allocated to a security resource with specific type and maximum capability. An example of scheduling of web scanner resources is shown in Figure 6, we can see that each time a web scan task will be allocated to the scanner with minimum load. Minimum load means maximum task execution capability, therefore, security protection efficiency can be improved.

Although the customized version of proposed resource allocation algorithm can also realize the expected target which is realizing the load balance among security devices so that improving security protection efficiency. However, one situation may leads to transient unbalance of load. For example, assuming one security resource are executing 30 tasks with average task progress 90%, the other security resource are executing 26 tasks with average task progress 10%, if there are another 10 Web Scan requests this time, these 10 task may be allocated to the first security resource because it has smaller composite load, this will cause load unbalance later and lower task execution efficiency. This may be a serious problem in data center. This defect may be overcome as introducing 'time factor' in the complete proposed algorithm.

What's more, using security devices that are not open enough to form security resource pool, the centralized management and scheduling of security devices in security control plane are difficult to realize.

B. *Solution 2*

As the development of network, SDS is the future development direction of network security. Therefore, it is necessary to create new style open security devices (applications) to meet the demand of software defined security architecture.

When designing virtualized and containerization new style open security devices, which are used for forming into security resource pool, their supported parameters should include the parameters that used to measure the task amount of security tasks. For example, a web scanner should support parameters includes maximum number of webpage that needed to be scanned, scan depth and amount of concurrency, etc. These parameters can reflect the task amount of a security task. Therefore, the security resources scheduling algorithm proposed in the paper can be used in future software defined security architectures so that realizing high performance resources scheduling in security resource pool. In this way, resource utilization of network security resources in the security resource pool can be improved and security task execution efficiency can be improved. Furthermore, network security can be improved.

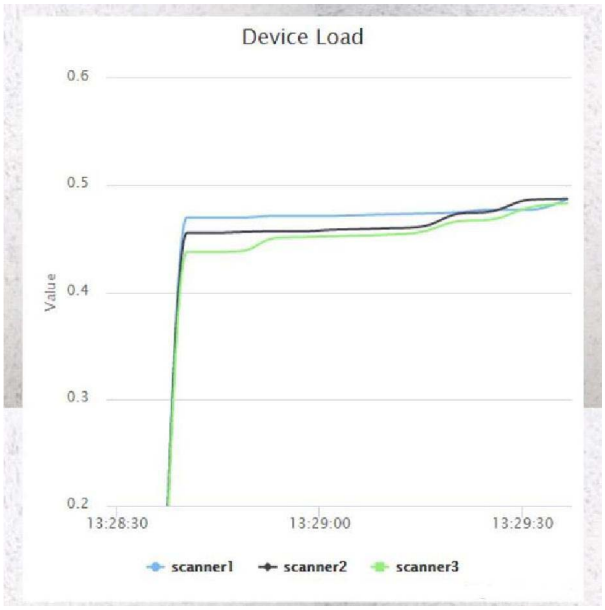

Figure 6. Scheduling of web scanner resources

V. CONCLUSION

This paper analyzes the reasons of putting forward software defined security and introduce three typical software defined security architecture as well as our own software defined security architecture. Based on this, this paper expounds the reason why open security devices are needed in SDS architecture and propose a method to open security devices. Taking advantage of the characteristic of centralized control in Software Defined Security, we designed a security resource scheduling algorithm which is applied in software defined security architecture with security resource pool. The algorithm can allocate a security task to a suitable security resource in the resource pool according to their type and abstracted capability. Therefore, resource utilization of network security resources in the security resource pool can be improved and security task execution efficiency can be improved. Furthermore, network security can be improved

The contributions of this paper are as follows:

1) We proposed a method to open security devices so that centralized management and scheduling of security resources become feasible and convenient
2) We proposed a kind of security protection task abstraction description method.
3) We proposed an execution time forecast method for security task according to abstraction description parameters based on BP neural network.
4) We proposed a security resource capability abstraction description method based on above works
5) We designed a security resource scheduling algorithm which is deployed in software defined security architecture with security resource pool and verified its performance.
6) We proposed parameters design of future new style security devices that applied in software defined security architecture.

As the number of security protection task is huge in cloud security center, a good performance resource scheduling algorithm is important. So, we will deploy the resource scheduling algorithm proposed in the paper when there has open security devices (applications).

ACKNOWLEDGMENT

This work is supported in part by the following grants: National High-tech R&D Program ("863" Program) of China (No. SS2015AA011709), Beijing Laboratory of Advanced Information Networks.

REFERENCES

[1] "The Fortinet Software-Defined Security Framework" https://www.fortinet.com/demand/gated/SDN-Security-Framework WhitePaper.html

[2] http://support.embrane.com/index.php?/Knowledgebase/Article/view/79/13/Download-heleos

[3] "Catbird:Private Cloud Security" http://info.catbird.com/private-cloudsecurity-white-paper.

[4] Wenmao Liu, Xiaofeng Qiu, Pengcheng Chen, Xinxin He, Xutao Wen, Dongsheng Wang, SDSA: a Programmable Software Defined Security Platform, International Conference on Cloud Computing Research and Innovation, CloudAsia2014, Oct 29th-30th, Singapore

[5] Weijia Wang, Xiaofeng Qiu, Li Sun, Rui Zhao."A Data Driven Orchestration Framework in Software Defined Security".IEEE International Conference on Network Infrastructure and Digital Content,2016

[6] A. Darabseh, M. Al-Ayyoub, Y. Jararweh, E. Benkhelifa, M. Vouk and A. Rindos, "SDSecurity: A Software Defined Security experimental framework," *2015 IEEE International Conference on Communication Workshop (ICCW)*, London, 2015, pp. 1871-1876

[7] LIUWenmao, QIU Xiaofeng, CHEN Pengcheng, WEN Xutao, HE Xinxin, WANG Dongsheng, LI Jun. SDN Oriented SoftwareDefined Security Architecture [J].Journal of Frontiers of Computer Science and Technology. 2015. 9.

[8] L. Yanbing, L. Xingyu, J. Yi and X. Yunpeng, "SDSA: A framework of a software-defined security architecture," in *China Communications*, vol. 13, no. 2, pp. 178-188, Feb. 2016.

[9] Feng T, Bi J, Wang K. Joint allocation and scheduling of network resource for multiple control applications in SDN[M]. 2014.

[10] Wang W, He W, Su J. Network intrusion detection and prevention middlebox management in SDN[C]// IEEE, International PERFORMANCE Computing and Communications Conference. IEEE, 2015:1-8.

[11] Hong W, Wang K, Hsu Y H. Application-Aware Resource Allocation for SDN-based Cloud Datacenters[C]// International Conference on Cloud Computing and Big Data. 2013:106-110.

[12] Maguluri S T, Srikant R. Scheduling Jobs With Unknown Duration in Clouds[J]. Networking IEEE/ACM Transactions on, 2014, 22(6):1938-1951