2012 32nd International Conference on Distributed Computing Systems Workshops

LiveSec: Towards Effective Security Management in Large-scale Production Networks

Kai Wang*†, Yaxuan Qi†, Baohua Yang*†, Yibo Xue†‡, and Jun Li†‡
* *Department of Automation, Tsinghua University, Beijing, China, 100084*
† *Research Institute of Information Technology, Tsinghua University, Beijing, China, 100084*
‡ *Tsinghua National Lab for Information Science and Technology, Beijing, China, 100084*
{*wang-kai09, ybh07*}*@mails.tsinghua.edu.cn*, {*yaxuan, yiboxue, junl*}*@tsinghua.edu.cn*

***Abstract*—Network security has become an increasingly important yet challenging issue in present production networks. State-of-the-art solutions cannot meet the overall requirements of high-efficiency security, due to the complicated configuration demands, heavy network traffic and ever-increasing network scale. In this paper, we present LiveSec, a scalable and flexible security management architecture, which achieves holistic security protection with good scalability and flexibility in large-scale networks. LiveSec employs a new Access-Switching layer to provide: 1) interactive policy-enforcement that enables fine-grain control for the end-to-end traffic of network tenants or users, 2) distributed load-balancing that dynamically dispatches security workload over incrementally-deployed security service elements, 3) application-aware network visualization that helps to identify and locate security events, via live traffic monitoring and historical traffic replay. LiveSec has been deployed in Tsinghua University since December 2010. Currently, we are successfully supporting more than 50 users simultaneously (wireless and wired), and over 200 VM-based service elements.**

***Keywords*-network management; network architecture; security; OpenFlow;**

I. Introduction

Production networks, e.g. data center networks and enterprise networks, have become an integral part of the Internet fabric today. Typically, large-scale production networks consist of thousands of commodity PCs, hosting a wide variety of applications and protocols under strict reliability and security constraints [1]. This makes the security management in production networks very complicated. Recent studies show that the correctness of manual configuration for the distributed networking equipments is essential to keep the entire network working successfully [2]. However, with the expansion of the scale, production networks ever-increasingly represent a high-demand yet challenging environment for correct and effective security management, mainly in the following three aspects:

• *Holistic security protection.* Regarded as the key infrastructure to provide services to multiple tenants or users, production networks highly desire holistic security solutions [3], to protect the network from both inside and outside malicious behaviors, and keep the network services always-on to guarantee network tenants or users against any data loss.

• *Scalable security performance.* To process the massive data delivery by data-intensive network services in real time, production networks extremely pursue wire-speed security performance, to follow the rapid development of network bandwidth. Hence the scalability is also distinctly important.

• *Flexible security management.* The rapid growth of the network scale, plus the ever-evolving networking equipments and the protocols, dramatically raise the network complexity and administrating difficulty, which further drives the security management both error-prone and expensive. Thus the demand for easy and efficient security management is ever-increasing.

In terms of these requirements, existing network architecture in production networks suffers from a series of limitations as follows.

• *Poor end-to-end security coverage.* Security middlebox, e.g. Firewalls and Intrusion Detection Systems (IDS), are generally deployed at the gateway of production networks. In order to protect between every pair of work zones (or even hosts) inside the network, more standalone security middleboxes have to be assigned in the end-to-end physical data paths. Furthermore, complicated policies are required to be enforced for coercing end-to-end flows to traverse specified middlebox. Both of these mechanisms are not only inflexible and costly, but also inefficient and too complicated for the security management tasks.

• *Single point of performance bottleneck.* Due to existing concentrated management architecture, production networks have to employ high-performance security middleboxes to fit the bandwidth of backbone network, which results in high cost and limited performance scalability. Even worse, failures are more often to happen at security middleboxes in overloaded work zone (single point failure), while the security resources will be wasted for those middleboxes in idle work zone.

• *Decentralized and complicated management.* For different work zones, network tenants or users, decentralized management needs to enforce various and complicated policies to control the networking and data delivery work as required. However, deploying security middleboxes in

1545-0678/12 $26.00 © 2012 IEEE
DOI 10.1109/ICDCSW.2012.87

modern networks further increases the complexity. Yet most existing solutions require substantial manual configuration by operators to achieve even moderate security. Distinctly, it is hard to eliminate misconfiguration. Various problems can arise to decrease the network reliability. On ther other hand, the locating of these problems is also quite difficult.

To address these issues, we present *LiveSec*, a scalable and flexible architecture for effective security management in modern production networks. LiveSec employ an Access-Switching layer to support fine-grain control to the end-to-end flows of network tenants or users. To the best of our knowledge, LiveSec is the first that considers incremental extensions based on existing network environment to achieve a clean-slate solution, rather than exploring a fork-lift replacement of the entire networking infrastructure from the ground up. With LiveSec, security management problems in production networks can be solved from:

• *Full-mesh security.* Security middleboxes can be added into production networks as service elements through the Access-Switching layer, and applied for any end-to-end network tenants or users, by enforcing policy of off-path access in the Access-Switching layer. Therefore, the security can be guaranteed for the entire network.

• *Linearly-increasing performance.* Service elements can be distributed anywhere in production networks, thus the heavy network traffic can be assigned to different service elements by flow-level or user-level load balancing. Therefore, the performance can be linearly raised by increasing the number of service elements. Meanwhile, the single point failure and the waste of processing resources can be also avoided.

• *Centralized management.* LiveSec employ a global controller to obtain the entire network information, e.g. network logical topology and Network Information Base (NIB) [4]. The controller can manage the global policies for every network tenant or user, and is in charge of unified policy enforcing, service element traversal, service-aware routing, etc. Therefore, the management is simplified but effective in LiveSec. Any security problems in the network can be quickly located when occurring.

All these advantages result from the novel design that all service elements can interact with the LiveSec controller, and LiveSec controller can do load balancing among multiple service elements according to their real-time load, and furthermore, take action against the end-to-end flow according to the detecting report of service element.

Finally, our key contributions are threefold:

• *Agile architecture.* LiveSec is functionally layered, including Network-Periphery layer, Access-Switching layer and Legacy-Switching layer. The Network-Periphery layer includes wireless and wired users, VM-based service elements. The Access-Switching layer leverages OpenFlow technologies to provide policy-based access management, traffic control and load balance for periphery layer, and the Legacy-Switching layer utilizes existing network architecture to build a high-performance interconnection for Access-Switching layer.

• *Elastic service.* Based on LiveSec, we provide diverse security services by introducing security service elements, such as intrusion detection, protocol identification, virus scanning, content inspection and so on. We design different methods to effectively manage the security elements. For example, a universal VM-based access method for off-path service element, an optimal routing method to make traversal of service element feasible, and a load-balance method to benefit the scalability of the performance of service elements.

• *Feasible implementation.* LiveSec has been deployed in the FIT building at Tsinghua University for nearly a year. We implement 10 Open vSwitches, 200 VM-based service elements, supplying services of intrusion detection and protocol identification, and 50 wireless or wired users. With full service support, LiveSec can achieve at least 8 Gbps for intrusion detection and 2 Gbps for protocol identification.

The remainder of this paper is organized as follows. Section II discusses the related work and background. The detailed design of LiveSec is illustrated in section III. Section IV describes the prototype implementation, and the results of deployment and evaluation are provided in section V. In the last section, we gives the conclusion.

II. Background

In this section, we first explain the state-of-the-art design for security management architecture of production networks. We then discuss the OpenFlow-based research and explain how we utilize OpenFlow to facilitate our work.

The latest and closest work is pswitch [3], where a policy-aware switching layer (PLayer) is proposed for production networks. Being plugged into pswitches in PLayer, security middleboxes can be placed off the data path of the end-to-end network tenants. Pswitches can explicitly forward the flows of different network tenants through desired sequences of security middleboxes, based on administrators' specified policies. However, *i*) PLayer critically relies on that security middleboxes have to be correctly wired with the accurate functional interfaces in pswitches, and *ii*) pswitches in PLayer should be deployed with security middleboxes respectively for each end-to-end network tenants. Both of these limit the scalability and flexibility of the management architecture.

In fact, according to the observation from 4D [5], the root cause of those management problems in today's data networks is that the control-plane decision logic and data-plane distributed protocols are inexorably intertwined in network elements. In clean-slate 4D architecture, network elements simply forward packets at the behest of the control plane, and collect measurement data to aid the control plane

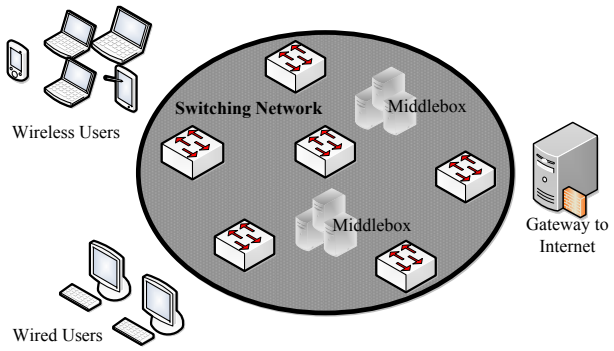

Figure 1. Traditional architecture

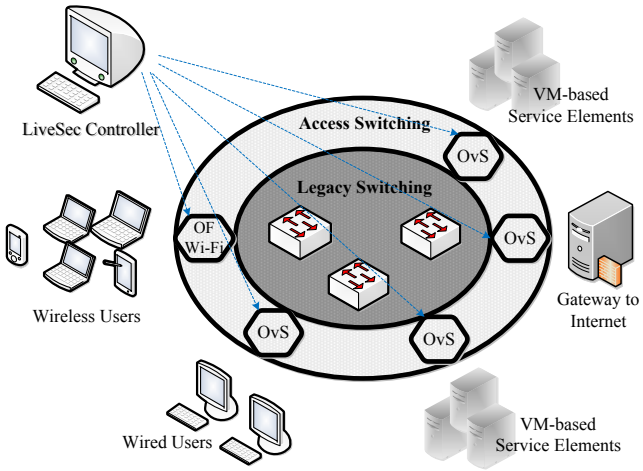

Figure 2. LiveSec architecture

in managing the network. The complete separation of control plane and data plane can bring higher robustness and better security for data networks.

Following the 4D project, the OpenFlow [6] technology is proposed for improving the management in enterprise networks. OpenFlow is dedicated to provide an open and programmable testbed for experimental innovation in campus networks. It couples distributed and dumb flow-based switches with a centralized and intelligent controller taking in control of flows, so that network-wide fine-grain policy defining and enforcing is supported. The emergence of OpenFlow motivates the researches on exploring better OpenFlow-based solution of problems in production networks [4][7][8].

Recent advances of the OpenFlow technologies [6][9][10][11] inspire our research on security management architecture. Our work fully utilizes the OpenFlow technologies from the following three features:

• *Fine-grain control.* By separating the control plane from the data plane, OpenFlow enables both simpler protocols and more sophisticated algorithms in NOX controller [9] for driving the operation of OpenFlow switches [6]. The NOX controller can run desired management applications to achieve flexible control for network monitoring, and govern the information of network elements such as OpenFlow switches and network tenants or users by the OpenFlow protocol to apply fine-grain control for traffic engineering.

• *Open technologies.* OpenFlow is open for researchers to do network application developing or network architecture designing, this is a great help to efficient implementation and development of our work. Open vSwitch [10], an open-source project under long-term support and develop, is purpose-built for achieving OpenFlow in virtualized environment. In order to support the access of VM-based security service elements as well, our work efficiently adopt the Open vSwitch.

• *Unified standard.* OpenFlow has approved technology standard specification, making the development and production of OpenFlow-enabled switches available. Meanwhile, most network equipment vendors would like to add the OpenFlow feature in their switch products without having to open up interfaces inside their boxes and thereby expose the internal workings. In summary, OpenFlow meets the benefit of both network vendors and tenants or users.

III. Networking Architecture

Before describing the architecture of LiveSec, we briefly discuss how our solution discriminates from the traditional designs.

A. Overview

From the logical topology perspective, the architecture of present production networks only includes two layers: the layer of intermediate switching network and the layer of terminal network tenants or users, as shown in Figure 1. This architecture is difficult for management because the path-computation of any end-to-end flow is governed by individual switches, which are distributed throughout the switching network. This distribution bundles the control-plane decision logic and data-plane packet forwarding together.

In order to solve this problem, we introduce a new layer, the Access-Switching (AS) layer, to accomplish the separation of control and data plane. The Access-Switching layer is logically between the Legacy-Switching (LS) layer and the Network-Periphery (NP) layer of traditional network architecture, as Figure 2 shows. Hence the 3-layer architecture of LiveSec is formed. The motivation behind this 3-layer logical networking architecture is that we can effectively enable end-to-end network management on accessing,

routing and policy enforcing via the new added Access-Switching layer. We will specify the design of each layer in the following part.

B. Legacy Switching

Legacy-Switching layer is established by existing traditional Ethernet switches (LS switches) and links, which provides physical interconnection between all the switches in the Access-Switching layer. For small production networks, the legacy switching network can be designed as a star architecture to interconnect the switches of the Access-Switching layer. However, for those networks of large scale, e.g., with tens of thousands of hosts, the architecture of legacy switching network will affect the interconnection performance, e.g., bandwidth or latency. In order to provide a high-performance interconnection for the Access-Switching layer, we expect that the legacy switching network should meet the following objectives:

• *Underlying layer-2 switching.* Legacy switching should render the network configuration of each host identical to what it would be, and support the migration of VMs without changing their IP address. Thus all VMs are connected with layer-2 switching. On the basis of layer-2 switching, any IP address can be connected to any port of an Ethernet switch due to flat addressing, so as to avoid affecting the operation of the other two layers.

• *Uniform high-bandwidth networking.* The upper limit of the end-to-end data delivery should depend only on the terminal hosts, rather than the networking bandwidth of legacy switching network. In addition, any end-to-end available capacity should be uniform for Access-Switching layer, no matter what the network topology is and how heavy the network traffic is.

Because the Legacy-Switching layer is the basis of the Access-Switching layer, we should satisfy the above requirements first to make the architecture more efficient. Here we can refer to previous works on the architecture of production networks. PortLand [12] or VL2 [13], which supports elastic scale from 1 host to 100,000, can be employed in our Legacy-Switching layer.

In the network of Legacy-Switching layer, IP routing and forwarding technologies, e.g. link state routing protocol, Equal Cost Multiple Path (ECMP) routing, IP anycasting and multicasting, are still applicable for underlying data delivery, as well as the DCN-specialized addressing protocols. All of these can be integrated into our solution to some extent.

The Legacy-Switching layer is deployed to supply traditional switching function for the Access-Switching layer, and our design yields the following three features due to its infrastructure.

• *Simplicity.* Due to the backwards-compatibility, our solution can be implemented without changing the traditional infrastructure in existing production networks. Hence we can incrementally deploy the Access-Switching layer through the interfaces provided by the Legacy-Switching layer.

• *Deployability.* As a fundamental switching network, the Legacy-Switching layer is completely transparent to the Access-Switching layer. Therefore, switches of the Access-Switching layer can be deployed anywhere, without caring how the LS switches are distributed and connected. Meanwhile the logical reachability between all switches in the Access-Switching layer can be guaranteed by physical interconnections in the Legacy-Switching layer.

• *Reliability.* Legacy-Switching network significantly relies on link and switch redundancy to provide fault tolerance for end-to-end interconnections. This infrastructure radically benefits our solution to maintain the reliability and support the scalability of the Access-Switching layer.

C. Access Switching

The Access-Switching layer is composed of LiveSec controller [14] (developed based on NOX), AS switches (OpenFlow-enabled switches, e.g., OvS [15]) and AS routers (OpenFlow-enabled wireless routers, e.g., OF Wi-Fi [16]). LiveSec controller and AS switches are deployed anywhere alongside the legacy switches, by connecting to the interfaces that the legacy switches expose for AS switches. At the same time, secure channels are established by AS switches to connect to the control-plane (LiveSec controller).

AS switches are responsible for providing legitimate interfaces for Network-Periphery layer, no more AS switches are allowed to be concatenated between the Legacy-Switching layer and the Network-Periphery layer but one, in order to provide the features as described below. And LiveSec controller accomplishes the centralized network management for network tenants or users.

1) Network topology: For LiveSec controller, the join and leave of each AS switch can be observed through the corresponding secure channel. Based on link layer discovery protocol (LLDP), LiveSec controller can dynamically discover the logical link between all switches. Hence the global network topology of access switching network can be mastered by the LiveSec controller in real time.

According to the description in Section 3.2, the legacy switching network provides logical reachability between any pair of AS switches. As a result, from the view of LiveSec controller, a full-mesh logical topology is formed among all AS switches in the Access-Switching layer. With this simple topology, any end-to-end data delivery merely requires abstract two-hop routing: the ingress AS switch and the egress AS switch. This feature simplifies the logical routing implementation in data plane.

Furthermore, no matter whether loops exist in the legacy switching network, our solution ensures a loop-free access switching network. We owe this feature to the spanning tree protocol (STP) or ECMP in the legacy switching network, because the actual routing between the ingress and egress AS

switches can avoid the loops in the legacy switching network by distributed loop-freeing protocols, thus any abstract two-hop routing based on AS switches will not be affected by redundant physical links in the production networks.

2) Location discovery: The host in the Network-Periphery layer can join in the network by connecting to the Network-Periphery interface that is provided by any AS switch. The Legacy-Switching interface of the AS switch is attached to the LS switch, and the ingress flows from the host will first pass through the AS switch. When the host runs for supplying or requesting application services, its ARP flow will be sent out from the Network-Periphery layer to the Access-Switching layer, and received by the connected AS switch. According to the protocol in [15], the first packet of this ARP flow will be encapsulated and forwarded to the LiveSec controller.

With the ARP packet, LiveSec controller can discover the corresponding host and learn its current location information, e.g. connected AS switch and corresponding port, MAC address, etc. Then LiveSec controller will record this location information of the fresh host in the routing table, or update the routing table for the existing host. If the host leave the network or is down, corresponding location information will be removed from the routing table due to ARP packet timeout.

Besides, with bidirectional ARP packets, LiveSec controller can also get the link information which indicates the port mapping between ingress and egress AS switch, and keep track of the mapping relationship in the link table. Hereto, LiveSec controller can manage all the information of both Network-Periphery layer and Access-Switching layer.

For the handling of broadcast packet such as ARP and DHCP, directly broadcasting will burden the legacy switching network and affect the access switching performance, because all physical links will be involved and all other AS switches will have to handle the broadcast packet again. To solve this problem, a dedicated directory proxy [12] should be employed to specially handle all ARP and DHCP resolutions by looking-up global host information maintained by LiveSec controller.

3) End-to-end routing: To deliver addressing, forwarding and routing for production networks environment, we should leverage the flow observation. Assuming one flow from the host in the Network-Periphery layer enters the ingress AS switch, and its first packet is reported to the LiveSec controller, the LiveSec controller will analyze this packet to find out the flow information, including VLAN id, two-end MAC addresses and Ethernet type (of the data link layer), two-end IP addresses and protocol (of the network layer), two-end TCP ports (of the transport layer) and so on. In this paper we call this flow information as the 9-tuple.

According to the MAC addresses, we can locate many information from the routing table, including the ingress AS switch *src-sw*, the Network-Periphery port *src-sw-inport* (where source destination host connect), the ingress AS switch *dst-sw*, and the Network-Periphery port *dst-sw-outport* (port where source destination host connect). Meanwhile, with the *src-sw* and *dst-sw*, LiveSec controller can find the Legacy-Switching port connection *src-sw-outport* and *dst-sw-inport* between these two AS switches from the link table.

Utilizing the above information, fine-grain forwarding and routing policy can be enforced by adding new flow entries in the flow tables of the ingress and egress AS switches respectively. For the ingress AS switch *src-sw*, the rule field of the flow entry should be matching the *src-sw-inport* and 9-tuple flow information, and the action field should be forwarding from the *src-sw-outport*. Similarly, the flow entry enabled in the egress AS switch *dst-sw* should contain the rule with matching *dst-sw-inport* plus other 9-tuple information, and the action with forwarding from the *dst-sw-outport*.

Once the flow table is established in every AS switch, the LiveSec controller will make the first and follow-up packets of the flow pass through the AS switch by looking-up the local flow table. End-to-end routing can works as follows: when any packet of the same flow enters the AS switch *src-sw* through the port *src-sw-inport*, its header matches the rule field of corresponding flow entry, and it will be forwarded to port *src-sw-outport* according to the action. Note that *src-sw-outport* and *dst-sw-inport* are logically interconnected, while the physical links between them can be optimal by the path computing based on distributed protocols, and the packet is normally routed to the port *dst-sw-inport* of AS switch *dst-sw* by the LS switches. Likewise, the packet matches corresponding flow entry and reaches the destination host after being sent from port *dst-sw-outport*.

The addressing, forwarding and routing for the flow in the other direction are the same. In fact, bidirectional flows can be simultaneously handled as a session. For the request flow, the 9-tuple flow information can be utilized at the same time, to construct the 9-tuple flow information of the corresponding reply flow based on the predefined session policy managed by LiveSec controller.

For the network environment that Pseudo MAC (PMAC) address and Actual MAC (AMAC) address are both applied [12], our solution only cares the packet type between the Network-Periphery layer and Legacy-Switching layer, thus the PMAC address is actually involved in the Access-Switching layer. Moreover, for the network instance that location-specific IP address (LA) and application-specific IP address (AA) are utilized for address resolution [13], our solution will not be affected at all, because our routing is based on layer-2 switching. After all, the procedure in legacy switching network is transparent for Access-Switching layer.

D. Network periphery

Based on the Legacy-Switching layer and the Access-Switching layer, the basic network architecture is established. The Network-Periphery layer is comprised of the tenants or users who utilize this architecture to accomplish various applications, and the Service Elements (SE) that provide network services (such as protocol identification, firewall, intrusion detection, virus scanning, content inspection, and so on).

Network tenants or users will join in the network via the Access-Switching layer. More detailedly, wireless users access the network over OF Wi-Fi, while wired users and service elements connect to the network via OvSes. The service elements are designed to be VM-based, so that they are well suited to plug-and-serve deployment of various network services.

1) Service element: To make the network service more scalable and flexible, the following two issues are necessary to be further taken into account: *i*) dynamic migration for elastic utilization of network service resources, and *ii*) centralized management for uniform and convenient control. Aiming for addressing these issues, we mainly contribute in the support of VM-based service element, and the design of communication mechanism between service elements and the LiveSec controller.

A service element can be viewed as an off-path middlebox [3] in the Network-Periphery layer, rather than the switching layer for traditional network architecture. Therefore, service elements can access the network via the Access-Switching layer. To meet the requirement of dynamic migration, we should deploy the service element based on VM, because the mobility of users and VMs can be guaranteed by existing OpenFlow technologies. Supported by OvS, VM access is feasible, hence we can introduce VM-based service element to provide elastic network service in LiveSec.

However, the LiveSec controller can be aware of the service element as a host, but cannot find out whether it is a service element, or what the network service is. Therefore, in order to manage the service elements effectively, they must be able to communicate with the LiveSec controller. According to the OpenFlow protocol, the first packet of every end-to-end flow will be forwarded to LiveSec controller, and only the packet header is used to discover the host information. Based on this observation, we design a communication mechanism with exploiting the packet content for the communication between service element and the LiveSec controller.

In the service element, we use a service daemon to encapsulate the desired message in a UDP packet with specialized format and identifier. When this UDP packet is sent out from the service elements and received by the AS switch, the flow table will be looked up to see whether there is a corresponding flow entry. If not, packet will be forwarded to the LiveSec controller. LiveSec controller will check the packet content via message parsing module to see if the message identifier is legitimate, for the right communication message, packet will be further analyzed to find out the detailed message content according to the message format. At the same time, LiveSec controller will not add the flow entry for this UDP flow, so that every message coming from the service element can be forwarded to LiveSec controller all the time.

In order to make the communication more secure, we can further employ a certification mechanism. All joined service elements must have the proven certificates issued by the LiveSec controller. Otherwise, the flows generated by the corresponding service element will be dropped in the ingress AS switch.

By leveraging the communication mechanism, a service element can deliver the information about its network service to the LiveSec controller. The first one is the real-time on-line message, which is used to confirm the existence of the service element and which type of network service it provides. Meanwhile the load information such as CPU utility, memory footprint and number of processed packets per second is attached. The other is the event report message, which is generated and sent only when the result of network service is produced. In addition to the service result, corresponding end-to-end flow information will be also included in the message. LiveSec controller will use this message to take relevant actions to the end-to-end flow for centralized management. In other words, the action is not taken by distributed service elements.

2) Features: Based on the design of VM-based service element and corresponding communication mechanism, our network architecture can support not only the distributed network services via service elements, but also the uniform management for all service elements and the network traffic. Therefore, the mobility and load balance of the service elements, the monitoring and control of entire network traffic are also enabled by our design.

In fact, from the perspective of PC, the legacy switching and access switching network in LiveSec construct the motherboard, while the VM-based service elements are the hardware resources, connected to the motherboard via network bus. Meanwhile, as a network operation system, LiveSec controller is in charge of managing these hardware resources via the communication mechanism, which is equal to the driver of these hardware resources. Therefore, in the sight of LiveSec controller, adding more service elements can be uniformly regarded as the increase of the capacity of hardware resources. Hence we can efficiently allocate desired resources for each user.

Compared with traditional network architecture, network event information can be recorded by system log of distributed network elements. However, distributed management cannot effectively utilize this information to locate the network problem. In LiveSec, we can master the network events by

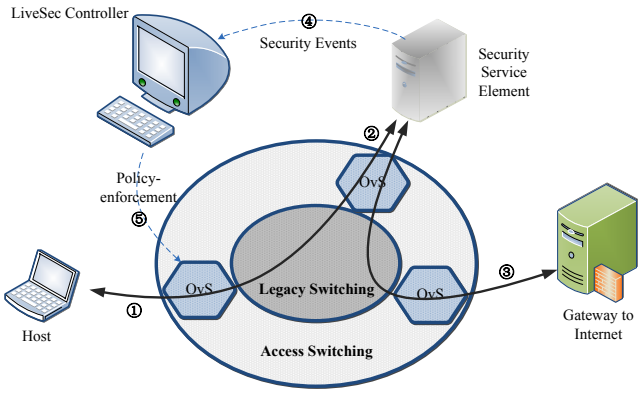

Figure 3. Interactive policy-enforcement

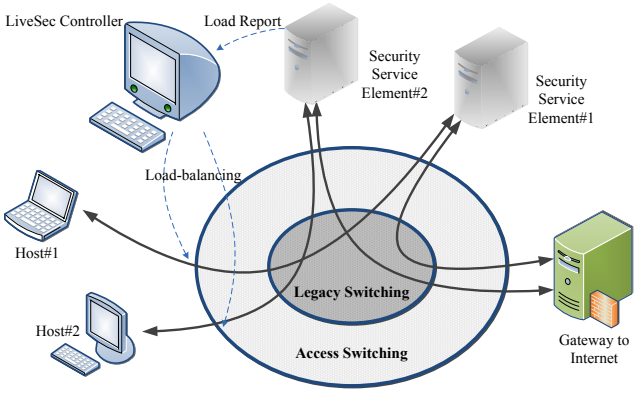

Figure 4. Distributed load-balancing

only first few packets. Because the log information is global, it is convenient to manage the network by visualizing the network environment, and locate the network problems by replaying the history events.

IV. LiveSec Implementation

In this section, we implement the security management architecture by applying the security service elements, which can provide service of intrusion detection. A series of significant features are also demonstrated through several instances.

A. Interactive policy enforcement

When a security service element is added into the network, it will send real-time on-line message to the LiveSec controller. The LiveSec controller will find the security service element according to the message, and record its location information, including the MAC address and the network service type, here is intrusion detection.

The LiveSec controller keeps a global policy table that is pre-configured and managed by the network administrator. The policy table describes whether or which security service element should be traversed for various end-to-end flows. In traditional architecture, complicated logic mechanisms such as removing physical connectivity, manipulating link costs or separating VLANs, have to be employed to make the target flow to pass through designated security middleboxes. Yet in our architecture, the traversal of any security service element can be flexibly guaranteed, under all conditions of policy by interactive policy-enforcing.

Figure 3 illustrates the feature of interactive policy-enforcing. When a user tries to access the Internet through the gateway, the LiveSec controller will loop up the policy table with the first packet of the flow. If the policy indicates that the flow must pass through the intrusion detection element, the LiveSec controller will redirect all the packets of the connection to the appointed service element for security check. After that, the checked packets are forward to the Internet. The service element will report security event of the connection to the LiveSec controller once malicious attack is detected in the flow. Based on the event, the LiveSec controller decides that a drop action is enforced to the ingress AS switch where the user is connected.

To facilitate this procedure, corresponding flow entries will be loaded in all involving AS switches to make the flow go along the desired logical path, so that the flow is completely scrubbed before leaving the gateway.

So when a flow enters the ingress AS switch, LiveSec controller will first *i*) add a flow entry in the ingress AS switch, with the action that changes the destination MAC address of the flow into the MAC address of the appointed service element and forwards the packet to the service element, then *ii*) add a flow entry in the AS switch connected with the security service element, with matching the modified flow from ingress AS switch and forwarding to the security service element, and *iii*) add another flow entry in the same switch with matching the new flow the security service element sends back, forwarding to the target of original flow, at last *iv*) add a flow entry in the egress AS switch to match the same new flow and send out to the gateway.

Because LiveSec controller has the global information, according to the first packet the original flow, all above flow entries can be calculated and enforced simultaneously. After all flow entries are established, the first and subsequent packets of the original flow will be sent out and forwarded in line with the flow table of each AS switch. The destination-MAC-address-modified flow will be normally leaded to the target security service element by layer-2 switching in legacy switching network.

After several packets of the flow are processed, if the security check is successful, the security service element will report a security event message including the 12-tuple information of the detected flow and the corresponding attack type. LiveSec controller will then modify relevant flow entries with the drop action in the ingress AS switch,

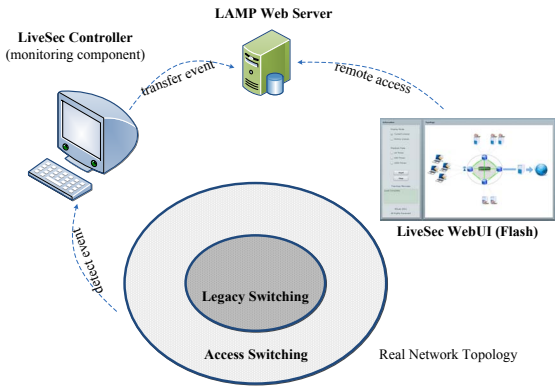

Figure 5. Visualization

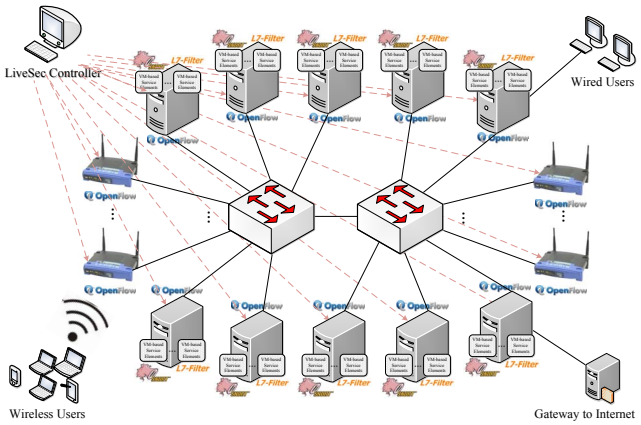

Figure 6. LiveSec deployment

to block this flow at the entrance. Hence the inner switching network will be completely protected from the outer terminal attacks. Therefore, our architecture enables uniform and flexible security management based on interactive policy-enforcing.

B. Load balancing

The throughput of single security service element is always limited. In order to make the security performance of the network scalable, we can add more security service elements and distribute them anywhere in the Network-Periphery layer. However, a bad balancing will result in a bad performance, e.g., partial security service elements are overloaded while others are idle. Thus we should balance the load among all elements globally. By achieving global balancing on the LiveSec controller, the whole network security performance can be linearly increased with the number of security service elements.

As Figure 4 shows, the load balancing utilize the AS network, rather than a dedicated load-balance appliance to distribute security work-load. Aware of the load information of every security service element, LiveSec controller can intelligently select the security service elements for each end-to-end connection according to pre-defined policies and dynamic network states.

According to pre-defined policies, LiveSec controller can do load-balancing with different granularity. For example, with few users but heavy network traffic, flow-grain load balance is preferred, or flows are equally assigned to different security service elements. However, when there are a large number of users, user-grain load balance is more effective in terms of both speed and efficiency.

For dynamic network states, LiveSec controller can utilize different dispatching algorithms such as polling, hash, queuing or minimum-load method. According to different requirements, different algorithms may be utilized to choose the proper security service element for each flow or user, so as to make the real-time load of each security service element as balanced as possible.

Overall, compared with traditional architecture, our LiveSec enables high-efficiency and scalable security management based on the support of distributed access of security service elements and the load balance among them. Moreover, all the security service elements are available for any end-to-end connection, thus the single point of failure can be avoided in LiveSec.

C. Service-aware traffic monitoring

Our architecture can also support service-aware traffic monitoring besides the above-mentioned features. Based on the application identification service elements, the application protocol of each flow can be analyzed, therefore LiveSec controller know the services status that each user is consuming. With this information, LiveSec controller can further master the network traffic distribution and service-aware statistics, and provide more interesting function, such as aggregate flow control.

D. Visualization

Leveraging the global topology information and network traffic information, our architecture provides dynamic visualization of many real-time network events, such as user join and leave, load condition of links and various service elements, which user is accessing which application service, where attacks happen, and so on. As Figure 5 illustrates, a LiveSec WebUI [14] is designed based on the backstage database to display the above topology and events, as well as history replay.

The procedure of the WebUI is described as below. At the backstage, when the event processing module receives a network event, it will report to the monitoring component. The monitoring component then gathers the information and records it to the database of a remote web server. The front-end website, which displays the topology and events, uses

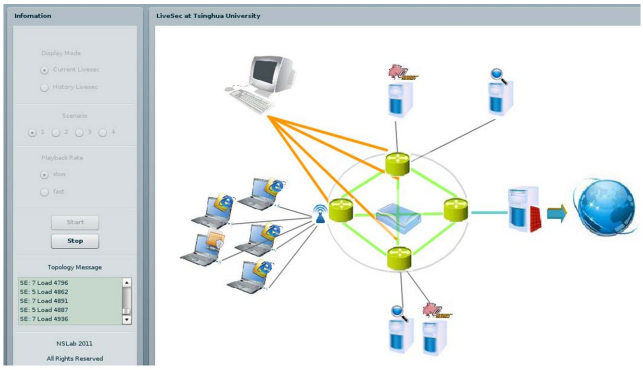

Figure 7. Normal network environment

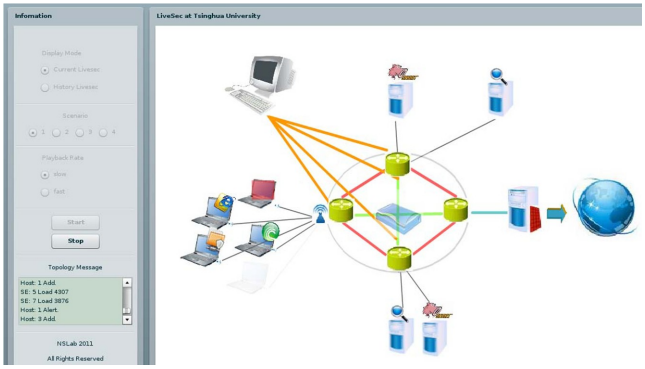

Figure 8. Network events happen: user leave, traffic increase, attack found

a timer's command to request the information from the remote web server periodically. The web server has been configured in the LAMP (Linux, Apache, MySQL and PHP) mode, and can fetch requested data from the database for the corresponding display in the website.

Compared to the existing OpenFlow visualization program ENVI, our design is simpler and more efficient for the implementation, due to *i*) the web-based display which only needs the browser enable Flash function and has no dependence on the operation system or any other library, and *ii*) the usage of database which makes mass data storage and operating possible, meanwhile allows multiple users to access the WebUI simultaneously.

V. Deployment

Since our implementation is based on commodity hardware and open-source software, new LiveSec networks can be quickly deployed for both research and daily use. Currently, LiveSec is massively deployed in the FIT building at Tsinghua University (testbed is shown in Figure 6). For the basic network environment of FIT building, networking center provides the core switching network for the whole building, and each storey is equipped with distributed secondary Ethernet switches to support the access of respective labs. Such a network practically supports three network segments, the access of over 3000 users and bandwidth of 100Mbps downstream for each user.

A. Network scale

We implement two switching and wiring closets with OpenFlow-enabled switches for the access of service elements and wired users. The OpenFlow-enabled switches are implemented based on Intel Xeon 5500 series servers with four Gigabit Network Interface Cards (NICs) and software package of OvS release version 1.1.0. We also deploy twenty OF Wi-Fi APs in various meeting rooms of the building for other wireless users. The OF Wi-Fi APs are applied with commercial Pantou based on the OpenWrt release version 10.03.

All 10 OpenFlow-enabled switches are both connected to the Gigabit backbone network of the building by two 24-port Gigabit Ethernet switches, and the bandwidth provided for every user will be no less than 100Mbps. From the perspective of LiveSec controller, these OvSes in the two machine rooms and other OF Wi-Fi APs distributed in different meeting rooms are all connected by an abstract traditional switching network and form the full-mesh Access-Switching layer around it.

B. Performance evaluation

1) Throughput: In the situation of UDP flows, single OvS can get up to 100Mbps access performance for wired users, and single Pantou can reach 43Mbps for wireless users [17].

Each OvS can support running 20 VMs as the service elements simultaneously with each service element connected to the OvS itself. We implement the functions of intrusion detection and application identification in service elements by porting open-source software tools, Snort [18] (an opensource intrusion detection system) and Linux L7-filter [19] (an opensource protocol identification system). Under the bypass mode, single VM-based service element can reach about 500 Mbps throughput, and the maximum performance of 20 VMs is limited to the Gigabit NIC of the physical host implemented with OvS. According to the test with HTTP flows, performance of single VM-based service element is 421 Mbps, and twice VM-based service elements raise the whole performance to 827 Mbps. Our result verified that the performance can be linearly increased with the number of VM-based service elements.

Normally, we have about 30 wireless users, 20 wired users, and 200 VM-based service elements supplying network services of intrusion detection and protocol identification. The performance of the LiveSec unit can achieve at least 8Gbps for intrusion detection and 2Gbps for protocol identification. In fact, the maximum capacity cannot be practically tested because the real-life traffic is not heavy, the traffic are primarily limited by the performance of the ingress OvS or OF Wi-Fi and their numbers.

2) Load balance: The load balance based on the selecting minimum-load method is effective in the practical test. The load is judged according to the number of received and processed packets. For the normal traffic, the real-time load deviation among multiple service elements is no more than 5%.

3) Latency: We test the network delay by pinging from the user to an Internet server. Compared with legacy switching network without access the Internet through OpenFlow-enable equipment, we can find that, LiveSec only increase the average latency by around 10%. This result will almost cause none affection to the user experience.

4) Visualization: The effect of visualization via WebUI is shown in Figure 7 and Figure 8, 3 OvSes and 1 OF Wi-Fi are deployed in this practical network, and only 2 intrusion detection service elements and 2 application identification service elements are on-line at present, connected to 2 OvSes respectively. All the users connected with the AS switches will access the Internet through LiveSec. From the figure, we can see that the logical topology of the entire network is full-mesh in deed.

In Figure 7, the network is running normally with 5 wireless users. Based on the service elements, we can monitor that 4 of them are browsing webpage, while the rest one is using the SSH service, and the network traffic is small. After a period of time, from Figure 8, we can observe that one user has left the network, meanwhile one user who was browsing webpage at first is downloading by BitTorrent right now, leading to the high utilization of the network links. In addition, another user is trying to access some malicious website, while this action is detected and reported by the service element immediately.

VI. Conclusion

In this paper, we propose LiveSec, an agile architecture for network security management in production networks. LiveSec achieves flexible security management with end-to-end security coverage and linearly scalable performance. Practical deployment in a campus networks shows that LiveSec enables lots of novel functions, including interactive policy-enforcement, effective load-balancing, application-aware monitoring, as well as network environment visualization and event replay. These features make the management in production networks more intelligent, convenient and flexible. Our experience in production networks demonstrated that LiveSec can greatly benefit the vast users by providing more secure and diverse network services.

Acknowledgment

We would like to thank Xinming Chen, Jeffrey Fong, Feng Xie, Yiyang Shao, Yang Gao and Haopei Wang for their work on helping build the platform, and we appreciate all the anonymous reviewers for their insightful suggestions. This work was funded by Tsinghua National Laboratory for Information Science and Technology (TNList) Cross-discipline Foundation.

References

[1] Amazon, *Elastic Compute Cloud*. [Online]. Available: http://aws.amazon.com/ec2/

[2] D. Maltz, G. Xie, J. Zhan, H. Zhang, G. Hjálmtỳsson, and A. Greenberg, "Routing design in operational networks: A look from the inside," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 27–40, 2004.

[3] D. Joseph, A. Tavakoli, and I. Stoica, "A policy-aware switching layer for data centers," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 51–62, 2008.

[4] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix: A distributed control platform for large-scale production networks," in *Proceedings of OSDI*, 2010.

[5] A. Greenberg, G. Hjalmtysson, D. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, "A clean slate 4d approach to network control and management," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 5, pp. 41–54, 2005.

[6] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[7] M. Yu, J. Rexford, M. Freedman, and J. Wang, "Scalable flow-based networking with difane," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 351–362, 2010.

[8] R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Can the production network be the testbed," in *Proceedings of OSDI*, 2010.

[9] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "Nox: towards an operating system for networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 105–110, 2008.

[10] B. Pfaff, J. Pettit, T. Koponen, K. Amidon, M. Casado, and S. Shenker, "Extending networking into the virtualization layer," in *Proceedings of HotNets*, 2009.

[11] A. Tavakoli, M. Casado, T. Koponen, and S. Shenker, "Applying nox to the datacenter," in *Proceedings of HotNets*, 2009.

[12] R. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "Portland: a scalable fault-tolerant layer 2 data center network fabric," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 39–50, 2009.

[13] A. Greenberg, J. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, and S. Sengupta, "Vl2: a scalable and flexible data center network," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 51–62, 2009.

[14] Y. Qi, F. He, K. Wang, X. Chen, J. Fong, F. Xie, Y. Shao, Y. Gao, Y. Xue, and L. J., "Livesec: Openflow-based security management for production networks," in *Proceedings of IEEE INFOCOM (Demo)*, 2009.

[15] *Open vSwitch*. [Online]. Available: http://openvswitch.org/

[16] *OpenWrt*. [Online]. Available: http://openwrt.org/

[17] *Pantou*. [Online]. Available: http://www.openflow.org/wk/index.php/Pantou_:_OpenFlow_1.0_for_OpenWRT

[18] *Snort*. [Online]. Available: http://www.snort.org/

[19] *Application Layer Packet Classifier for Linux*. [Online]. Available: http://l7-filter.sourceforge.net/