ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)
Vol 3, Issue 2, February 2018

Network Security Virtualization

Vrushali N. Huchhe
[1] Department of Computer Science and Engineering Marathwada Shikshan Prasarak Mandal's Deogiri Institute of Engineering & Management Studies, Aurangabad Maharashtra State, India 2017-2018

Abstract: **Network security management became more complex in recent years due to the necessity of deploying more network security devices at different positions/sites inside the already complex networks. The flexible transition and maximum usage of correct security devices at right places at a required time with minimal management price is very difficult. NSV presents a concept of network security virtualization which virtualizes security resources to network administrators' users and thus maximally uses pre-installed security devices. It's also able to provide security protection to required networks with minimum management price. For verification of the concept, there is a prototype system NETSECVISOR which do the maximum use of existing fixed position security devices and maximally uses software-defined networking technology to virtualize network security functions. NETSECVISOR contains-**

(1) a simple script language to record security services and policies

(2) a set of routing algorithms to decide shortest routing paths for different security policies based on different requirement and

(3) a set of security response functions to handle security incidents. NETSECVISOR can be deploys in both virtual test networks and a commercial switch networks to evaluate its performance and feasibility The evaluation results show that the prototype only adds a very small overhead while providing required network security virtualization to network users/administrators.

***Keywords*: SDN, NETSECVISOR, Open flow.**

I. INTRODUCTION

As the requirement of network services, are increasing network structures are became more complicated.

A cloud network shows this type of complicated network management with different security demands in different internal sub-networks. A typical cloud network commonly consists of a large amount of hosts and network devices to provide services to a large number of dynamic tenants, each having a logically separated network. It becomes more complex if there are extra middle-boxes in the network environment. Today, many security devices are deployed to improve the performance, sturdiness, and security of networks. The security devices can provide many benefits to networks, but it makes the network more complex to handle. So, there is need to solve this problem.

An additional security device makes network security more complex. Also the security devices have many different security functions to solve different problems. For example, firewall & Network intrusion detection system (NIDS) controls the network access & observes the attacks. So, the network administrator should select appropriate security functions/devices and employ them into proper places. It is very difficult for the administrator, to forecast network attacks of various network tenants and the administrator cannot know the requirements of different tenants before. So, the fixed security devices cannot be in the appropriate positions that can provide the different security requirement of different network users.

Therefore, it is required to leverage fixed security devices, as Well as abstract these security devices to serve an interface for network users.

Therefore, there is a new concept of Network Security Virtualization (NSV) that maximally uses fixed location, security devices and provides active, flexible, and on-demand security services to the tenants. Therefore, users do not need to aware the correct position of every security devices.

NSV has two techniques

(i) Clearly manage the flows to desired network

Security services, and

(ii) Provides network security response Functions on a network device.

It maximizes the utilization of pre-installed security devices, NSV clearly redirect network flows to desired security devices when needed. For example, if a security policy wants that a network flow should be monitored by a security service, NSV technology reroutes the flow to the mentioned security middle devices. It provides a security reaction on each network device. Latest techniques provide a method to manage network flows actively at a network device, e.g. SDN; can understand some basic security reply functions at a network device. It can conduct required security response functions on a network device when needed.

II. LITERATURE SURVEY

In [1] paper, R. Ballard proposed Open SAFE, a system which allows the random way of traffic for security observing applications at line rates. It also presents a flow specification

language ALARMS which handles management of network monitoring appliances very easily. It shows a validation of currently undertaking to observe traffic across the network. Open SAFE has three components: a set of design abstractions about the flow of network traffic; ALARMS (A flow specification language, and an Open Flow component which implements the policy. For the ease of handling monitoring architecture to the network administrators, it uses ALARMS, a language for random route management for security traffic. ALARMS utilize the abstractions to create simple policy language syntax to describe paths. Paths are defined between named components, and each component cause to a distribution rule in the situation of multiple, parallel components. ALARMS are a high-level programming language which depends on a low-level programmatic interface to a network switch.

In [2] Sekar explores NIDS or NIPS deployment through discerning monitoring packets at diverse nodes. In this paper, V. Sekar describes a design that leverage spatial, network wide chances for sharing NIDS and NIPS functions. In case of NIDS, it assures that no node is overloaded by using a linear programming arrangement while giving detection responsibilities to nodes. It shows a prototype NIDS implementation to examine traffic per these assignments, and shows that the approach can be achieved. In case of NIPS, it presents how to do the maximum use of specialized hardware (e.g., TCAMs) to decrease the outline of redundant traffic on the network. These hardware conditions make the optimization problem NP-hard, and also give practical nearly exact algorithms based on randomized rounding. In this paper, a systematic formulation is given for effectively handling NIDS and NIPS deployments. Network-wide organized method, is used where various NIDS/NIPS abilities can be optimally shared throughout the various network locations depending on the operating conditions – traffic profiles, routing policies, and the resources ready at each location.

In [3], an infrastructure is proposed for Network-wide NIDS deployment that maximally uses three scaling opportunities: on-path sharing to divide responsibilities, repeating traffic to NIDS clusters, and collecting together intermediate results to divide expensive NIDS processing. It is challenging to equalize both the compute load across the network and the total communication cost incurred via replication and aggregation. It implements a backwards-compatible method to enable existing NIDS architecture to maximally use these benefits. It shows that the proposed method can significantly decrease the maximum computation time, also provides best elasticity under traffic variability, and gives enhanced detection coverage. A general NIDS design is proposed to maximum use of three opportunities: offloading processing to other nodes on a packet's routing path, traffic replication to off-path nodes (e.g., to NIDS clusters), and aggregation to split expensive NIDS tasks. It allows networks to understand these benefits with fewer changes to existing NIDS software. Many real-world arrangement of networks show that this system decrease the maximum compute load substantially, provides best elasticity under traffic variability, and offers improved detection coverage.

In [4] paper, describes FRESCO, an Open Flow security application development framework proposed to simplify the process of rapid design, and modular arrangement of OF-enabled detection and mitigation modules. FRESCO, is an Open Flow application, which gives a Click-inspired programming framework that allows security researchers to implement, share, and compose together, various security detection and mitigation modules. It shows the application of FRESCO with the implementation of many well-known security defenses as Open Flow security services, and use them to analyze various performance and efficiency of proposed framework. FRESCO is used to solve the issues that can accelerate the constitution of new OF-enabled security services. FRESCO exports a scripting API that allows security practitioners to code security monitoring and threat detection logic as modular libraries. These modular libraries present the basic processing units in FRESCO, and may be distributed and connected together to provide complicated network security applications. It presents the FRESCO security enforcement kernel. It shows that FRESCO produces less overhead and allows active creation of well-known security functions with substantially fewer lines of code.

In [5] consider two aspects of Open Flow that accept security challenges, and propose two solutions that could solve the problem. The first challenge is the inherent communication traffic constriction that comes between the data plane and the control plane, which an opponent could take advantage by supporting device a control plane saturation attack that interrupts network operations. Even well mined relating to conflict models, such as scanning or denial-of-service (DoS) activity, can produce more strong impact on Open Flow networks than usual networks. To solve this problem, introduced an extension to the Open Flow data plane called connection relocation, which considerably decreases the amount of data to- control-plane communications that comes during such attacks. The second problem is that of allowing the control plane to expedite detection of, and reaction to, the changing flow dynamics within the data plane. For this introduced actuating triggers over the data plane's existing statistics collection services. These triggers are fixed by control layer applications to both record for asynchronous call

ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)
Vol 3, Issue 2, February 2018

backs, and fix conditional flow rules that are only activated when a trigger condition is validated within the data plane's statistics module. It describes AVANT-GUARD, an implementation of two data plane extensions, explains the performance impact, and analyze its use for forming more scalable and flexible SDN security services. The aim of AVANTGUARD is to create SDN security applications more scalable and reactive to active network threats. The challenge, which to be solved here, is the inherent traffic constriction introduced by the interface between the control plane and the data plane that known opponent can take the advantage. Connection migration allows the data plane to shield the control plane from such saturation attacks. The second problem is the issue of reactivates. A SDN security application requires expeditious access to network statistics from the data plane as a method for quickly responding to network threats. To solve this, it introduces actuating triggers that automatically fix flow rules when the network is under illegal coercion.

III. SYSTEM DESIGN

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 is in charge of two main functions. First, it takes the information of security devices from a cloud administrator, and stores that information into a device table in NETSECVISOR for usage. Second, this module also takes security requests from each network users, and it converts them into security policies and registers the policies into a policy table. So, this module has two informations: (i) locations/types of security devices from a cloud administrator and (ii) security policies from each user. It makes system to manage network security devices easily.
Response manager takes detection results from security devices, and it allows security response strategies that are mentioned in security policies, when it is required. For e.g. if a user mentions a security policy to drop all corresponding packets when a threat is detected by a NIDS, the response manager will enable drop function to remove network packets belonging to the detected network flows on a network device.
Allowed functions will be recognized as a set of network flow rules, which are sent to routers or switches, and so the system can maximally use each network device as a kind of security device (e.g., firewall).

Routing rule generator forms routing paths to control each network flow. When forming routing paths, this module checks security polices of each user to fulfill their needs. For e.g., if a user defines a security policy that mentions all network flows to port 80 should be checked by a NIDS attached to a router A, then this module created (a) routing path(s) which allows all network packets routing to port 80 pass through the router A. It helps the system assign security needs to each security device depending on value and usefulness

Flow rule enforcer allows flow rules to each OpenFlow router and switch. If the response manager allows response strategies or the routing rule generator creates routing paths, this module converts them into flow rules that could be recognized by OpenFlow routers/switches. After conversion, it sends converted rules to concerning routers or switches.
Data manager acquires network packets from routers or switches to hold up to some security devices send their detection results to NETSECVISOR. It holds packets are for allowing some in-line style security functions as how generic Intrusion Prevention Systems provide. This module does not hold packets all the time, but only captures and stores when needed. The working of NETSECVISOR is as follows. A network administrator records network security devices (both physical devices and virtual appliances) to NETSECVISOR. After registration, cloud users required to create their security requests and send them into NETSECVISOR.. Then, NETSECVISOR parses the submitted security requests to understand the intention of tenants and writes the corresponding security policies to policy table. Next, if NETSECVISOR receives a new flow setup request from a network device, it checks whether this flow is matched with any submitted policies. If it is, NETSECVISOR will create a new routing path and corresponding flow rules for the path. At this time, NETSECVISOR guarantees that the routing path includes required security devices that are defined in a matched policy (i.e., the first NSV function). After this operation, it enforces flow rules to each corresponding network device to forward a network flow. If any of security devices detects malicious connection/content from monitored traffic, they will report this information to NETSECVISOR.

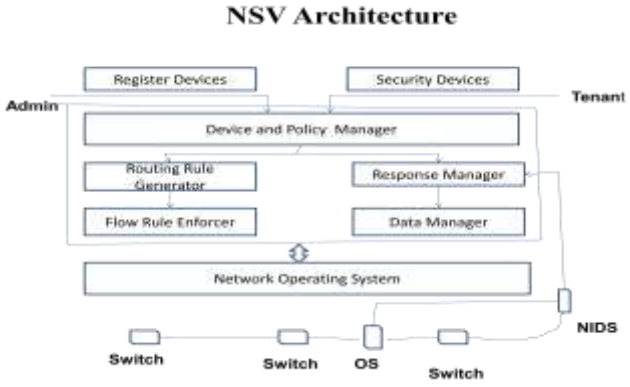

Figure 1: NSV Architecture

3.1. Working of Netsecvisor

A network administrator records network security devices both physical devices and virtual appliances to NETSECVISOR After registration cloud users required to create their security requests and send them into NETSECVISOR Then NETSECVISOR analyzes the submitted security requests to understand the aim of users and writes the corresponding security policies to policy table Next if NETSECVISOR receives a new flow setup request from a network device it checks whether this flow is matched with any submitted policies If it is NETSECVISOR will create a new routing path and corresponding flow rules for the path At this time NETSECVISOR assures that the routing path includes required security devices that are defined in a matched policy i e the first NSV function After this operation it allows flow rules to each corresponding network device to forward a network flow If any of security devices detects malicious connection/content from monitored traffic they will report this information to NETSECVISOR Based on the report and submitted policies NETSECVISOR enables a security response function to respond to malicious flows accordingly.

3.2 Registration of Security Devices

To use pre-installed fixed security devices, a cloud administrator requires recording them to NETSECVISOR using a simple script language. The script language asks for the following information in registration: (i) device ID, (ii) device type (e.g. firewall and IDS), (iii) device location (e.g., attached to a router A), (iv) Device mode (passive or in-line), (v) supported functions (e.g., detect HTTP attacks).

3.3 Creation of Security Policies

After a network administrator record security devices for a cloud network to NETSECVISOR, the information of the recorded security devices is shown to users using the cloud network by NETSECVISOR. Then, the users can define their security requests taking into account recorded security devices and security functions allowed by NETSECVISOR. The script for a request consists of 3 fields: (i) flow condition, which describes the flow to be observed, (ii) function set, which defines the needed security devices for observing or investigating, and (iii) response strategy, which defines how to manage the flow if a threat is detected. The policy syntax is: {{flow condition}, {function-list}, {action-list}}. Currently, NETSECVISOR supports 5 different response strategies and they are drop, isolate for passive mode and drop, isolate, redirect for in-line mode. Here, it provides an example script for the following security request: one user (IP = 10.0.0.1) wants all HTTP traffic regarding to his IP to be observed by a firewall and IDS, and it wants to drop all packets detected 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 user, and it converts them into security policies that can be applicable to a SDN enabled cloud network. At this time, NETSECVISOR requires to convert user described high-level constraints into more specific network level conditions, and it also maps function set into security devices registered before.

3.4 Decision of Routing Paths

If NETSECVISOR finds network packets meeting a flow condition specified by a security policy, then it will direct these packets to satisfy security requirements. When NETSECVISOR forward network packets, it should take into account the following two things: (i) network packets should pass through specific security devices to meet the security needs, and (ii) the produced routing paths for network packets should be optimized. There are various existing routing algorithms for intra-domain to find shortest paths. However, they cannot be used directly for our case. Since network packets only consist of the source and destination information, existing routing methods cannot discover needed ways to locations where security devices are fixed. .
NETSECVISOR supports two modes of security devices which are passive mode and in-line mode. For a passive mode device, it can route the traffic 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 observed by this device. The generated routing path should satisfy the needs from different modes of security devices. Also, a network may contain only passive mode devices or in-line mode devices, or both the two kinds. Latest software-defined networking technologies (e.g., OpenFlow) provide several interesting functions, and one of them is to control network flows as we want. With the help of this function, we propose 4 different routing algorithms, which can satisfy different requirements. It defines the following 4 terms to explain our 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. To describe the proposed algorithms more clearly, we will provide concrete examples to illustrate the key concept of each

ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)
Vol 3, Issue 2, February 2018

algorithm. For the illustration, we use a simple network structure as shown in Figure 2.

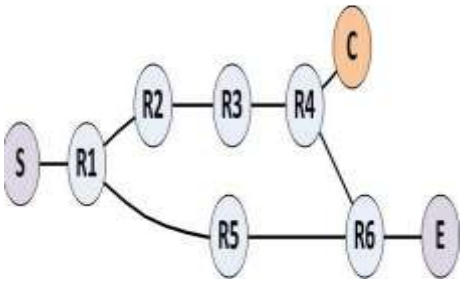

Figure 2: Layout

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). We assume that node S sends packets to node E, and our example security policy is specified that all packets from node S to node E should be inspected by security device C. Furthermore, Figure 3 shows the traditional packet delivery based on the shortest path routing without considering the need of security monitoring.

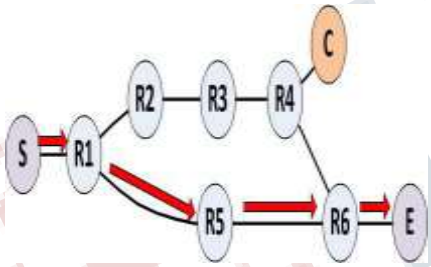

Figure 3: Shortest Path

Thus, packets from node S are simply sent through the path of (S → R1 → R5 → R6 →E), and obviously in this case they cannot be checked by the security device C. Next we will describe how our new algorithms work and illustrate them on the same network structure.

3.4.1 Multipath-Naïve

It is a simple algorithm to visit each security node regardless of the path between a start node and an end node. In this algorithm, NETSECVISOR first finds the shortest path between a start node and an end node. Then, NETSECVISOR also finds the shortest paths between a start node and each security node. If NETSECVISOR has all paths, it sends packets to all obtained paths. This method is based on a function of Open- Flow, which can send network packets to multiple output ports of a router. Thus, NETSECVISOR can send network packets to different paths simultaneously. This method is summarized in Algorithm 1.

Input: S (start node)
Input: E (end node)
Input: Ci = security node i , i = 1, 2, 3, .., n
Output: FPm (multiple shortest paths)
P0 ← find_shortest_path(S, E);
foreach Ci do
Pi ← find_shortest_path(S, Ci);
foreach Pi do
if Pi _⊂ P0 then
FPm ← Pi ;
foreach Pj do
if i _= j and Pi _⊂ Pj then
FPm ← Pi ;

An example case for this algorithm is shown in Figure 4, 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.

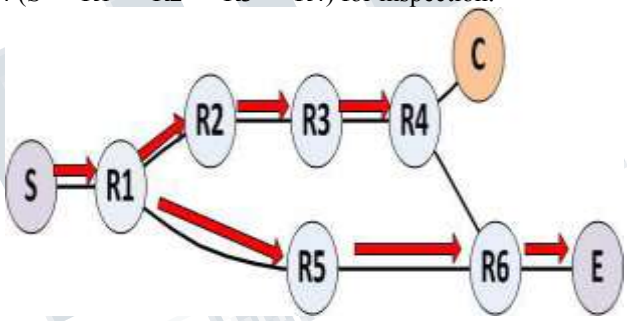

Figure 4: Multipath-Naïve

3.4.2. Shortest-Through

The second approach is to discover the shortest path between a start node and an end node passing through each intermediate security node. Finding this path is more complex than finding the shortest path between two nodes, because in this case, it should make sure that the found path includes all intermediate nodes. To do this, NETSECVISOR finds all possible connection pairs (e.g., if there are multiple security nodes, (a start node, a security node 1) and (a security node 1, a security node 2)) among all nodes including a start, an end, and security nodes,
and then, it investigates the shortest paths of each pair. After this operation, it checks possible paths between a start node and an end node, and it could generate multiple paths. Finally, NETSECVISOR finds the path which has the minimum cost value. This approach is formalized in Algorithm 2,

Input: S (start node)
Input: E (end node)
Input: Ci = security node i , i = 1, 2, 3, .., n
Output: FP, one shortest path
foreach Ci do
Mi ← Ci ;

ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)
Vol 3, Issue 2, February 2018

Mi ← S;
Mi ← E;
foreach Mj do
foreach Ml do
if Mj _= Ml then
Pj,l ← find_shortest_path(Mi , Ml);
while Rk _= NULL do
Rk ← permutation(M1, M1, ..., Mn+1, Mn+2,);
if first element of Rk = S then
if last element of Rk = E then
Qt ← Rk ;
foreach Qt do
foreach Pj,l do
if Pj,l ⊂ Qt then
T Pt ← Pj, j;
FP = min(T P1,T P2, ...);
an example case is presented in Figure 5.

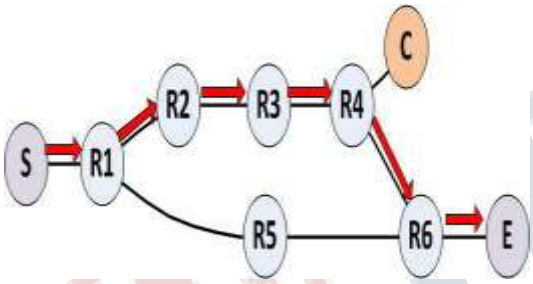

Figure 5: Shortest-Through

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).

3.4.3 Multipath-Shortest

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, we try to propose an enhanced version of Algorithm 1. The concept of this enhanced algorithm is similar to that of Algorithm 1. However, this approach does not find the shortest path between a start node and each security node; instead it finds a node, which is nearest to a security node and in the shortest path between a start node and an end node. If it finds the node, 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). Thus, network packets are delivered through the shortest path, and they are delivered to each security node as well. This approach is presented in Algorithm 3.
Input: S (start node)
Input: E (end node)
Input: Ci = security node i , i = 1, 2, 3, .., n
Output: FPj , multiple shortest paths)
P0 = find_shortest_path(S, E);
FP ← P0;
foreach Ci do
foreach n j in P0 do
T Pi, j ← find_shortest_path(Ci , n j);
FP ← T Pi, j ;

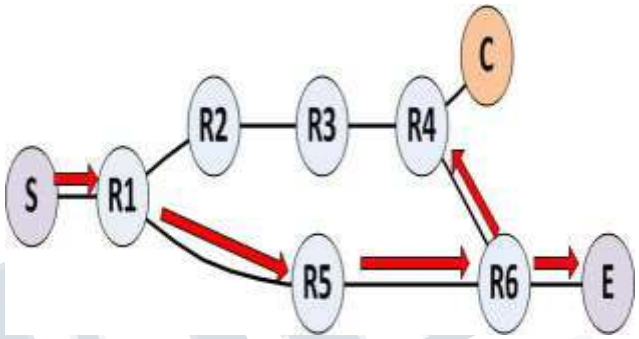

Figure 6: Shortest-Through

Figure 6 presents an example scenario for this algorithm. It first finds the shortest path between S and E, and it discovers the shortest path between R4 and nodes on the found shortest path, which is R6 → R4.

3.4.4. Shortest –inline

If a security device is fixed as in-line mode, the situation should be changed. For passive monitoring devices, we can 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 take into account both of security nodes and security links (between two nodes). Even though a path includes two nodes for a link, it does not assure that the link is used for the path, because each node could be linked to other nodes. To solve this problem, we modify our Algorithm 2 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.
Input: S (start node)
Input: E (end node)
Input: Cim,n= security link i between m node and n node
Output: FP, one shortest path
foreach Ci do
Mi ← Ci ;
Mi ← S;
Mi ← E;
foreach Mj do
foreach Ml do

ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)
Vol 3, Issue 2, February 2018

if Mj _= Ml then
Pj,l ← find_shortest_path(Mj , Ml);
while Rk _= NULL do
Rk ← permutation(M1, M1, ..., Mn+1, Mn+2,);
if first element of Rk = S then
if last element of Rk = E then
foreach Cim,n do
if (m, n) _⊂ Rk then
continue;
Qt ← Rk ;
foreach Qt do
foreach Pj,l d

T Pt ← Pj,l
FP = min (T P1,T P2, ...);
An example for this case is presented in Figure 3 (f), and it shows that there is a security node (C) on the link between R3 and R4.

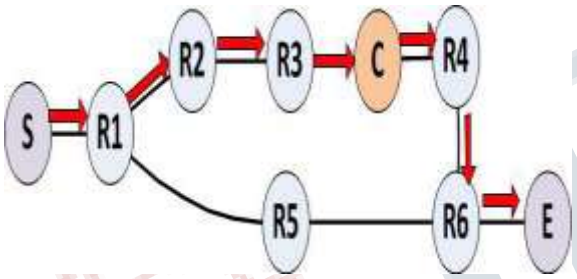

Figure 5: Shortest-inline

The selected path is the same as the path found in Algorithm 2. However, to find a path considering an inline device, we need to make sure that the link, where an inline device is, is on the routing path.

A. Enabling Security Response Functions

NETSECVISOR provides a way of enabling 5 security response strategies, and they do not need adding physical security devices or changing network configurations for managing packets. These methods can be operated into two different modes: (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 sent to a target host.

In this passive mode, NETSECVISOR supports two response strategies. First, NETSECVISOR can drop packets that relate to detected network flows. This strategy is useful to stop some later malicious packets in the flow, but it does not assures 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, and both)

IV. COMPARISON

Comparison of each algorithm and presented their pros/cons and suitable using scenarios in table.

Table 1: Comparison of each algorithm

Algorithm	Pros	Cons	When to use
A1: Mulripath-Naïve	Simple & fast	Redundant flows	Enough n/w capacity, Delay is important
A2:Shortest-Through	No redundant path	Computation overhead, when multiple	Not enough n/w 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 devices(e.g. IPS)

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 1 summarizes the strong or weak points of each algorithm, as well as the recommended scenario to use each algorithm. For example, we can see that 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. 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, our system will automatically select Algorithm 4 for them. In general, the system allows the users to

specify/configure the algorithms they want. Also, the system can automatically select an algorithm for users if the users simply specify their high-level priorities.

V. CONCLUSION

This paper introduces a Concept of Network security virtualization (NSV) that can virtualize security resources/ functions and provide security response functions from network devices when required. It implements a new prototype system NETSECVISOR, to show the application of NSV. NSV prototype system can be used in complex networks like cloud. NSV enables the Administrator a great control over a network infrastructure. NSV can virtualizes specific network functions and allow then to run as individual nodes connecting with other communication and network services.

REFERENCES

1) J. R. Ballard; I Rae, and A. Akella, "Extensible and Scalable network monitoring using openSAFE" in Proc. USENIX Internet Netw. Manage Conf. Res. Enterprise Netw 2010,p.8.

2) 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. 361–372.

3) R. Wang, D. Butnariu, and J. Rexford, "OpenFlow-based server load balancing gone wild," in Proc. 11th Hot-ICE, 2011, p. 12.

4) S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, and M. Tyson, "FRESCO: Modular composable security services for software-defined networks," in Proc. 20th Annu. Netw. Distrib. Syst. Secur. Symp. (NDSS), Feb. 2013, pp. 1–16.

5) S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "AVANT-GUARD: Scalable and vigilant switch flow management in software-defined networks," in Proc. 20th ACM Conf. Comput. Commun. Secur. (CCS), 2013, pp. 413–424.