Design of a Security Service Orchestration Framework for NFV

Hu Song[1], Qianjun Wu[2], Yuhang Chen[2], Meiya Dong[3(✉)], and Rong Wang[4]

[1] State Grid Jiangsu Electric Power Co., Ltd. Information and Communication Branch, Nanjing, China
[2] Information System Integration Company, NARI Group Corporation, Nanjing, China
[3] Taiyuan University of Technology, Taiyuan, China
dongmeiya@163.com
[4] China Energy Investment Corporation, Beijing, China

Abstract. Network functions virtualization (NFV) emerges as a promising network architecture that separates network functions from proprietary devices. NFV lowers the cost of hardware components and enables fast and flexible deployment of network services. Despite these advantages, NFV introduces new security challenges. Currently, there is little research on a holistic framework to solve these security issues. In this paper, we propose a security service orchestration framework that can construct a cooperative working mechanism for NFV security. We present the demand analysis and describe the system design principles and implementation details. We propose a lightweight holistic architecture design of the security service orchestration system to solve current security issues. The system's effectiveness is also shown based on technical review.

Keywords: NFV · Network security

1 Introduction

NFV (Network Functions Virtualization) [11] is an emerging technology that decouples network functions from dedicated devices and thus provides an agile and cost-efficient way to deploy network functions and services. The advantages of NFV have been recognized by both industry and academia [7,15,16]. Since it can reduce the cost of ownership by implementing network services on commercial off-the-shelf hardware, telecommunication giants such as AT&T are embracing NFV architecture and providing business solutions based-on NFV [1,2]. According to a recent report [3], the global NFV market size is projected to reach 36.3 billion USD by 2024.

However, the successful deployment of NFV inevitably introduces new challenges in network security management. For example, since NFV turns network functions into software modules, an attack on a software module might affect

J. Liu et al. (Eds.): MobiCASE 2020, LNICST 341, pp. 37–52, 2020.
https://doi.org/10.1007/978-3-030-64214-3_3

others located on the same virtual machine. In addition, many NFV systems are built on open-source projects such as Openstack and OSM. Potential software bugs in these projects might lead to security threats [4].

In recent years, much efforts have been made to improving NFV security [8,9,18,20]. Basile et al. [5] proposed a software component named Policy Manager, which allows users to specify their security requirements and automatically selects required virtual network functions based on policy refinement techniques. In [6], the authors further added support for automatic enforcement of security policies as a part of the Orchestrator, and designed an optimization model which was able to select the best way to refine network policies. Marchetto et al. [13] proposed a network modeling approach for formal verification of forwarding behavior. Pedone et al. [17] designed a security framework to protect end-users which is compliant to the Security-as-a-Service paradigm. While all these studies focus on solving NFV security issues, a holistic framework is still in need.

To this end, we aim to design software-defined security architecture and provide a systematic framework to ensure security in NFV in this paper. It can reasonably allocate virtual security device resources and deliver various security services to improve adequate security protection to achieve intelligent security services. The rest of this paper is organized as follows. In Sect. 2, the overall architecture and requirements analysis of NFV are discussed. In Sect. 3, the architecture design of the security service orchestration system is analyzed, and an overall NFV security service orchestration architecture is proposed. Section 4 concludes the full paper.

2 NFV Architecture

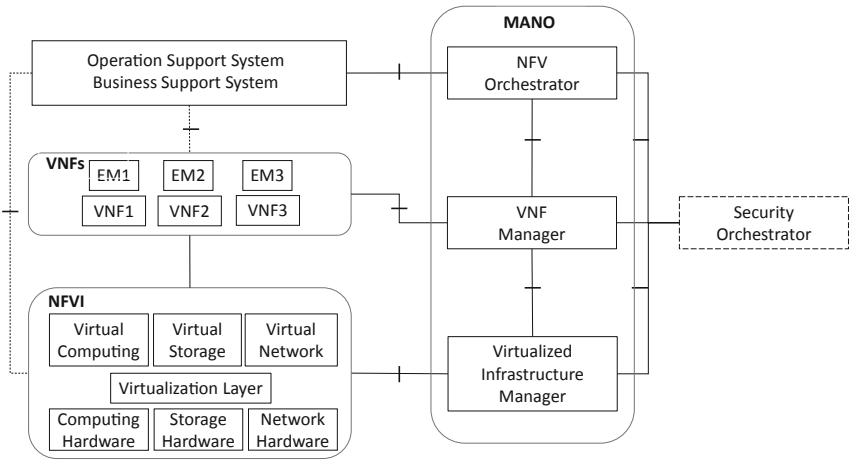

Fig. 1. NFV architecture

As shown in Fig. 1, an NFV architecture consists of Network Functions Virtualization Infrastructure (NFVI), Virtualized Network Functions (VNFs), and Management and Network Orchestration (MANO).

- NFVI is built on the basis of non-proprietary hardware like x86 servers. It abstracts computing hardware, storage hardware, and network hardware into virtual resources, and provisions these resources to support the executions of VNFs.
- VNFs are virtualization of network functions. They are deployed and running on virtual machines. Element Management Systems (EMs) are responsible for fault, configuration, accounting, performance, and security monitoring VNFs.
- MANO are composed of Virtualized Infrastructure Manager (VIM), VNF Manager, and NFV Orchestrator (NFVO). VIM controls and manages computing, network, and storage resources. VNF Manager is responsible for the lifecycle management of VNF instances. And NFVO performs global resource management across multiple VIMs and maintains network services of VNFs.
- Operation Support System (OSS) and Business Support System (BSS) are a collection of system management applications, supporting network management, configuration management, customer management, etc.

In this paper, we also propose a Security Orchestrator as part of the framework. It can both connect these functions and make real-time security assessment on the process of data flow of the whole NFV system.

2.1 Demand Analysis

The design of security orchestration system for NFV should include the following components:

(1) Web business layer: The front-end display of the orchestration system is the entrance to the interaction between the orchestration system and the user, and it is also the generator of the orchestration strategy. The Web layer can generate corresponding orchestration strategies according to the actual needs of users, and deliver them to the orchestration engine.
(2) Application development standards: In the orchestration system, security devices of different manufacturers are accessed through a proxy App. The agent App northbound interacts with the orchestration engine by following the "standard", while the southbound converts the "standard" into device interface data to achieve the adaptation of the device to the orchestration system. Developers develop applications under standards. Generally speaking, the development standard can be either a software development SDK or an API interface specification. This system uses the API interface specification.
(3) Orchestration strategy: The orchestration system needs to provide a set of description specifications of the orchestration logic, which is convenient to transform the orchestration scenario into a program parseable orchestration strategy.

Table 1. Demand analysis

Functional requirement	Explain
Web Business Layer	Web services for orchestration system based on App Store project development
Layout engine	Parse the security policy under the Web, and realize the task generation, scheduling and execution
Safety Controller	equipment management module, resource scheduling module, service chain module
Virtual security equipment management platform	Virtual security device startup, registration, network configuration and device information query interface
Application development standards	Develop development standards for proxy applications of security equipment for vendor adaptatio
Organization strategy	Provides a description mechanism to translate choreographed scenarios into standard security policies

(4) Orchestration engine: The orchestration engine is the core module of the orchestration system. Its main functions include orchestration strategy analysis, orchestration task scheduling, and execution.
(5) Orchestration scheduling module: The orchestration scheduling module is located inside the security controller to implement the scheduling function of security resources. In order to improve the effective utilization of safety resources, the centralized scheduling of safety resources needs to adopt a suitable resource scheduling algorithm. The input of the resource scheduling algorithm includes various security policy requirements issued by the application plane on the one hand, and the security resource information obtained from the data plane on the other. The security policy of the application plane can reflect the security needs of users, and the data of the data plane can reflect the actual situation of security resources. If the requirements of the application plane and the security resource capabilities of the data plane can be reasonably matched using the resource scheduling algorithm, it can improve the utilization rate of security equipment, reduce the queuing time, and speed up the system's response to security threats.
(6) Virtual security device management platform: It effectively manages various virtual security devices under a virtual network environment. In the security service orchestration system, it is responsible for the startup and configuration of the underlying security device and provides the device status query interface and device configuration interface to the security controller (Table 1).

3 Security Service Choreography System Architecture Design

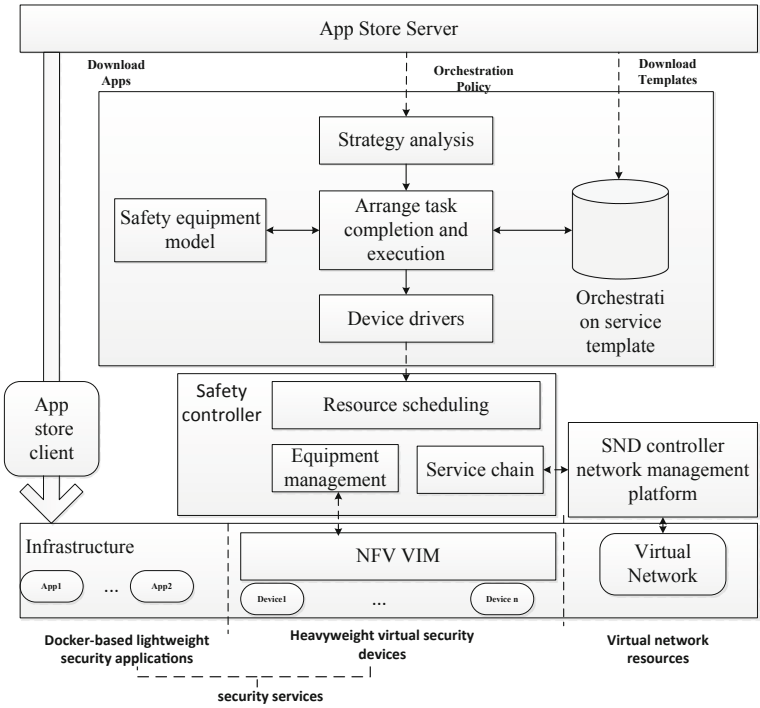

Fig. 2. Overall framework of security service orchestration system architecture

Based on previous analysis of the requirements for the security service orchestration system, we divide the orchestration system into four major parts for design and development, namely the web business layer, the orchestration engine, the relevant modules within the security controller, and the virtual security device management platform. Among them, the relevant internal modules of security control include equipment management module, a resource scheduling module, and a service chain module. This topic implements the service chain module in general scenarios (Fig. 2).

The top layer of the architecture is a web business layer developed based on the AppStore platform, which is generally deployed on a central server to provide users with web interaction services and generate orchestration strategies. Below the Web layer is the client environment. The deployed modules include orchestration engines, security controllers, and infrastructure layers and services. The orchestration engine is responsible for receiving orchestration policies and orchestration service templates issued by the AppStore, generating orchestration tasks,

and scheduling execution. The security controller receives the resource invocation request of the orchestration engine and selects appropriate equipment from the resource pool to perform the protection task through the resource scheduling module. The infrastructure layer provides the computing, storage, and network resources required for security services to operate. The service chain module of the security controller is interfaced with the SDN control platform to realize the secure service chain and provide a guarantee for the correct implementation of the orchestration strategy at the bottom.

3.1 System Workflow

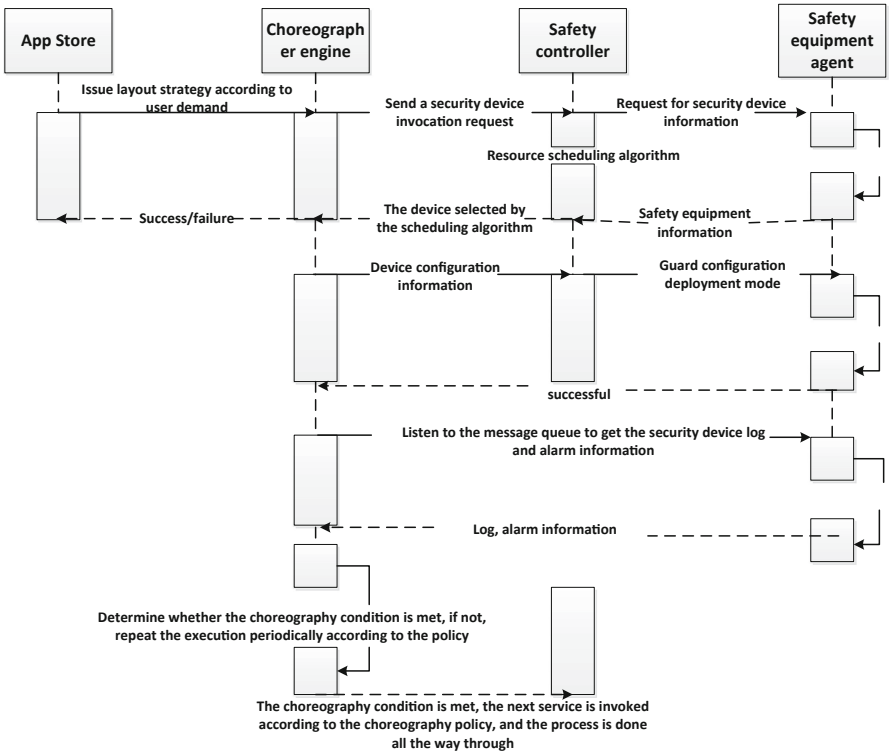

Fig. 3. Workflow of the security service orchestration system

Firstly, AppStore recommends related security orchestration scenarios to users based on their asset information. After the user selects and customizes the orchestration method of the security service, the AppStore sends the corresponding orchestration strategy to the orchestration engine. Then, the orchestration engine analyzes the orchestration strategy and generates a Job for it (persistent task). Finally, the orchestration engine executes according to the scheduling method

specified by the policy. The orchestration strategy supports flexible job execution methods, such as executing once every 5 min, keeping the task all the time, or stopping the task after executing it several times (Fig. 3). The following details the job execution process:

(1) In the beginning, the orchestration engine calls the first security application of the orchestration scene according to the orchestration strategy. If the application itself can complete the security service, the security task execution result is returned directly to the orchestration engine.
(2) If the security service requires a security device to implement, the security application sends a device call request to the security controller. Request parameters include protection target information, security device type, and device configuration parameters.
(3) The safety controller selects the appropriate safety equipment through the resource scheduling algorithm according to the safety equipment type and other business parameters and sends a protection task to the selected device.
(4) The safety controller collects the task execution results of the safety equipment and returns the information to the safety application. The security application further processes the collected logs and alarm information, and converts it into an interface format recognized by the orchestration engine, then returns it to the orchestration engine.
(5) Finally, the orchestration engine compares the information returned by the security application with the orchestration strategy. If the triggering condition is met, the next security application is called for protection according to the orchestration strategy. If the triggering condition is not met, the first one is called again according to the task scheduling method defined by the orchestration policy security application.

3.2 AppStore Layer

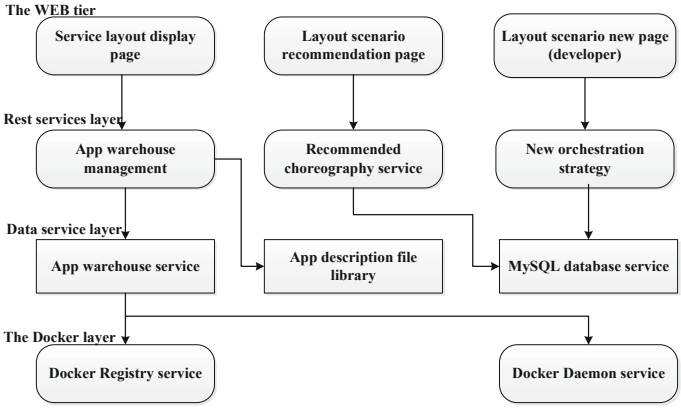

Fig. 4. AppStore cloud architecture

The web business architecture of the orchestration system is shown in Fig. 4. The AppStore layer is divided into four sublayers, from bottom to top: Docker sublayer, data service sublayer, REST service sublayer, and WEB sublayer. Among them, the Docker sublayer is used as the underlying operating environment. The AppStore platform uses Docker container technology to package the App and provides services such as Docker image management and container management. The data service sublayer implements data persistence and saves the orchestration strategy information that has been issued. The REST service sublayer implements the encapsulation of business logic and provides REST API to invoke. The WEB sublayer provides interface-related UI for smooth user operation.

AppStore is the entrance to various network security services. Currently, it is mainly a web security scan. After selecting the corresponding security service, it fills in the relevant parameters to issue the security service. This implementation case is a web vulnerability scan. In the security controller, several main modules used are App Managers, which are used to manage the supported applications. Device Manager manages various information of the currently enabled devices, such as IP address ports and service types. EventManager is the control of internal interactive events, and the multiple modules interact through subscription push events. BootAgent controls the startup and recycling of virtual machines, and the resource pool is the server resource cluster where web vulnerabilities are located.

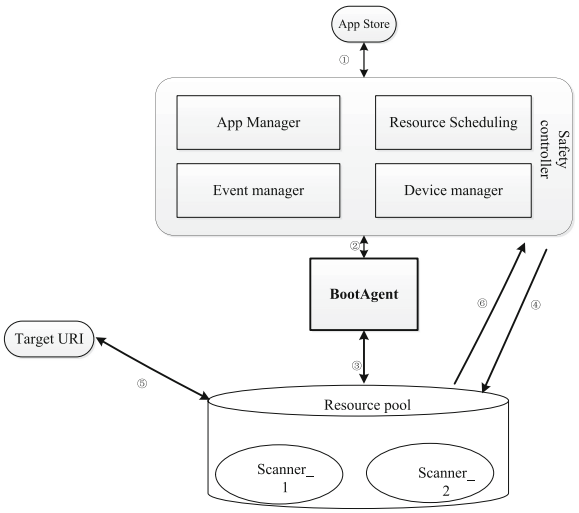

Fig. 5. Resource pool

In Fig. 5, the infrastructure resource pool of the architecture can generally reach the scale of thousands of servers. With the introduction of virtualization

technology, the size of server nodes (VMs) will further increase; meanwhile, the cloud computing resource pool improves resource utilization. The data traffic generated by server resources of the same size is bound to be more potent than traditional construction methods.

Data Center Network Hierarchical Model. With the introduction of NFV, the physical resources at the infrastructure layer are pooled into virtual resources to provide software functions for the business layer. A data center based on the NFV architecture should have sufficient bandwidth resources and various network facilities. In order to ensure the normality of virtual network elements. In operation, the internal management mechanism seems particularly relevant. The designed hierarchical model of the data center network is shown in Fig. 6.

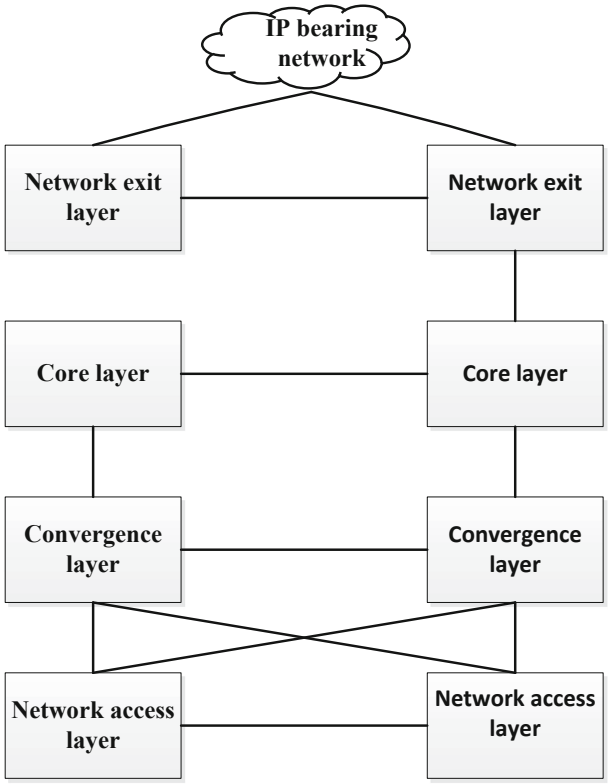

Fig. 6. Layered model of data center network

Data centers should ensure the flexible division of network areas. They are constructing an NFV-based data center that meets the data center design principles. Different application systems can be divided into different areas.

There must be a core layer that is responsible for core switching functions. It is the core part of the entire network and acts as a transmission bus. Compared with other levels, the features of the system are enhanced by directly adding equipment at the core switching layer to improve scalability and availability. At the same time, the access device must be connected to the network access layer to complete the function of the device accessing the network. An aggregation layer is added between the network access layer and the core layer to complete the aggregation function of the access device. Finally, externally accessed resources need to increase the network exit layer, and at the same time, add network firewalls, load balancing, and other equipment to the data center inside and outside. This layered design can not only ensure the security of each region, increase flexibility, but also have good scalability.

In order to ensure that the network elements deployed in the data center can communicate securely and follow the principles of flexibility, reliability, and scalability, the network is divided into network exit layers, core layers, aggregation layers, and network access layers using horizontal partitioning and vertical layering Four levels, as shown in Fig. 7:

(1) The network exit layer is used to connect the internal network and the external network. At the same time, it can play the function of internal and external network information conversion and control the information forwarding of the internal and external network.
(2) The core layer is connected to the network exit layer and connected to the convergence layer, which mainly implements the core switching equipment information forwarding and control.
(3) The aggregation layer is connected to the core layer and connected to the aggregation layer, which mainly implements the aggregation function of the access layer equipment.
(4) The service access layer mainly provides network access functions to ensure regular access to terminal equipment. Considering the high reliability, high availability, and security of the network, the deployment plan is as follows: Configure VLAN + VRF on the core switch to achieve isolation from other business systems and configure QOS to meet RCS (converged communication) service bandwidth requirements.

In the server's internal virtualization layer, the virtual switch (VSW) function should be supported to meet the requirements of throughput, CPU, and memory utilization, to achieve virtual machine switching, VLAN differentiation, and port speed limit functions.

3.3 Orchestration Engine Design

The top layer of the Orchestration Engine is the REST API module, which provides orchestration policy registration/deregistration interface, registered orchestration policy query interface, and Job query interface to the AppStore platform. The middle layer is the core function module of the orchestration engine, which

implements the orchestration strategy analysis and orchestration task scheduling functions. Among them, the security device model is a high abstraction of security devices by the orchestration engine, and each type of security device corresponds to a device model in the engine. The orchestration service template defines the mapping relationship between the input and output of two apps, which can be dynamically loaded by the orchestration engine and is the critical design to break the barriers between applications. The lowest level is the database module and the security service driver module. The database module is used to store the registered orchestration strategies and job information to achieve data persistence. The security service driver is used to connect the orchestration engine with different manufacturers and different types of security devices. It is independent of the orchestration engine in the form of App (agent application). The agent application is developed in the north direction according to the SDS equipment standard API interface specification and receives the security protection tasks issued by the orchestration engine. The south direction interfaces with the security controller and is used to issue protection dispatch information for equipment dispatch requests and equipment identification (Fig. 8).

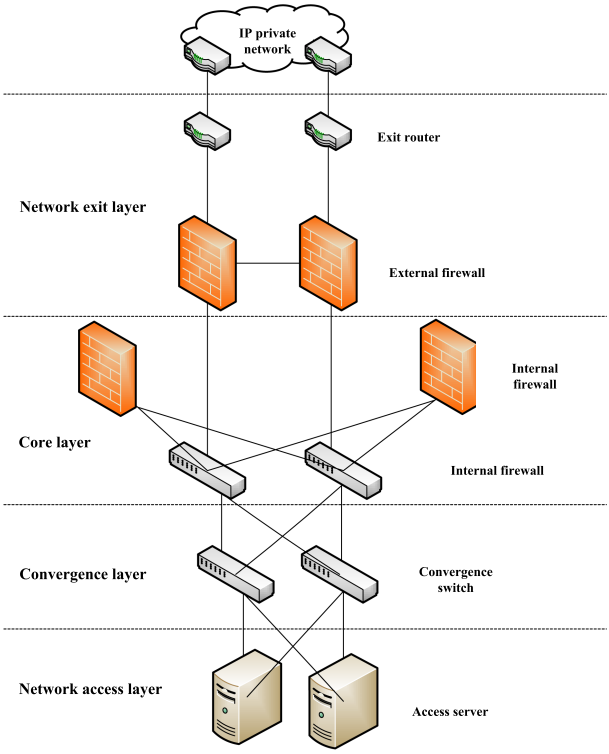

Fig. 7. Data center network layering scheme

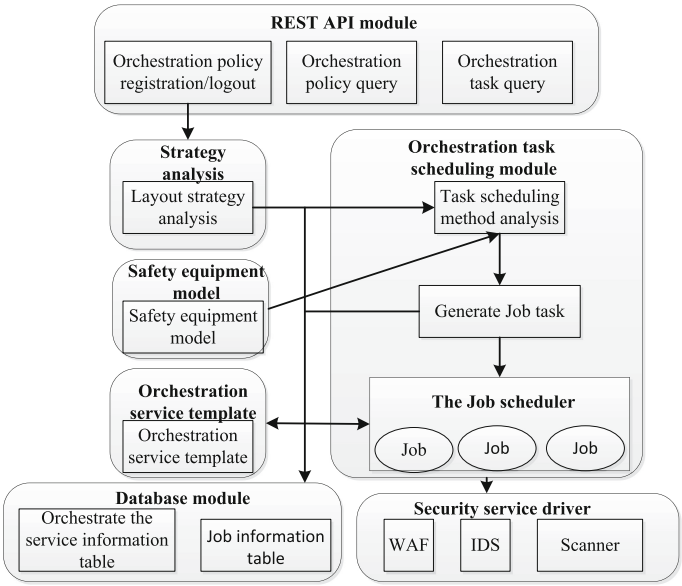

Fig. 8. Orchestration Engine Design Framework

3.4 Virtual Security Management Platform

The virtual security device management platform is written in Python and combines open source technologies such as DNSmasq, libvirt, OpenvSwitch, and Linux namespace. In this section, we explain the solution of this system from

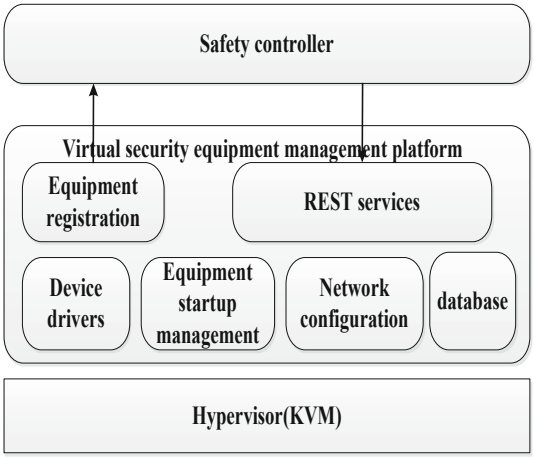

Fig. 9. Virtual Security Management Platform

the perspective of device startup and registration, device network management, and database selection (Fig. 9).

(1) Device startup and registration
In the software-defined security architecture, the security resource pool must shield the differences between different manufacturers and different types of devices, and at the bottom layer, it must implement unified management of device startup and registration. Since security devices are all virtualized in the security resource pool, the management of security devices is equivalent to the control of virtual machines. Establish a virtual security device model that is as general as possible. If a device is different from the general model, the user can implement personalized functions in subclasses by inheriting the general class. In other words, the management platform's access to new equipment is implemented by adding "plug-in classes." In addition, when the management platform is started, it reads the configuration file information, starts a number of security devices in advance, and uniformly registers with the security controller. The configuration file contains the type, number, image information, and drive information of the boot security device.

(2) Equipment network management
The equipment management platform network topology design is shown in Fig. 10. In the picture, WAF (Web Application Firewall) and RASS (Remote System Evaluation System) are virtual security devices, each with three network cards, which are the management port, data stream entrance, and data stream exit. Correspondingly, these three network cards are respectively connected to three OVS (OpenvSwitch) bridges of br-con, br-in, and br-out. Among them, the management port is used to transmit management layer data, and the data flow in and out is used to transmit network traffic. The above two security device deployment modes are also common deployment modes of the management platform. Since the device does not have a business request when it is initially started and does not require an external network to be accessible, it is not necessary to assign an external network IP to the device at startup, and only the connectivity between the management platform and the device management port can be ensured. Based on the above considerations, a port on the control bridge br-con is ba-DHCP-if, and this port is bound to the DHCP server service to provide local DHCP for the virtual security device. In project realization, DNSmasq is used in this project. DNSmasq is a small and convenient tool for configuring DNS and DHCP, suitable for small networks. The address and related commands assigned by DHCP can be configured to a single host or to a core device (such as a router). DNSmasq supports both static and dynamic DHCP configuration methods. This project uses the dynamic DHCP method, and the interval of the IP address pool to be allocated is 120.0.0.1/24. Ba-router is a Linux network namespace, which is a network namespace. It is used to implement NAT (Network Address Translation) between the virtual security device network and the host network, that is, the IP address segment allocated by the local DHCP service. NAT forwarding enables virtual security devices to connect to external

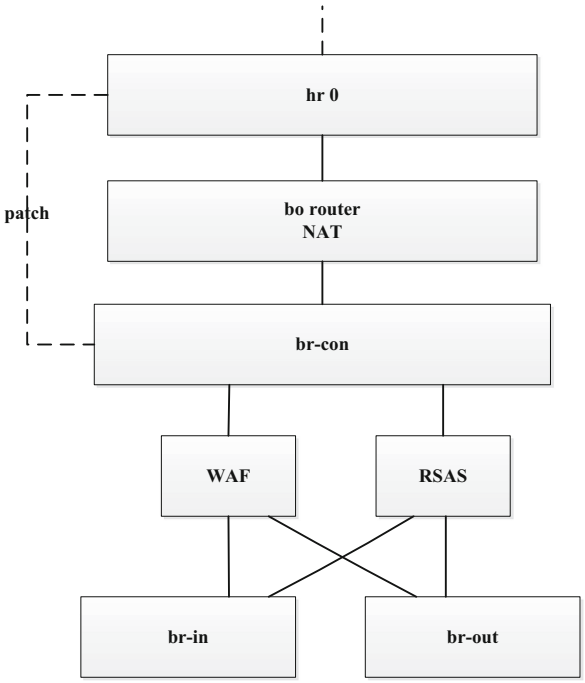

Fig. 10. Management platform network topology

networks. In business, the virtual security device does not have a protection task at the initial startup and does not require external network access [14].

However, when many commercial virtual security devices are launched, they will first go to the cloud of the security vendor to verify the validity of the local license, and then decide whether to provide routine security services. Therefore, NAT in this solution is to ensure that the virtual security device can communicate with the cloud regularly when it is started.

The brO in the topology is a bridge connected to the external network. br-con and brO need to be connected in a patch mode to ensure that after the virtual security device assigns an external network IP, external access traffic can correctly reach the virtual security device.

(3) Database

The information that the device management platform needs to store is mainly device information, and the database selected by this system is Redis. Redis is a remote in-memory database. It not only has stable performance but also has replication characteristics and a unique data model for problem-solving. Redis supports five different types of data structures, and many problems can be stored directly without data conversion.

Besides, through replication, persistence, and client sharding, users can easily extend Redis into a system that can contain hundreds of GB of data and process millions of requests per second.

4 Conclusion

NFV is considered as a foundation of modern networking [19]. It can support various applications such as Internet of things [10,12] and mobile computing [21]. Based on software-defined security architecture and NFV technology, this paper designs and implements a security service enhancement system. The innovation lies in building a bridge between security services, changing the delivery mode of cloud security services, and transforming from the delivery of a single security service to the delivery of multiple security services in a coordinated security protection scheme. The change of the delivery mode not only lowers the threshold for using security services and improves the user experience, but also enhances the security protection efficiency and resource utilization of the cloud platform, and reduces the cost of cloud computing vendors. The orchestration system in this article is deeply integrated with software-defined security technology, maximizing the advantages of software-defined security technology, shielding the differences between various security vendors' devices in the data plane, and weakening the intricate technical details of the underlying system. With which the orchestration engine can focus on the processing of high-level orchestration logic.

Acknowledgement. This research is supported by the Science and Technology Projects of State Grid Jiangsu Electric Power Co., Ltd. (J2019123), NSFC Project No. 61772358 and NSFC Project No. 61572347.

References

1. AT&T Embraces Network Functions Virtualization and May Open Source its NFV Platform. http://alturl.com/mzsyy
2. AT&T FlexWare. http://alturl.com/2c6ev
3. Network Function Virtualization (NFV) Market. http://alturl.com/xps2u
4. Openstack : Security Vulnerabilities. https://www.cvedetails.com/vulnerability-list/vendor_id-11727/Openstack.html
5. Basile, C., Lioy, A., Pitscheider, C., Valenza, F., Vallini, M.: A novel approach for integrating security policy enforcement with dynamic network virtualization. In: Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft), pp. 1–5. IEEE (2015)
6. Basile, C., Valenza, F., Lioy, A., Lopez, D.R., Perales, A.P.: Adding support for automatic enforcement of security policies in NFV networks. IEEE/ACM Trans. Netw. **27**(2), 707–720 (2019)
7. Basta, A., Kellerer, W., Hoffmann, M., Morper, H.J., Hoffmann, K.: Applying NFV and SDN to LTE mobile core gateways, the functions placement problem. In: Proceedings of the 4th Workshop on All Things Cellular: Operations, Applications, and Challenges, pp. 33–38 (2014)

8. Farris, I., Taleb, T., Khettab, Y., Song, J.: A survey on emerging SDN and NFV security mechanisms for IoT systems. IEEE Commun. Surv. Tutor. **21**(1), 812–837 (2018)
9. Firoozjaei, M.D., Jeong, J.P., Ko, H., Kim, H.: Security challenges with network functions virtualization. Future Gener. Comput. Syst. **67**, 315–324 (2017)
10. Gong, W., et al.: Fast and adaptive continuous scanning in large-scale RFID systems. IEEE/ACM Trans. Netw. **24**(6), 3314–3325 (2016)
11. Hawilo, H., Shami, A., Mirahmadi, M., Asal, R.: NFV: state of the art, challenges, and implementation in next generation mobile networks (vEPC). IEEE Netw. **28**(6), 18–26 (2014)
12. Liu, H., Gong, W., Miao, X., Liu, K., He, W.: Towards adaptive continuous scanning in large-scale RFID systems. In: IEEE INFOCOM 2014-IEEE Conference on Computer Communications, pp. 486–494. IEEE (2014)
13. Marchetto, G., Sisto, R., Valenza, F., Yusupov, J.: A framework for verification-oriented user-friendly network function modeling. IEEE Access **7**, 99349–99359 (2019)
14. Naudts, B., Flores, M., Mijumbi, R., Verbrugge, S., Serrat, J., Colle, D.: A dynamic pricing algorithm for a network of virtual resources. Int. J. Netw. Manag. **27**(2), e1960 (2017)
15. Ordonez-Lucena, J., Ameigeiras, P., Lopez, D., Ramos-Munoz, J.J., Lorca, J., Folgueira, J.: Network slicing for 5G with SDN/NFV: concepts, architectures, and challenges. IEEE Commun. Mag. **55**(5), 80–87 (2017)
16. Palkar, S., et al.: E2: a framework for NFV applications. In: Proceedings of the 25th Symposium on Operating Systems Principles, pp. 121–136 (2015)
17. Pedone, I., Lioy, A., Valenza, F.: Towards an efficient management and orchestration framework for virtual network security functions. Secur. Commun. Netw. **2019** (2019)
18. Reynaud, F., Aguessy, F.X., Bettan, O., Bouet, M., Conan, V.: Attacks against network functions virtualization and software-defined networking: state-of-the-art. In: 2016 IEEE NetSoft Conference and Workshops (NetSoft), pp. 471–476. IEEE (2016)
19. Stallings, W.: Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud. Addison-Wesley Professional, Boston (2015)
20. Yang, W., Fung, C.: A survey on security in network functions virtualization. In: 2016 IEEE NetSoft Conference and Workshops (NetSoft), pp. 15–19. IEEE (2016)
21. Zhao, Y., Li, J., Miao, X., Ding, X.: Urban crowd flow forecasting based on cellular network. In: Proceedings of the ACM Turing Celebration Conference-China, pp. 1–5 (2019)