A

SEMINAR REPORT ON

"NETWORK SECURITY VIRTUALIZATION : NETSECVISOR."

SUBMITTED

TO

SAVITRIBAI PHULE PUNE UNIVERSITY

UNDER THE PARTIAL FULFILLMENT

OF

BACHELOR OF ENGINEERING

(COMPUTER ENGINEERING)

BY

MRUDULA BORKAR
ROLL NO: 14CO302

UNDER THE GUIDANCE

OF

Prof. J.S.CHINCHOLE

AISSMS College of Engineering
Pune-411001
Department Of Computer Engineering
2015-2016

Abstract

In recent years as the network is rapidly increasing, network security management is becoming more and more complicated. To increase the security of network, more and more network security devices/ middle-boxes needs to be deployed at various locations. This increases complexity of network. Also, current management is inflexible and the security resource utilization is inefficient. The flexible deployment and utilization of proper security devices at needed time with low management cost is difficult to achieve.
To address this problem, new concept of network security virtualization is introduced. It virtualizes security resources/functions to network administrators/users and it maximally utilizes existing security devices. It also enables security protection to desirable networks with minimal management cost. To verify this concept, a prototype system is designed called as NETSECVISOR, which utilizes existing pre-installed security devices and software-defined networking technology is leveraged to virtualized network security functions.
At its core, NETSECVISOR contains:
1) A simple script language to register security services and policies;
2)A set of routing algorithms to determine optimal routing paths for different security policies based on different needs; and
3) A set of security response functions/strategies to handle security incidents.
NETSECVISOR is deployed in both virtual test networks and a commercial switch environment to evaluate its performance and feasibility. The evaluation results show that this prototype only adds a very small overhead while providing desired network security virtualization to network users/administrators.

TABLE OF CONTENTS

Chapter 1
Introduction

1.1 Social Issue:

As need of securing network increases, need of new hardware resources changes. As the security is more important, new resources are incorporated in system which makes the existing components idle which in turn results in e-waste. Hence, it is necessary to make proper use of resources to avoid e-waste and save environment.

1.2 Problem Statement:

To develop a system which can effectively use the available resources for securing a network and other network related to issues in order to avoid idleness of resources.
Today a huge problem is faced over the world that is Electronic-waste, and it is necessary to come up with different solutions which can lessen or avoid E-waste. By ignoring this problem, there is a possibility that many toxic materials like mercury exposed in the environment and which can cause a lot of damage to the society and environment.

1.3 Objective:

- To minimize the need of security devices.
- To maximize the utilization of resources.
- To maintain security flow in the network whenever necessary.

1.4 Motivation:

There is an urgent need to maximize the utilization of existing resources i.e. existing pre-installed devices. As well as abstraction of security resources is necessary to provide simple interface to the people who are actually using it. Hence, motivated by this problem, new concept of NSV i.e. Network Security Virtualization is introduced which leverages pre-installed security devices and provide dynamic, flexible and on-demand security services to the users.

As the demand of networked service is increasing, network architectures are becoming more and more complicated. The disadvantages are with complicated networks are firstly that they are not good for scalability and secondly that they are provided on the call speech path. Also these networks are expensive to formulate. For example, a campus network has to be designed and different departments should be covered in it. Also, different departments may have different network and security protection policies. It may cause complex network management and inefficient usage of network security resources.
When there are additional middle-boxes in the network environment, situation is more complicated. Nowadays, many middle boxes are employed to improve the performance, robustness and security of networks. For example, a load balancing proxy server is installed in network to distribute network flow into target servers. And it is installed on the location from where all

flows should be distributed and are passing. Although, the middle boxes provide many benefits to network, they also make the network more complicated to manage.
Security of networks is essential but it is also true that when additional security devices/middleboxes are added, network configuration or management gets significantly complicated. In addition, security devices have many diverse security functions to serve different purposes. For example, a firewall installed in a system controls network access, a network intrusion detection system to monitor exploit attacks in network payloads and a network anomaly detection system to detect DDoS attacks. Therefore, network administrator has important task to choose reasonable security functions/devices and deploy them into reasonable places. It is hard for administrator to predict possible network threats and possible demands from the customer. Thus, there is a possibility that the functions/devices are not deployed at the best location where it can best serve the diverse security needs of diverse network users.
Hence, maximization of resources is the compelling need nowadays. Also, providing simple user interface to the people who may not be aware of the security device information is necessary. To overcome this problem, a new system is proposed i.e. Network Security Virtualization. To maximize the utilization of resources, network flows should be redirected to that particular resource whenever necessary. Besides the flow controlling, it also provide a way of enabling security response function on each network device[1].There are some technologies which provides a method to control network flow dynamically at a network device i.e.,Software Defined Networking(SDN). With the help of this technology, realization of security response function can be done.Also, by extending this technology, necessary security response functions are operated on a network device when they are required.
Software Defined Networks i.e.SDN offers a promising alternative for *middlebox policy enforcemrnt* by using logically centralized management, decoupling the data and control planes, and providing the ability to programmatically configure forwarding rules[4]. Software Defined Netwroks facilitate rapid and open innovation at the network control layer by providing a programmable network infrastructure for computing flow policies on demand. However, the dynamism of programmable networks also introduces new security challenges that demand innovative solutions[2].
To verify these ideas, a prototype system has been implemented known as NETSECVISOR. The main goal of this NETSECVISOR is to provide dynamic, user-friendly,transparent and on-demand security services to various users in a large cloud network. Software Defined Networks i.e.SDN technology and it's popular realization i.e. OpenFlow is used. More specifically, Rather than physically adjusting the location of security devices, SDN/OpenFlow is used to control the path of traffic to leverage pre-installed, static security devices to provide dynamic service monitoring services. In addition, some basic security response functions are enabled, such as network isolation on network devices.
In this prototype design, following things are included-
(i)a user-friendly policy script language for both administrators and tenants,
(ii)routing algorithms,
(iii)response strategies.

Chapter 2
Literature Survey

There are some related studies to this work. OpenSafe and Jingling are similar studies to this work. They also provide a script language to monitor networks.Compared with them, this work focuses more on security monitoring in clouds, considers characteristics of different security devices, and designs different routing algorithms and response strategies for security needs.
Sekar et al. suggested approaches for NIDS or NIPS deployment through selectively monitoring packets at different nodes and load balancing [6] through traffic replication and aggregation. Even though they is similar, this study finds new routing paths for security devices, and provides in-depth analysis of routing algorithms, and provides more diverse routing and response strategies considering security devices. In a recent study [7], it is shown that it is possible to redirect traffic from Enterprise networks to a cloud service for network processing. However, this study used a relatively complex way to implement that. NETSECVISOR leverages SDN/OpenFlow to simply the routing and provides various flow detouring and response strategies for security purpose.
Most recently, a new concept very close to this study is proposed, i.e., Network Function Virtualization [9], which proposes to put network middle-box functions into virtual machines a centralized server farm thus providing efficient network services. While conceptually similar, proposed NSV differs in several fundamental aspects:
First, it does not need to convert network middle-box functions into virtual machines and relocate into a centralized place, instead it can rely on existing pre-installed security devices and transparently control network flows to desirable devices when necessary.
Second, NSV focuses on providing security virtualization, instead of generic network functions. It has some security specific features, e.g., enabling security response functions and providing a user-friendly security policy script language for both administrators and tenants.
There exist some studies on middle box policy enforcement using SDN. While these studies share some spirit, they are fundamentally different in goals and technical approaches. These studies mainly propose to bind between packets and their "origins" and ensure that packets follow the right sequence of middle boxes. In this, focus is mainly on specific security challenges: How to generate the right routing path that leverage pre-installed, static security devices to provide dynamic security services? How to provide a user-friendly way for tenants to specify security policies without worrying about information of security devices (i.e., location, number/kind of devices), operating security functions, and underlying network architecture/workload. Furthermore, how to respond to security alerts? All these are unique in this. Finally, this approach could potentially benefit from these existing complementary studies [3], [4], particularly if service chaining problem is properly dealt with.

Chapter 3
Proposed System

Here,architecture and operation scenario of NETSECVISOR is discussed.

3.1 Concept Of Network Security Virtualization

What is virtualization? Simply put, its the process of creating a virtual, rather than physical, version of something. Virtualization can apply to computers, operating systems, storage devices, applications, or networks.Virtualization uses software to simulate the existence of hardware and create a virtual computer system. Doing this allows businesses to run more than one virtual system and multiple operating systems and applications – on a single server. This can provide economies of scale and greater efficiency.Network virtualization is a method of combining the available resources in a network by splitting up the available bandwidth into channels, each of which is independent from the others, and each of which can be assigned (or reassigned) to a particular server or device in real time. Each channel is independently secured. Every subscriber has shared access to all the resources on the network from a single computer.
The main concept of network security virtualization(NSV) is to virtualize the security resources or security functions and providing on-demand security to any(possible) network/places in a user-friendly manner.It mainly includes two functions:
(i)Transparently delivering network flows to desirable security devices
(ii)Enabling network security response functions into network device whenever necessary.
The first function allows us to provide security services to any network flow that requires the services. And, the second function helps us to distribute security services to each and every network device.

3.2 Architecture Of NETSECVISOR

It is difficult to realize the network security virtualization with the traditional network technology because it lacks certain features. To address this issue, an emerging networking techniques are leveraged. Software Network Architecture(SDN)[1], which can help us to control network flows dynamically and monitor whole network status easily. This prototype system is implemented based on SDN.

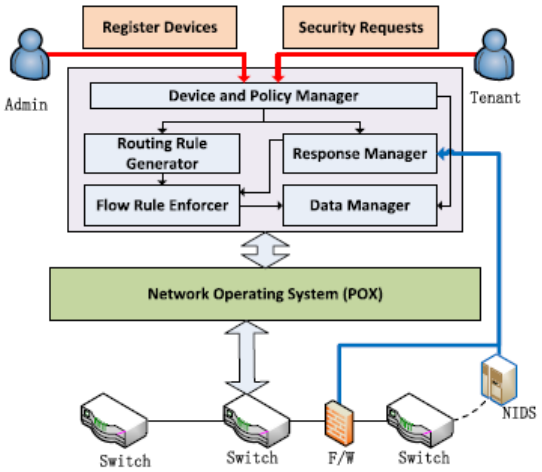

Fig.3.1 : Overall Architecture [1]

As shown in above figure, NETSECVISOR contains five main modules-
(i)Device and Policy Manager
(ii)Routing Rule Generator
(iii)Flow Rule Enforcer
(iv)Response Manager and
(v)Data Manager.
Device and Policy Manager has two important tasks.First task is to receive the information of security devices from cloud administrator and store that information in device table of NETSECVISOR. Second task is to receive security requests from each network tenant and translate it into security policies and store these policies in policy table. As the outcome of this module,two types of information are collected i.e.
(i)location/type of security devices and
(ii)security policy from each tenant.
It makes handling of network security devices easily.
Routing rule generator creates routing paths to control each network flow. While creating routing paths, this module investigates security polices (from each tenant) to satisfy their requirements. For example, if a tenant defines a security policy that specifies all network flows to port 80 should be inspected by a NIDS attached to a router A, then this module produces routing path(s) to let all network packets heading to port 80 pass through the router A. It helps system assign security requirements to each security device based on efficiency (in terms of security resource management) and effectiveness (in terms of finding reasonable security devices).
Response manager receives detection results from security devices, and it enables security response strategies that are defined in security policies, when it is needed. For example, if a tenant defines a security policy to drop all corresponding packets when a threat is detected by a NIDS, the response manager will enable drop function to discard network packets belonging to the detected network flows on a network device.Enabled functions will be realized as a set of network flow rules, which are sent to routers or switches, and thus each network device as a kind of security device(e.g., firewall) is leveraged.
Flow rule enforcer enforces flow rules to each OpenFlow router and switch. If the response manager enables response strategies or the routing rule generator produces routing paths, this module translates them into flow rules that could be understood by OpenFlow routers/switches.

After translation, it sends translated rules to corresponding routers or switches.
Data manager captures network packets from routers or switches to hold until some security devices send their detection results to NETSECVISOR. The reason why it holds packets is to enable some in-line style security funtions as what generic Intrusion Prevention Systems provide. This module does not hold packets all the time, but only captures and stores when necessary (i.e., a security policy specifies an in-line mode action for response)[1].

A working of NETSECVISOR can be stated as follows-
A network administrator registers all the security devices i.e. physical and virtual appliances to NETSECVISOR. After registration, cloud tenants need to create their security requests and submit it to NETSECVISOR. Then, NETSECVISOR parses the requests sent by tenants to understand the goal or intention of tenants and accordingly writes the security policies to the policy table. If NETSECVISOR receives a new flow setup request from a network device, then it checks whether the new request is matched with any submitted policies. If it matches, NETSECVISOR creates new routing path and corresponding flow rules for this new path. At this time, NETSECVISOR also guaranteed that the required security devices are also included in the path which are defined in security policy. After this, it also enforces flow rules to each network device to forward the network flow. If any security device detects malicious content from the traffic, then this information is immediately reported to NETSECVISOR. Nased on the submitted policies, security response function is enabled to respond to this malicious flow accordingly.

3.3 Registration Of Security Devices

To use the existing resources that are the security devices, a cloud administrator has to register these devices to NETSECVISOR using simple script language. There are various scripting languages available. In registration, following information has to be filled out i.e.
(a)Device ID
(b)Device Type
(c)Device Location
(d)Device Mode
(e)Supported Functions.
This can be illustrated using an example-
Let's assume that Intrusion Detection System(IDS) is installed into a cloud network, and it is connected to the switch whose data path ID is 121. Moreover, assume that this IDS is operated in passive mode and it protects network from the DNS attacks. This IDS can be registered to NETSECVISOR using following script:
[1,IDS,121,passive,detect DNS attack]

3.4 Creating Security Policies

After the registration of security devices by cloud administrators to NETSECVISOR, the respective information of the security device is shown to the cloud tenants by NETSECVISOR. Then, by considering registered security devices and security functions enabled by NETSECVISOR, tenants can define their security requests. Security requests of tenants are described with script language which is provided by NETSECVISOR.
The script for a request contains three main field which includes:
(i) flow condition, which specifies the flow to be monitored (or controlled),
(ii) function set,which denotes the necessary security devices for monitoring or investigating, and

(iii) response strategy, which defines how to handle the flow if a threat is detected.
The policy syntax is:
flow condition, function-list, action-list.
Currently, NETSECVISOR supports 5 different response strategies (two in passive mode and three in in-line mode), and they are drop, isolate for passive mode and drop, isolate, redirect for in-line mode. This can be explained using a example script:
Security Request- one tenant (IP = 10.0.0.1) wants all HTTP traffic regarding to his IP to be monitored by a firewall and an IDS, and it wants to drop all packets issued as attacks by the firewall and the IDS. This request can be sent to NETSECVISOR with the following script:

```
{{((DstIP = 10.0.0.1 OR SrcIP = 10.0.0.1) AND (DstPort = 80 OR SrcPort = 80))},
{firewall, IDS},{drop}}
```

Finally, NETSECVISOR receives security requests from each tenant, and it translates them into security policies that can be applicable to a SDN enabled cloud network. At this time, NETSECVISOR needs to translate tenant defined high level conditions into more specific network level conditions, and it also maps function set into security devices registered before.

3.5 Routing Path

If NETSECVISOR finds network packets meeting a flow condition specified by a security policy, then it will route these packets to satisfy security requirements.When NETSECVISOR routes network packets, it should consider the following two conditions:
(i) network packets should pass through specific security devices to meet the security requirements, and
(ii) the created routing paths for network packets should be optimized.
There are several existing routing algorithms for intradomain (e.g., OSPF) to find optimal paths. However, they can not be employed directly for yjis case. Since network packets only contain the source and destination information, existing routing approaches can not discover necessary ways to locations where security devices are installed. Thus, own approaches are created in which NETSECVISOR supports two modes of security devices which are passive mode and in-line mode. For a passive mode device, the traffic is forwarded to pass through the device, or just mirror a duplicate to the device and forward the original traffic in another way. For an in-line mode device, all traffic should pass through and be monitored by this device. The generated routing path should satisfy the requirements from different modes of security devices. Besides, a network may contain only passive mode devices or in-line mode devices, or both the two kinds. Thus, aim is to design different algorithms for different usage scenarios. Recent software-defined networking technologies (e.g., OpenFlow) provide several interesting functions, and one of them is to control network flows as per need. With the help of this function, 4 different routing algorithms are proposed, which can satisfy different requirements.To explain algorithms more clearly, four terms are defined:
(i) start node, a node sends network packets,
(ii) end node, a node receives the packets,
(iii) security node, a node mirror packets to a passive security devices, and
(iv) security link, a link on which in-line security devices are located. Among the proposed 4 algorithms, 3 of them (i.e., Algorithm 1 - 3) are designed for security policies which only use passive mode devices, and 1 of them (i.e., Algorithm 4) is suggested for policies which have in-line security devices such as a firewall and a NIPS. To describe algorithms more clearly,The path between two nodes should be found out. A network can be characterized using a graph

structure, which consists of nodes (hosts or routers or switches) and arcs (physical links between devices). In this graph structure, some paths has to be discovered between a start node, which sends network packets, and an end node, which receives network packets. At this time, usually shortest path between a start node and an end node is procured in order to deliver network packets efficiently. The problem of finding the shortest path between two nodes is a type of a linear programming, and it can be formulated as the minimum cost flow problem.

Enabling Security Response Function

After the inspection of network flows with specified security devices, a tenant can decide response method or response function for the network packets which are detected as malicious or suspicious by security devices. Some action has to be taken on these detected packets. A user may want to discard all detected packets or isolate some infected hosts. Implementing these methods is very hard using existing network devices, because there is a requirement of additional inline devices which can handle network packets or changing the configurations of network device. For example, consider that some intrusion detection systems are installed into a cloud network. In this case, tenant may want to drop all the infected packets which are detected by Intrusion Detection System.But as these systems can not drop the packets, some other methods are stated to support the request of tenant.

To address this issue and to provide more flexible response methods, NETSECVISOR provides 5 security response strategies which does not require including physical security devices or changing the network configurations to handle the packets. These methods can be operated in two different methods:(i)passive mode and (ii)in-line mode.

Passive mode response strategies are similar to strategies by existing network intrusion detection systems that mirror network traffic for investigation and generate alerts. In this mode, some malicious network traffic could have been already delivered to a target host. In this passive mode, NETSECVISOR supports two response strategies. First, NETSECVISOR can drop packets that belong to detected network flows. This strategy is useful to stop some later malicious packets in the flow, but it does not guarantee that none of malicious packets are delivered to the target host. Second, NETSECVISOR can isolate a specific host or a VM, if it is detected as malicious. In this strategy, NETSECVISOR is able to block sending network packets to a detected host or a VM, or from a detected host or a VM. A tenant can specify which kind of packets should be blocked (i.e., packets from, packets to, or both). The operation of these two strategies are shown in Figure 3.2.

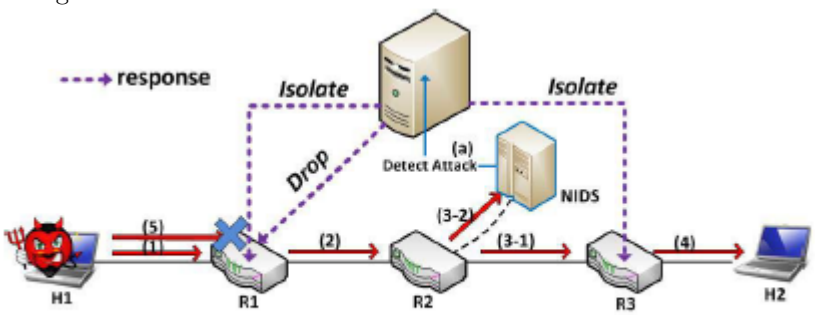

Fig.3.2:Response Function - passive mode drop/isolate(attack)[1].

If a host H1 sends a malicious packet (1) to a host H2, this packet will be mirrored by a router (3-2) and delivered to a NIDS. Then, NIDS detects this as malicious and reports to NETSECVISOR (a). However, since NIDS is installed in passive mode (mirror network traffic), it does not interfere network flows, thus, the packet is also delivered to a target host H2 (3-1)

(4). If NETSECVISOR receives an alert from NIDS, NETSECVISOR will enforce flow rules to router R1 for dropping packets from H1 (drop strategy)or enforce flow rules to router R1 (or R3) for dropping any packets from/to H1 (or H2)4 (isolate strategy). NETSECVISOR also provides three interesting in-line mode response strategies: (i) drop, (ii) isolate, and (iii) redirect.In the case of in-line strategies, NETSECVISOR holds network packets until security devices send their decision results.

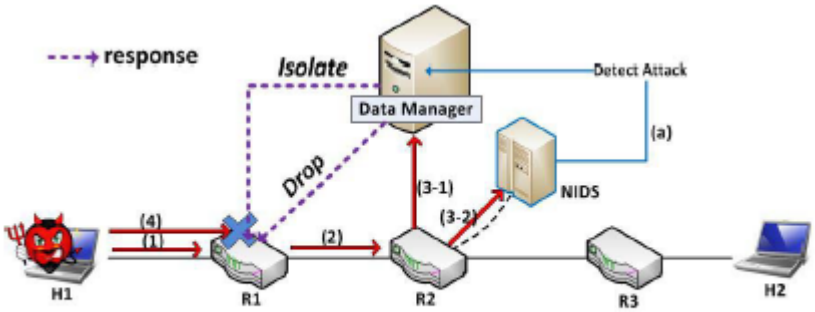

Fig.3.3:Response function-inline mode drop/isolate(attack)[1]

If NETSECVISOR receives detected results and the response strategy for these packets is in-line mode drop, it discards all packets. Thus, a host, which an attacker tries to infect, does not receive any malicious packets. This action is denoted in Figure 3.4. In this case, a malicious host H1 sends an attack packet (1) to H2, then the packet is delivered to a NIDS (3-2) and hold by data manager module (3-1). If NIDS decides the packet is malicious (a), NETSECVISOR discards held packets, and it also enforces flow rules to a router for drop/isolate strategy. If NIDS considers the packet is benign (this scenario is depicted in Figure 3.3), the held packets will be forwarded to a target host (H2) automatically (4) and (5).

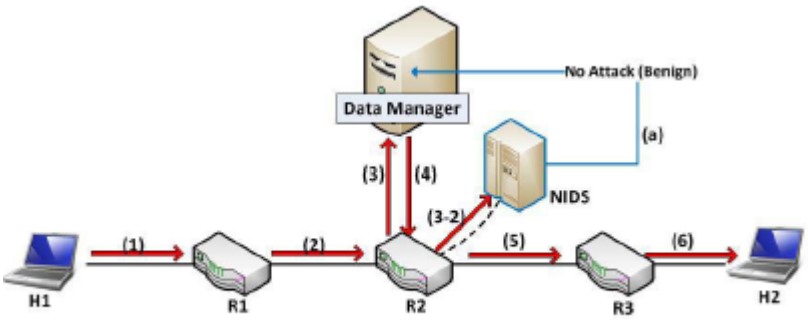

Fig.3.4:Response function-inline mode forward(benign)[1]

Both drop and isolate strategies are similar to passive mode except the in-line mode guarantees that no malicious packets are delivered to victims. Redirect strategy is different from drop/isolate, and it does not drop packets but detours packets to another networks or hosts. To handle malicious packets, honeyhost are employed, which is a decoy host imitating a normal one. Redirect strategy can be used to redirect malicious traffic to the honeyhost transparently (without changing any network configurations or applications). Like other in-line mode strategies, redirect strategy also holds packets in data manager, and if security devices detect some malicious packets, they will be redirected to another host. Figure 3.5 shows this scenario. Here, the held packets (4) and detected packets (7) are redirected to a honeyhost for further investigation.

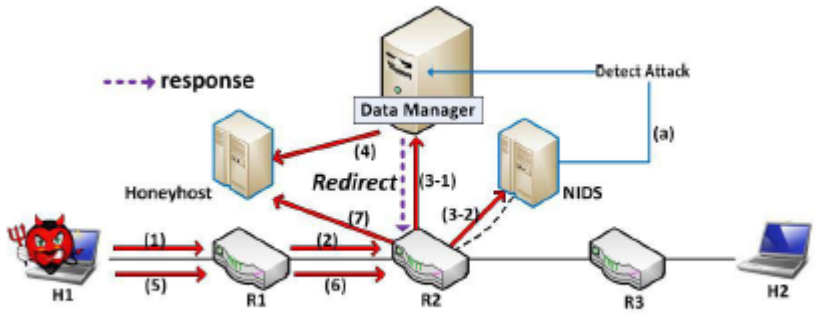

Fig.3.5:Response function-inline mode redirect(attack)[1]

Chapter 4
Algorithm/Protocol used

If NETSECVISOR finds network packets meeting a flow condition specified by a security policy, then it will route these packets to satisfy security requirements.When NETSECVISOR routes network packets, it should consider the following two conditions:
(i) network packets should pass through specific security devices to meet the security requirements, and
(ii) the created routing paths for network packets should be optimized.
There are several existing routing algorithms for intradomain (e.g., OSPF [13]) to find optimal paths. However, they can not be employed directly for this case. Since network packets only contain the source and destination information, existing routing approaches can not discover necessary ways to locations where security devices are installed. Thus, new approaches are required to be designed.
What kinds of basic properties should be in the routing algorithms? As NETSECVISOR supports two modes of security devices which are passive mode and in-line mode. For a passive mode device, the traffic can be forwarded to pass through the device, or just mirror a duplicate to the device and forward the original traffic in another way. For an in-line mode device, all traffic should pass through and be monitored by this device. The generated routing path should satisfy the requirements from different modes of security devices. Besides, a network may contain only passive mode devices or in-line mode devices, or both the two kinds. Thus, aim is to design different algorithms for different usage scenarios.Recent software-defined networking technologies (e.g., OpenFlow) provide several interesting functions, and one of them is to control network flows.
With the help of this function, 4 different routing algorithms are suggested, which can satisfy different requirements. Also, the following 4 terms are clarified in the interest of explaining algorithms more clearly:
(i) start node, a node sends network packets,
(ii) end node, a node receives the packets,
(iii)security node, a node mirror packets to a passive security devices, and
(iv) security link, a link on which in-line security devices are located. Among the proposed 4 algorithms, 3 of them (i.e., Algorithm 1 - 3) are designed for security policies which only use passive mode devices, and 1 of them (i.e., Algorithm 4) is suggested for policies which have in-line security devices such as a firewall and a NIPS. Here it is important to know that it is not possible to use one particular algorithm for the implementation of NETSECVISOR. As the type of security devices and configurations are different, several different algorithms which are efficient for the system can be used.
To describe algorithms more clearly, path between two nodes is to be determined. A network can be characterized using a graph structure, which consists of nodes (hosts or routers or switches) and arcs (physical links between devices). In this graph structure, some paths between a start node, which sends network packets, and an end node, which receives network packets has to be revealed. At this time, usually the shortest path between a start node and an end node is ascertained to deliver network packets efficiently. The problem of finding the shortest path between two nodes is a type of a linear programming, and it can be formulated as the minimum cost flow problem.
To describe the algorithms more clearly, provide concrete examples are provided to illustrate the key concept of each algorithm. For the illustration, a simple network structure is considered as shown in Figure 4.1(a). It contains six routers (R1 - R6), a start node (S), an end node (E), and a security device (C) attached to node R4 (thus R4 is a security node). Assume that

node S sends packets to node E, and for example, security policy specifies that all packets from node S to node E should be inspected by security device C. Furthermore, Figure 4.1(b) shows the traditional packet delivery based on the shortest path routing without considering the need of security monitoring. Thus, packets from node S are simply delivered through the path of $(S- > R1- > R5- > R6- > E)$, and obviously in this case they can not be inspected by the security device C. Next, the working of algorithm and their illustration on same network structure is described.

Algorithm 1:(MultiPath Naive)

This is a simple algorithm regardless of the path between a start node and an end node. This approach is based on function of open flow which can send network packets to multiple output ports of a router. Thus, NETSECVISOR can simultaneously send network packets to different paths.

1.Find shortest path between start node and an end node.
2.Find short path between a start node and each security node.
3.Deliver packets to all obtained paths.

An example case for this algorithm is shown in Figure 4.1 (c), and here, this algorithm finds the shortest path between S and E for a packet delivery and the other shortest path between S and R4 $(S- > R1- > R2- > R3- > R4)$ for inspection.

Algorithm 2:(Shortest Through)

The second approach is to find the shortest path between a start node and an end node passing through each intermediate security node. Finding this path is more complicated than finding the shortest path between two nodes, because in this case, the established path should include all intermediate nodes.

1.Find all possible connection pairs among all nodes.
2.Investigate shortest path for each pair.
3.Check possible paths between a start node and and end node and generate multiple paths.
4.Find the path which has minimum cost value.

This approach is formalized in Algorithm 2, and an example case is presented in Figure 4.1 (d). In this case, it finds the shortest path between S and E that passes through R4, and the path is like the following $(S- > R1- > R2- > R3- > R4- > R6- > E)$.

Algorithm 3:(Multipath-Shortest)

As mentioned previously, OpenFlow supports the function of sending out network packets to multiple outports of a router simultaneously, and Algorithm 1 is based on this function. However, it may not be efficient, because it can create multiple redundant network flows. Thus, an enhanced version of Algorithm 1 is proposed. The concept of this enhanced algorithm is similar to that of Algorithm 1.

1.Find a node which is closest to a security node and is in the path between a start node and an end node.
2. If node is found, it asks this node to send packets to multiple output ports.
(i) a port, which is connected to a next node in the shortest path, and
(ii) (a) port(s), which is (are) connected to (a) node(s) heading to (a) security node(s).
3.Thus, network packets are delivered through the shortest path, and they are delivered to each security node as well.

Figure 4.1(e) presents an example scenario for this algorithm. It first finds the shortest path between S and E, and it finds the shortest path between R4 and nodes on the found shortest path, which is $R6- > R4$.

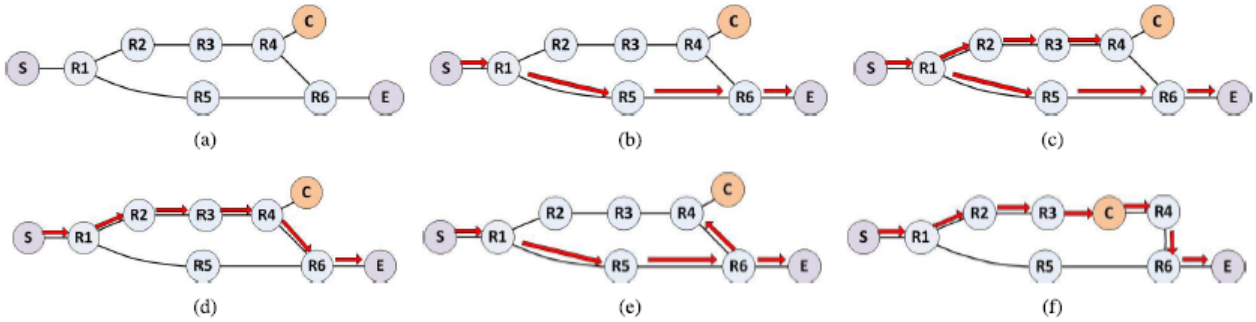

Fig.4.1:Example scenario of each algorithm.(a)layout (b)shortest-path (c)A1:multipath-naive (d)A2:shortest-through (e)A3:multipath-shortest (f)A4:shortest-inline[1]

Algorithm 4:(Shortest-Inline)

Previously, three algorithms are presented, and they are applicable to the case, if security devices attached to security nodes are passively monitor network packets. However, if a security device is installed as in-line mode, the situation should be changed. For passive monitoring devices, simply find a path passing through each security node, however, in the case that there is a security device working in-line mode, it is required to consider both of security nodes and security links (between two nodes). Even though a path includes two nodes for a link, it does not guarantee that the link is used for the path, because each node could be linked to another nodes. To address this issue, Algorithm 2 is refitted to make sure that it should include security links in the generated path. Thus, this Algorithm 4 has a routine checking whether security links are included or not. This algorithm is summarized in Algorithm 4.

An example for this case is presented in Figure 4.1(f), and it shows that there is a security node (C) on the link between R3 and R4. The selected path is the same as the path found in Algorithm 2. However, to find a path considering an inline device, make sure that the link, where an inline device is, is on the routing path.

Usage Scenario and Comparison of Each Algorithm:

Algorithm	Pros	Cons	When to use
A1: Multipath-Naive	Simple and fast	Redundant flows	Enough network capacity, delay is important
A2: Shortest-Through	No redundant path	Computation overhead, when multiple devices	Not enough network capacity, delay is not so important
A3: Multipath-Shortest	Efficient routing path	Computation overhead	Not many hops (e.g., communication between inside VMs)
A4: Shortest-Inline	Guarantee passing through a specific link	Computation overhead, when multiple devices	For an inline security device(e.g., IPS)

Table 4.1:Comparison of Each Algorithm[1]

Finally, by comparing each algorithm and presented their pros/cons and suitable using scenarios in Table 4.1. Understanding strong or weak points of each algorithm will help us find a more suitable routing algorithm for specific situation in a cloud network environment. Table 4.1 summarizes the strong or weak points of each algorithm, as well as the recommended scenario to use each algorithm. For example, the advantage of Algorithm 2 (Shortest- Through) is that it will not increase much network traffic and the disadvantage is its relatively high running complexity (actually it has the highest computation complexity among four). It is mostly suitable to use if the overall network capacity is a concern while the communication delay is not a concern.

Algorithm 1 (Multipath-Naive) is suitable for cloud networks with enough capacity, and their communication delay is of importance. Algorithm 3 (Multipath-Shortest) is mostly suitable for relatively short paths without many hops. Algorithm 4 (Shortest-Inline) is obviously suitable for

inline security devices. In fact, if the users want to use in-line devices, system will automatically choose Algorithm 4 for them. In general, the system allows the users to specify/congigure the algorithms they want. Also, the system can automatically choose an algorithm for users if the users simply specify their high-level priorities (e.g., communication or computation cost), based the summary in Table 4.1.

Chapter 5
Theorotical/Experimental study

Implementation

This prototype is implemented on top of the POX controller, a popular lightweight controller system for OpenFlow networks. NETSECVISOR contains approximately 1,200 lines of python code. The two data structures (device table and policy table) in the device and policy manager are implemented as simple hash tables. The response manager is implemented as a simple network server and it opens a network connection to receive detected results from each security device. The data table in the data manager is a simple hash table, whose key is created from network 5-tuple information (i.e., source or destination IP address,source or destination port, and protocol) of a network flow. The routing rule generator firstly uses the topology discovery component in POX to learn the underlying topology of the network as a graph structure.It also collects network status information to estimate cost of each network link through periodically sending query through POX APIs. Finally,the flow rule enforcer translates the routing rules into flow rules and sends the flow rules to relevant switches through POX APIs (e.g., ofpf lowmod)

EVALUATION

A. Evaluation Environment

To verify the feasibility and evaluate the efficiency of NETSECVISOR, three different network topologies are emulated, and two topologies are running on a virtual network environment and one topology is running on a real commercial switch environment.

1.Virtual Network Environment:
Mininet is selected, which is popularly used for emulating OpenFlow network environments, to emulate two different network topologies:
(i) 12-router configuration and
(ii) 64-router configuration.
In each case, one passive mode security device and one in-line mode security device (for evaluating Algorithm 4) is installed. In addition, two hosts are created for a start node and an end node for packet traversal.
Commercial Switch Environment: A 6-router network topology is created as presented in Figure 4 with 6 OpenFlow-enabled switches (2 LinkSys WRT54GL switches and 4 TP-Link TLWR1043ND switches), 3 hosts for NIDS, and 2 hosts for a client and a server.

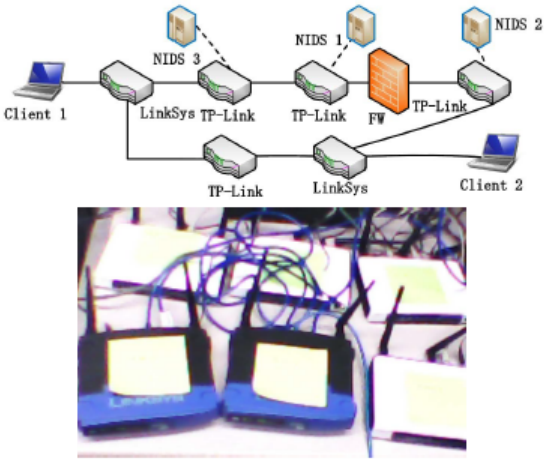

Fig.5.1:6 router testbed[1]

The firmware of LinkSys and TP-Link devices is changed to enable OpenFlow functions, and Snort is installed for NIDS. In the current status,only 6-router network is created because of the hardware limitation. However,more diverse network topologies will be tested with more real network devices in near future.
Generation Time And Network Cost Measurement
Four metrics are measured to estimate the performance overhead of NETSECVISOR. First, measure the routing path generation time of each algorithm. It is an important feature to estimate the performance of NETSECVISOR, because it determines how many network flows can be handled by NETSECVISOR. For example, if NETSECVISOR can create a routing path in 1 ms, it means that NETSECVISOR can handle 1,000 network flows in 1 second. Of course, there are many ways to increase the number of flows for handling, thus, it does not mean that the routing path generation time strictly restricts this number. Second, estimate the network cost that represents total cost of delivering a packet between a start node and an end node.Cost can be estimated by other methods (e.g., mea-

suring network link status). In evaluation, the number of network links between a start node and an end node are measured after generating the flow rule as the network cost. Third, measure the CPU and memory overhead of NETSECVISOR, when it determines a routing path. Fourth, measure the average response time between a client host and a server host, when NETSECVISOR sets up a routing path between them. In this case, to understand the response time when a client sends packets to a server with different routing paths. When calculation of each metric is done, routing algorithms are compared with an simple baseline routing module (except CPU and Memory overhead case). This module is based on the Dijkstra's shortest path algorithm, which is well known and widely used, and this algorithm is denoted as shortest in the test results. Based on the comparison, the overhead of the proposed routing algorithms can be estimated.

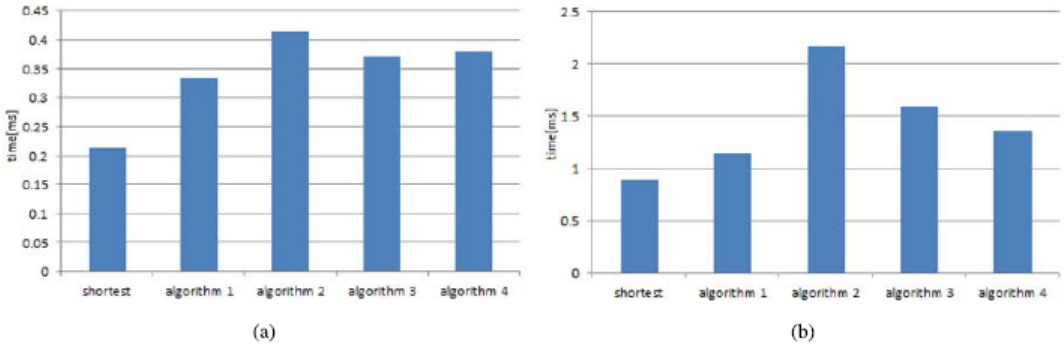

Fig.5.2:Flow Rule Generation Time Measurement (a)16 routers (b)64 routers[1]

Virtual Network Environment:
The results for the routing path generation time are shown in Figure 5.2, and observations are made that the proposed routing algorithms add relatively high overhead (from 20 to 100 percent) compared with the baseline module. However, the argument is that the time is still reasonable because it can still handle around 500 network flows per second in the worst case (i.e., running Algorithm 2 in 64-router configuration), and it can be improved by optimized implementation (all test results are based on the un-optimized prototype system). An interesting thing is that the added relative overheads (in terms of percentage) are decreasing when applied more routers. For example, in the case of 12-router configuration, Algorithm 1 shows nearly 50 percent of overhead, but it is reduced into around 20 percent when 64 routers are used. This is probably because in the case of larger networks, finding the shortest path between nodes (the primitive for both baseline and algorithms) consumes most of time for routing path calculation (instead of new routing calculation). It implies that if applying NETSECVISOR into a large-scale cloud network, the overhead could be reduced significantly compared with the based module.

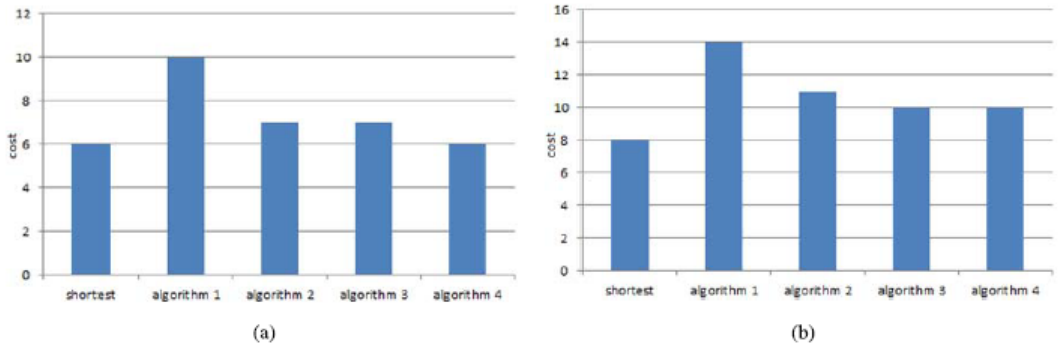

Fig.5.3:Network Cost Measurement (a)12 routers (b)64 routers[1]

Figure 5.3 presents the test results of network cost for each case. Here, Algorithm 1 adds much more overhead than other cases, and it is natural because this algorithm copies network flows to be monitored. However, other algorithms add small overhead, because they try to find the shortest paths for their purpose.
Commercial Switch Environment: Figure 5.4 presents the test results of the routing path generation time (a) and the network cost (b) for the commercial switch environment. In this test, the number of security devices are changed from 1 to 3 to understand the overhead more clearly (In the figure, 1 location denotes that 1 security device is installed). In the case of the routing path generation time, the overhead is somewhat noticeable, which is an expected result from the virtual network environment case.

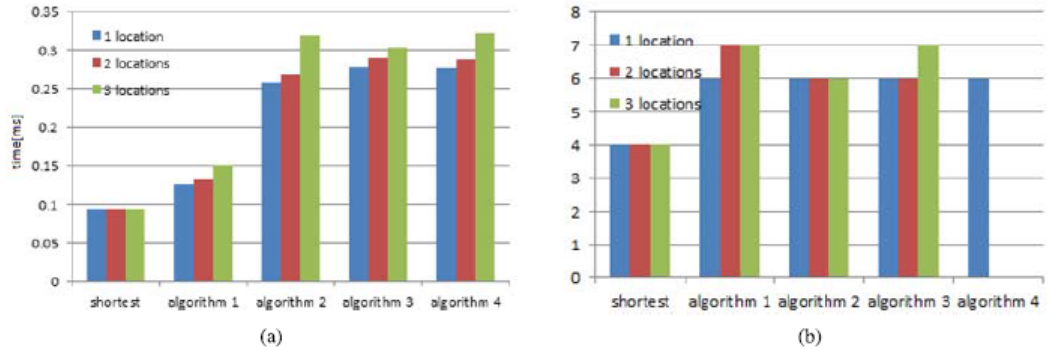

Fig.5.4:Router topology (a)Generation time measurement (b)Network cost measurement[1]

An interesting thing from the results is that even though by installing more security devices into a network, it does not cause much serious overhead. Of course, it is possible that the number of routers for testing is limited (i.e., 6), and thus installing more security devices may not produce complicated path calculations. The test results of network cost are presented in Figure 5.5, and it clearly shows that Algorithm 1 adds more overhead than other algorithms, which is the same result that is found in the above.

CPU and Memory Overhead and Response Time

CPU and Memory Overhead for Virtual Switch Environments:

By evaluating the CPU and memory usage overhead in each topology. The overhead is introduced when NETSECVISOR generates routing paths. The baseline is in idle state and the usage of CPU and memory is zero. For different topologies, calculate the absolute increment of usage when generating routing paths compared to this baseline. Since The CPU and memory usage overhead of each algorithm is found out, case is quite close to each other, the result of Algorithm 4 can be used as a representative example. The overhead results are shown in Figure 5.6, and it clearly states that the memory overhead is no more than 1 percent, which can be ignorable. In addition, these routing algorithms do not produce serious CPU overhead (the 64-router topology only adds about 6 percent overhead). The results imply that algorithms can be optimized by using more CPU power and memory. For example, employ some techniques to hold all (or most of) created routing paths into a memory space to reduce the time for routing path generation (a kind of cache).

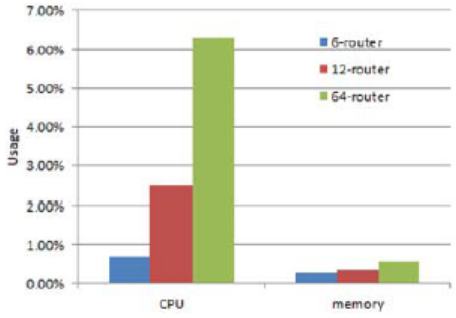

Fig.5.5:CPU and Memory Overhead[1]

Average Response Time for Commercial Switch Environment: To measure the average response time between two peers (i.e., Client 1 and Client 2 in Figure 5.1(left))for each algorithm, set up routing paths between two peers with each algorithm. After generating the routing paths, send 60 ping from Client 1 to Client 2 to measure the average response time. In this case, the routing paths for each algorithm are the same as the paths in Figure 2, and the test results are shown in Figure 5.6. The results clearly show that if Algorithm 1 or Algorithm 3 is used, a client cannot feel any additional delay, and that is because these two algorithms guarantee that they deliver packets from the client to the server through the shortest path. Other two algorithms (2 and 4) add a little more delays compared with other methods, and it is very natural because they take longer paths.

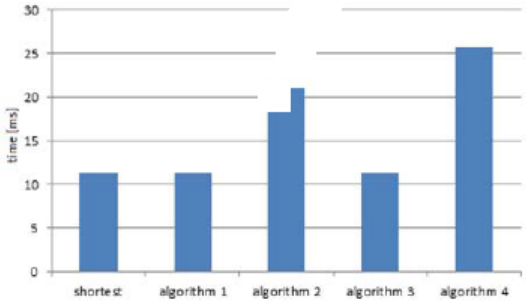

Fig.5.6:Response Time[1]

Discussion on Scalability

NETSECVISOR is designed to scale to cloud level network. A large cloud network often contains millions of tenants and VMs. That introduces a data plan scalability issue. There may be tens of thousands of security requests generated per second. Generating routing paths for all of them can be a huge burden for NETSECVISOR. A single instance of NETSECVISOR may not be able to handle that. Meanwhile, the inherent bottleneck of the flow rules amount in the data plane is another possible limitation. Some optimization methods are required to make this system scale well. To solve this issue, some existing research solutions can be leveraged. To address the bottleneck of generating routing paths, generating of each routing path can be done separately and asynchronously, which means system can run in a distributed manner with the help of existing distributed SDN controller platforms to generate the routing path.It is also possible have multiple NETSECVISOR working in parallel. Thus, separately and asynchronously generating of routing paths for different policies could be a potential solution to the scalability challenge. To address the bottleneck of data plane flow entries, scalable and efficient networking system could be implemented in some heavily loaded nodes (e.g. aggregation switches, ToR switches).

Chapter 6
Advantages and Disadvantages

6.1.ADVANTAGES

1. Easy,flexible and efficient security management.
2. Proper utilization of security devices/middle-boxes.
3. Abstraction of security resources.
4. NETSECVISOR provide on-demand, flexible and dynamic security services to the user.

6.2.DISADVANTAGES

1. In some cases, NETSECVISOR may not generate routing paths.
2. NETSECVISOR may suffer from mistakes of tenants.
3. NETSECVISOR increases the trusted computing base (TCB) of the network controller.

Conclusion

New concept of network security virtualization is introduced which virtualizes pre-installed security devices/functions and provide network response function whenever necessary from security devices. New prototype system i.e. NETSECVISOR is proposed to illustrate the utility of Network Security Virtualization i.e.NSV. This system is evaluated on both i.e. virtual networks and commodity networks. And with the help of results, I can conclude that NSV is a right step towards building more secure, extensible and dynamic network environments efficiently and effectively.

Futurescope

There are several limitations in current work which can be overcome in future. First, there could be some cases that NETSECVISOR cannot generate routing paths. For example if a network administrator specifies two security devices that cannot communicate with each other, algorithms will fail in generating paths. However, NETSECVISOR can still show a warning message for this physically impossible routing path.
Second, NETSECVISOR may suffer from tenants' mistakes or even malicious tenants if they register misconnfigured or malicious policies. NETSECVISOR increases the trusted computing base (TCB) of the network controller (POX in the current implementation). It is also possible that network controllers such as POX contain vulnerabilities that can be exploited by attackers to change the policies/flow rules. This is a separate research direction that worth further investigation from the research community (one of future work). This issue is clearly out of the scope of NETSECVISOR and future work is needed in this area. Third, currently system is tested only in a small research environment with no more than 64 switches. In future work, plan is to evaluate in a larger-scale environment (e.g. GENI, enterprise clouds). When dealing with a set of various devices,it is necessary to chain the services, which may involve more issues such as the order of services in chaining, the potential routing conflict/confusion for different devices. These interesting research problems can be solved in the future.

Appendix

References

[1]Seungwon Shin,Haopei Wang and Guofei Gu, "A First Step Toward Network Security Virtualization : From Concept To Prototype", in IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY,VOL. 10,NO.10,OCTOBER 2015.
[2]Philip Porras, Seungwon Shin, Vinod Yegneswaran, Martin Fong, Mabry Tyson, Guofei Gu, "A Security Enforcement Kernel for OpenFlow Networks", in HotSDN'12,August 13,2012,Helsinki,Finland.
[3]Sayed Kaveh Fayazbakhsh, Vyas Sekar, Minlan Yu, Jeffrey C. Mogul, "Flow-Tags:Enforcing Nework-Wide Policies in the presence of Dynamic Middlebox Action", in HotSDN'13,August 16,2013,Hong Kong, China.
[4]Zafar Ayyub Qazi, Cheng Chun Tu, Luis Chiang, Rui Miao, Vyas Sekar, Minlan Yu, "SIMPLE-fying Middlebox Policy Enforcemenr using SDN", in SIGCOMM'13 August 12-16,2013, Hong Kong, China.
[5]OpenFlow."Open Networking Foundation. [online]. Available: https://www.opennetworking.org/sdnresources/openflow"
[6]V. Heorhiadi, V. Sekar, and M. K. Reiter, New opportunities for load balancing in network-wide intrusion detection systems, in Proc. ACM CoNEXT, 2012, pp. 361372.
[7]J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, Making middleboxes someone elses problem: Network processing as a cloud service, in Proc. ACM SIGCOMM, 2012, pp. 1324
[8]N. Foster, M. J. Freedman, R. Harrison, J. Rexford, M. L. Meola, and D. Walker, Frenetic: A high-level language for OpenFlow networks, in Proc. ACM Workshop Program. Routers Extensible Services Tomorrow (PRESTO), 2010, Art. ID 6.
[9]ETSI. Network Function virtualization. [Online]. Available: http://portal.etsi.org/NFV/, accessed Mar. 10, 2016.

List of Figure

List of Tables