information

MDPI

Article

Security Policy Scheme for an Efficient Security Architecture in Software-Defined Networking

Woosik Lee and Namgi Kim *

Department of Computer Science, Kyonggi University, Suwon 16227, Korea; wslee@kgu.ac.kr
* Correspondence: ngkim@kgu.ac.kr; Tel.: +82-31-249-9662

Received: 9 April 2017; Accepted: 11 June 2017; Published: 13 June 2017

Abstract: In order to build an efficient security architecture, previous studies have attempted to understand complex system architectures and message flows to detect various attack packets. However, the existing hardware-based single security architecture cannot efficiently handle a complex system structure. To solve this problem, we propose a software-defined networking (SDN) policy-based scheme for an efficient security architecture. The proposed scheme considers four policy functions: separating, chaining, merging, and reordering. If SDN network functions virtualization (NFV) system managers use these policy functions to deploy a security architecture, they only submit some of the requirement documents to the SDN policy-based architecture. After that, the entire security network can be easily built. This paper presents information about the design of a new policy functions model, and it discusses the performance of this model using theoretical analysis.

Keywords: SDN; NFV; security architecture; security policy

1. Introduction

Nowadays, hardware (H/W)-based security architectures are moving toward software (S/W)-based security architectures. Software-defined networking (SDN) [1] makes it possible to implement an S/W security architecture, which consists of a control plane and a data plane. The control plane sends a message about the requirement specifications to the data plane using OpenFlow application program interfaces (APIs). When the data plane receives the requirement message from the control plane, the data plane builds the virtual network topology. Then, the network functions virtualization (NFV) technique [2] is used to deploy virtual security devices on the switches or routers with OpenFlow functions.

Recently, a number of global companies have tried to integrate SDN into their architectures. Cisco proposed using the open networking environment (ONE) technique instead of SDN. The SDN market was expected to reach sales of about $2 billion in 2016 [3]. Moreover, researchers at a number of universities and companies have studied security topics related to SDN. Currently, about 90 companies, including HP, IBM, NEC, and NICIRA, have taken part in open networking foundation (ONF) industry standard institutes, and they have commercialized the controllers and the network virtualization functions [4–6].

However, previous SDN systems have some disadvantages, such as a lack of scalability, performance degradation, difficult network management, and vulnerable security. In order to solve these problems, it is necessary to consider a security SDN-based architecture, which contains solution functions such as efficient security paths, the rearrangement of security topology, and management policies for switches and routers. In this paper, we propose a security policy scheme for an efficient security architecture in SDN. The proposed security policy scheme has four policy configuration functions: separating, chaining, merging, and reordering. If we use these functions, we can easily manage entire security systems.

This paper makes several contributions to the existing literature. First, the proposed SDN policy-based architecture solves the existing network management problem of insufficient scalability, and the meta-model, the pseudocode, and the logic layer related to it are demonstrated. Second, we analyzed the cost of the physical devices using mathematical analysis techniques. Finally, we constructed a simulation to demonstrate the effectiveness of the proposed policy-based SDN technique. For the policy functions of separating, chaining, merging, and reordering, we also demonstrated the utility of policy-based SDN techniques.

This paper has the following structure. In Section 2, we discuss works related to security SDN. Section 3 describes the background of security SDN. Section 4 defines four policy functions: separating, chaining, merging, and reordering. Section 5 explains the security policy meta-model, and Section 6 explains the pseudocode and examples for the four policy functions. In Section 7, we analyze the proposed system theoretically. In Section 8, we analyze experimental results to show that the proposed security policy scheme is superior. Lastly, we summarize this paper and describe the direction of future work.

2. Related Work

Some previous studies have investigated the ability of SDN/NFV systems to defend against DDoS attacks [7–11]. Wan et al. proposed a global system architecture to defend against distributed denial-of-service (DDoS) attacks [7]. This system has two steps: detection and response. In the detection step, the suspicious attack flows are extracted from the received flows based on diverse security-role-mapping. After that, the global system generates alert messages, and sends them to the system's security managers. Khatua et al. proposed the Monitoring and Optimizing Virtual Resources (MOVR) architecture [8]. The MOVR architecture consists of a provisioner, a deployer, a controller, a monitor, a policy-arbitration-Quality of Service (QoS) module, and an application repository. Furthermore, MOVR manages and optimizes the resource use of cloud applications. Aicherry et al. proposed optimal algorithms to solve the problem of optimized virtual machine (VM) placement when the location of the data sets is given [9]. The proposed algorithms try to capture the best tradeoff point between minimizing latencies and incurring bandwidth costs. These algorithms were evaluated and proven using a simulation study. Fischer presented a survey of virtual network embedding (VNE) algorithms [10]. In general, VNE algorithms deal with the main resource allocation challenge in network virtualization. This causes two sub-problems: Virtual Node Mapping (VNoM), where the virtual nodes should be allocated in the physical nodes; and Virtual Link Mapping (VLiM), where the virtual links that connect these nodes should be mapped to paths connecting the corresponding nodes in the substrate network. Chowdhury et al. introduced new VNE algorithms that provided better coordination between the two phases [11]. The proposed algorithms, D-ViNE and R-ViNE, use deterministic and randomized rounding techniques, respectively [11]. That study used a simulation to prove the performance of the proposed algorithms in terms of the acceptance ratio and the revenue of the cost incurred by the substrate network in the long run [11]. Unlike the previously mentioned studies, Chowdhury et al. [12] and Jo et al. [13] focused on security policy based on SDN/NFV architectures. Chowdhury et al. proposed a policy-based inter-domain virtual network (PolyViNE) embedding framework that embeds end-to-end virtual networks in a decentralized manner [12]. They also proposed a forwarding mechanism for the location aware virtual network, based on a hierarchical addressing scheme (COST) and a location awareness protocol (LAP), to rapidly embed virtual networks. Jo et al. proposed a centralized network policy controller to efficiently handle packet flows in a multi-provider network environment [13], and discussed the construction and characteristics of the proposed controller. Moreover, they also provided various examples of virtual scenarios [13]. However, the results of the studies mentioned above are lacking in scalability, because many security vendors adopt different types of security architectures. In order to solve these problems, we propose an SDN policy-based scheme for an efficient security architecture.

There are many more recent studies related to SDN [14–34]. According to Saxena [14], the latest SDN-based trends can be classified as a programming protocol, or as a ClickOS-based or generic SDN security system. We have also analyzed representative elements, such as distributed SDN controllers, Kandoo, DevoFlow, AVANT-GUARD, and FRESCO, as OpenFlow security systems. Satasiya et al. [15] demonstrated SDN security enhancements using firewalls in a distributed scenario environment. In particular, OpenFlow switches efficiently adjusted the firewall policy, changed the policy, and applied the security policy; that study demonstrated the performance through an experiment in the comment window [15]. Shin et al. [16] approached the question of how to use SDN to enhance security. In our study, we analyzed the definition and advantages of each field by classifying them into four SDN features: dynamic flow control, network-wide visibility with centralized control, network programmability, and a simplified data plane. In [18], an example of normal network flow using a simple data flow was shown; it provided an example of a man in the middle of a security attack, and an example of a denial-of-service (DoS). Ranjbar et al. [19] applied the policy-box concept to the Secure Sockets Layer (SSL) protocol to authenticate a hello message using the handshake message technique, thereby enhancing the safety of data transmission and reception. In [21], a security policy checking technique was applied to a flow table by including a conflict detection module and a conflict resolution module to check the security policy efficiently. Similarly, in [22], we discovered and responded to external attacks through a security architecture. Using experiments, [24] showed that throughput can be increased by applying OpenFlow to e-mail transmissions and receptions efficiently. In [25], a new framework was proposed, in which a security agent manages attack detection and monitors topology information, while the application generates blocking flows and reads and filters syslogs in a new type of security framework. Liu et al. [26] presented a very simple form of firewall network address translation (FW-NAT) connection and cache-FW-NAT for security service chaining. In [27], a scenario was presented in which the security of multiple devices is efficiently achieved by applying SDN to various devices in an Internet of Things (IoT) environment. Rawat and Reddy [32] described the basic SDN architecture, and explained intrusion and anomalous attack definitions for DDoS, the architecture's applications, and other energy efficient SDN techniques. In [33], the possibility of SDN was proposed, and various SDN-related topics were addressed.

3. Background

A distributed denial-of-service (DDoS) attack is one type of security attack. There are many cases of this type of security breach, such as the attack on the official residence of the Korean president in 2009, as well as the attack on banks and broadcast websites in 2013. Consequently, the need to construct a security architecture has increased quickly in order to deal with various hacking attacks, including advanced persistent threats (APTs) [35]. In this paper, we consider a target system to be a system that protects against attacks including DDoS attacks.

An SDN/NFV security architecture consists of a control plane and a data plane (Figure 1). The control plane contains the orchestration system and the SDN/NFV controller system, which exchanges control messages with legacy systems. More specifically, the control plane can perform a series of steps to set the path of a data packet flowing into the data plane, and it utilizes the data collected before setting the path. In particular, the control plane includes service chaining. Service chaining can logically apply route changes and ensure security using various types of security policies that exist in the service chaining pool in the data plane.

The data plane builds the security system from the service chaining pool, which has virtual security machines based on the control messages received by the control plane. More specifically, the data plane includes an open flow. The data flowing in the open flow receives a configuration message from the control plane's service chaining, which then creates the network; the configured network then transmits the attack packets sent from the malicious user. It has a security function for processing. Security features can vary for each open switch, and the throughput for attack packets can

also vary from switch to switch. Moreover, the self-defense SDN security system of the data plane, which uses legacy detailed analysis/detection devices, detects anomalous traffic through the sFlow.

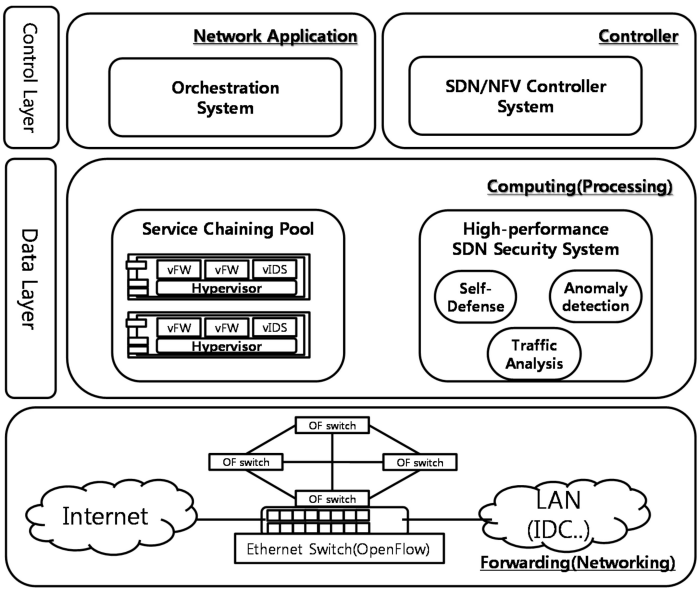

Figure 1. Software-defined networking (SDN)/ network functions virtualization (NFV) System Architecture.

For example, suppose computer company A is preparing to release a new product. At this time, computer company A places DDoS attacks on computer companies B and C so that B and C cannot transmit bad content to A's products on the same day. At this time, due to the DDoS attack of A, the systems of companies B and C become paralyzed, and company A can release its new product safely. In this scenario, when the SDN security system is applied, the DDoS attack is detected efficiently, and the attack data is efficiently solved by distributing the paths of the attacked companies B and C. All of the other network attacks can be efficiently solved using the SDN security system.

4. Security Policy Scheme Security Architecture

We propose a novel security architecture using various policies based on SDN (Figure 1). This architecture consists of four layers: the presentation layer, the application layer, the control layer, and the infrastructure layer. In the presentation layer, system managers can approach diverse services through the web or the console terminals. In the application layer, we deploy our proposed policy configurable methods to easily manage security systems. In this layer, a variety of commands can be given to the control layer through service channels in order to configure the system. The control layer contains several functions, such as the OpenFlow APIs, service chaining, self-defense, and legacy devices. Lastly, the infrastructure layer builds the security networks based on the information received by the control layer.

The policy configurable schemes are deployed on the application layer of the policy security architecture. This architecture lays the foundation for utilizing four policy functions by applying a virtualized network function (VNF) like Figure 2. It provides four policy configurable functions, as follows:

- Separating: this divides the virtual services and decreases the size of the attack flows using the load balancer.
- Chaining: this links many VNFs to prevent various attack flows and constructs big security systems.

- Merging: this combines unnecessary VNFs to optimize the security system and the system's resources.
- Reordering: this reorders current VNFs depending on the type and strength of the current attack flows.

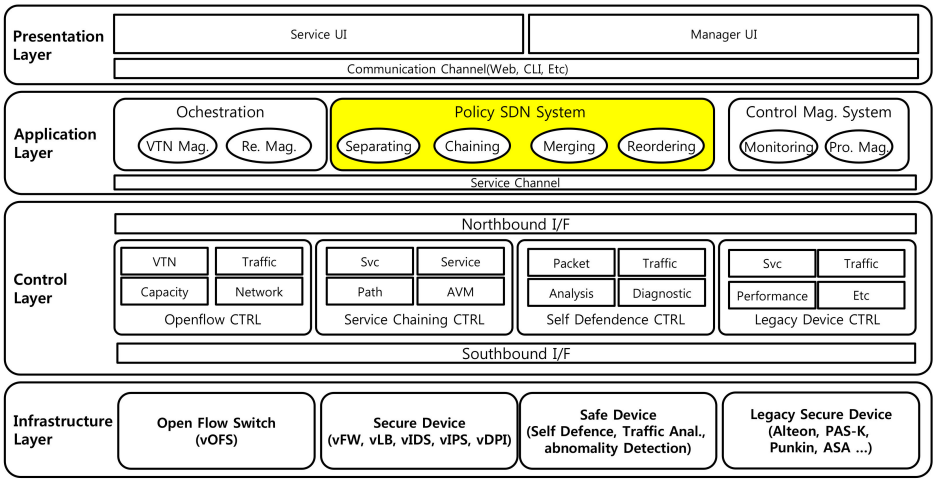

Figure 2. Policy security architecture.

We can easily manage the virtual security network system using SDN-based, security configurable policy functions. Therefore, if SDN system managers use an SDN policy-based security architecture, they do not need to spend time trying to understand the complex system's structure. Moreover, they can easily prepare security methods based on different types of attack flows.

5. Security Policy Meta-Model

If system managers want to use the policy configurable methods of an SDN-based architecture, they should consider various configurable uses based on the type of attack flows. Then, the security policy meta-model is required for setting up an SDN policy-based architecture. In Figure 3, the security policy meta-model consists of an SDN management policy, SDN management policy conditions, and the entities of the SDN management policy actions.

In the entities of the SDN management policy actions, the SDN management policy entity contains actions for mapping particular conditions related to various attack flows. The subordinate entities of the SDN manage five conditions: the time, the location, the target, the report, and the event. The time condition configures the data collection time—the gap between the communication times—to obtain the security state information. Here, the security state information means a type of network attack, a pattern of a network attack time, an attack position, and an attack amount, in order to select a security policy. For example, if a data packet arrives every five seconds and the data packet suddenly flows in the same pattern every second, the network management device determines the packet as being abnormal, and it informs the network security manager about the security state information.

The location condition configures the information about including or excluding areas to restrict the security location. The target condition has a role that is similar to the location condition. It configures the including or excluding devices of the security targets. The report condition configures the type of virtual security status information. The event condition configures the event occurrence conditions based on the current security states. Then, the SDN management policy actions send the resulting, refined information about the policy conditions to the action destinations.

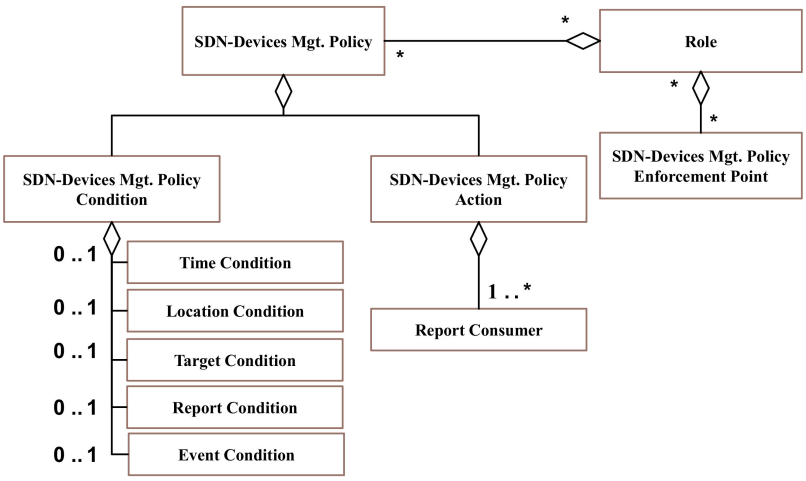

Figure 3. Security Policy Scheme security architecture.

Depending on the policy, the role is given individually, and the role of the action that takes an action against the corresponding attack flow is given according to the role. The role is given a Policy Enforcement Point (PEP) based on when the attack comes in. Also, when a real-time data packet is checked, and the policy is changed and the processing for the attack is not necessary, the PEP is lost and the corresponding role is not performed. When a new attack comes in again, the PEP is given. This type of iteration takes advantage of Role-Based Access Control (RBAC), which provides a new role without redundant privileges.

6. Security Policy Schemes

This section presents a discussion of the pseudocodes, and provides examples of four security policies—separating, chaining, merging, and reordering—in order to demonstrate how to use the policy configurable methods in a real security infrastructure.

Definition 1 defines a data packet. A data packet is a combination of a normal packet and an attack packet. If a malicious attacker is not present in the network environment, only normal packets flow. However, if a malicious attacker is present in the network environment, and if an attack packet exists, a normal packet and an attack packet flow together. In this case, the normal packet is represented in the form of a normal packet. For example, 50 is a normal packet, based on a total of 100 packets. Similarly, if there is an attack packet, it is expressed as (normal packet)/(attack packet). In a network environment where 30 attack packets and 60 normal packets are transmitted based on 100 packets, the data packets are expressed as 60/30 (60 normal packets/30 attack packets).

In this paper, a data packet is regarded as being a combination of a normal flow and an attack flow. A data packet is expressed as a normal flow using a mathematical expression method. It is expressed as (normal flow)/ (attack flow) when an attack flow is also included in the data packet.

Definition 2 defines the network as containing a physical layer and a virtual layer. A device without a security function is placed in the physical layer, and represented by using the symbol N (number). For example, if there are three devices in the physical layer, N1, N2, and N3 are placed in that layer. The virtual layer is a device with security functions, and it is expressed as S (number). The security processing capability of the devices placed in the virtual layer is expressed in the form of Process: (number). The virtual device may be temporarily stripped of its physical layer; in that case, the SN (number) symbol is used. A device belonging to a physical layer does not have the ability to process an attack flow, whereas a device belonging to a virtual layer does have that capability. Attack flow processing capability is expressed in the form of Process: (number). In the case of an SN node, it is used in the same form as a virtual layer device.

The basic resource of the device placed in the physical layer of the system discussed in this paper is set to 100. The cost of processing based on the data packet flowing to each device is calculated using Formula (1). Then, according to the calculation's result, the cost that is consumed by matching the information presented in Table 1 is calculated.

Definition 3 further defines the physical layer. In the case of a device belonging to the device, it is assumed that it, basically, has 100 resources, and the amount of data added to each device is calculated by using Formula (1). The costs are calculated based on the values obtained from that formula, as shown in Table 1.

Formula (1) shows the formula for calculating the cost. The input data x of Formula (1) means the size of the data packet flowing to the security equipment. x is placed in the molecule of the formula. Moreover, the denominator of the formula is the amount of device capacity that the security device can handle. Using the output data from Formula (1), we can get the cost value from Table 1.

Formula (1) yields a value based on the amount of data flowing into the device versus the resources the device can accommodate. In the proposed architecture discussed in this paper, since the device capacity is limited to 100, the denominator value is always fixed at 100. For example, if 50 data packets are flowing through the device, the value of 50/100 = 0.5 is given by Formula (1).

$$\phi(x) = \frac{x}{device\ capacity} \tag{1}$$

The values obtained from Formula (1) are calculated based on the information presented in Table 1. The cost calculation is based on [36], which shows that the value increases sharply on the basis of 1.

Table 1. Experimental parameters.

Min.	Max.	Cost
0	1/3	1
3/1	2/3	3
2/3	9/10	10
9/10	1	70
1	11/10	500
11/10	∞	5000

Figure 4 shows the pseudocode of the separating configurable method. As seen in the discussion of the relationship between the available resources and the loads of attack flows presented in Section 6, the separating method adds a new security device if the size of the attack flow exceeds 1. If that occurs, it reconfigures the previous network paths to stop massive attack flows. More specifically, the Separating () function gets input data which is an attack flow. After that, the Separating () function uses Formula (1) to obtain the cost. If the cost is 1 or more, the Separating () function creates a new virtual device through vDevice (). Then, it places the newly created vDevice by calling the devices () function in the virtual network topology. Then, the path () function is called to rearrange the path so that packets can flow efficiently to the newly arranged vDevice.

```
Function Separating()
Input: attack flow: a, target: d
If φ_device(a) > 1 then
newDevice ← new vDevice();
vNetwork.devices(newDevice);
vNetwork.path(d, newDevice);
end if
```

Figure 4. The pseudocode of the separating configurable method.

Figure 5 shows an example of the separating configurable method. As seen in Figure 5a, 100 data packets flow to N2, which belongs to the physical layer. Therefore, based on Formula (1), the cost imposed on N2 is calculated as follows: (100) = 100/100 = 1. If the value is 1, the cost is 500 (Table 1). If the cost is 500, and if the amount of data packets flowing in N2 is effectively reduced, the cost can be drastically reduced. If the SN node is placed in the physical layer, the cost can be reduced. Figure 5b shows how the SN nodes are deployed to distribute the data packets. At this time, N2 and SN1 have a value of ϕ (50) = 50/100 = 0.5, based on Formula (1). According to the information presented in Table 1, the cost is 3; thus, the cost has decreased dramatically.

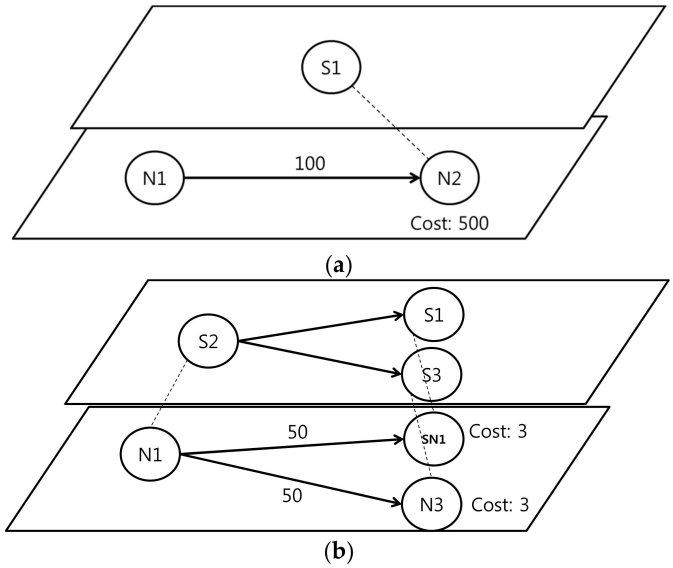

Figure 5. The example of the separating configurable method. (**a**) Network cost problem; (**b**) Solved network cost using separating method.

Figure 6 shows the pseudocode of the chaining configurable method. This method chains virtual security devices to detect and reduce the number of attack flows. If the SDN-based security network of a particular attack flow is already built, the chaining configurable method just passes the received attack flows to previous security systems. If not, it adds a new virtual security device and chains it to other security devices. More specifically, the Chaining () function receives an attack flow as input data. The Chaining () function determines whether the attack flow attack type is included in the current vNetwork. If there is no device capable of handling the attack type, the vDevice () function creates new security equipment of the corresponding type. It puts the created device into the vNetwork to process the attack flow. Otherwise, the Chaining () function will call the corresponding security device and chain it with the existing device, so that the attack flow can be processed.

```
Function Chaining()
Input: attack flow: a, target: d
If vNetwork.check(Type(ϕ_device(a))) != True then
    newDevice ← new vDevice(Type(ϕ_device(a));
    vNetwork.chain(newDevice);
else
    chainD ← vNetwork.find(Type(ϕ_a(x)))
    vNetwork.chain(chainD);
end if
```

Figure 6. The pseudocode of the chaining configurable method.

Figure 7 shows the use of the chaining rule to cope with an unexpected network attack flow. First, a 50/30 data packet flows through N2 placed in the physical layer. Here, 30 is an attack flow, but since S1, which is placed in the virtual layer, does not have a security policy corresponding to 30, a new type of security node is needed. At this time, of course, a new node may be placed in N1 to distribute the data packet or to reduce the amount of data, but it is not easy for the Start node to detect or handle the attack pattern. Therefore, it is efficient to arrange a new node that can process an attack flow between N1 and N2. If a new SN1 node is deployed using the chaining method, the attack packet can be efficiently processed. Therefore, 80 data packets are transmitted through SN1, 50 data packets eliminate the attack packets, and the cost of N3 is reduced to 3.

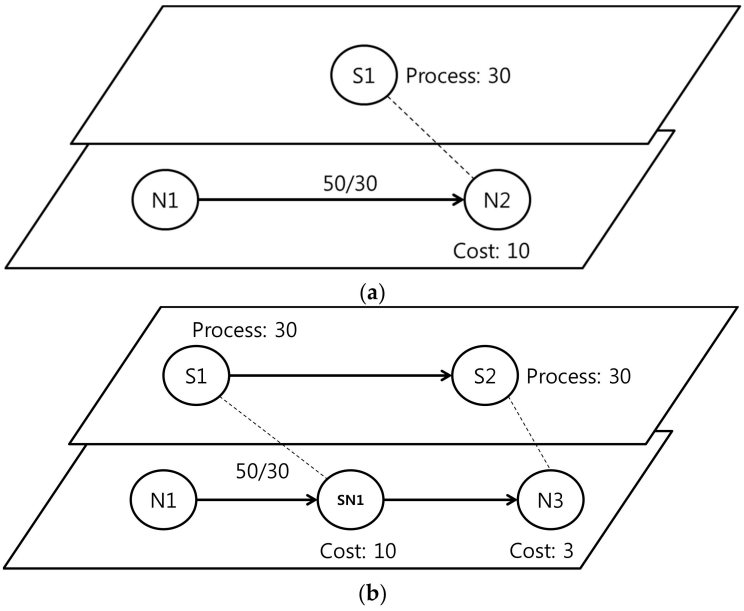

Figure 7. The example of the chaining configurable method. (**a**) Network attack flow problem; (**b**) Solved network cost using chaining configurable method.

Figure 8 shows the pseudocode of the merging configurable method. This method removes some of the previous security network to increase network resource utilization. This algorithm first checks the gap between the size of the existing security network and the current attack flows. If the size of the virtual network is larger than the attack flows, the algorithm removes useless virtual security devices. More specifically, the Merging () function receives a value of attack flow as input data. If the value is smaller than the current capacity of the vNetwork, we know that a large amount of resources are used unnecessarily. At this time, the Merging () function automatically finds unnecessary devices and merges unnecessary processing devices. After that, it relocates the resources through the rearrange () function.

```
Function Merging()
Input: attack flow: a, target: d
If a < vNetwork.resource() then
rest ← vNetwork.resource() − φ_device(a)
vNetwork.remove(rest);
vNetwork.rearrange(rest)
end if
```

Figure 8. The pseudocode of the merging configurable method.

Figure 9 provides an example of the merging configurable method. If all of the nodes process the attack flows, the security nodes are not needed. Unnecessary nodes increase the resource consumption of the hardware. The merging configurable method automatically removes the unnecessary network nodes. As seen in Figure 9b, S1 is removed. Therefore, the proposed SDN policy-based architecture reduces resource consumption.

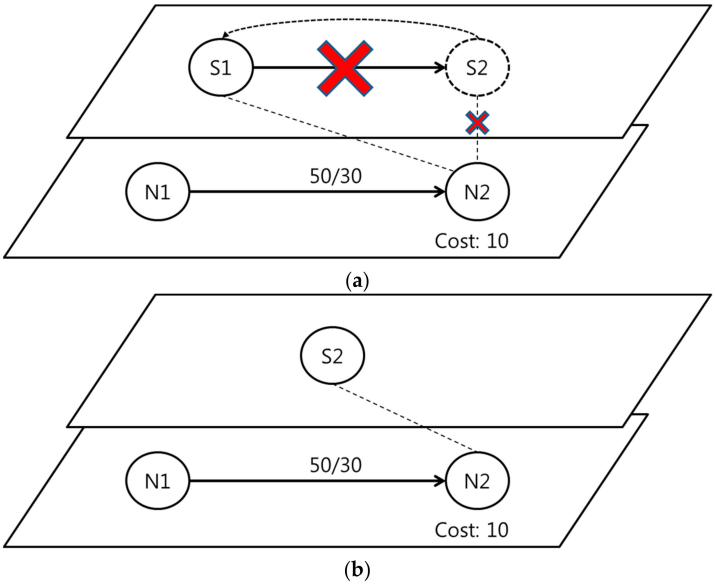

Figure 9. The example of the merging configurable method. (**a**) Unusable network usage; (**b**) Removing unusable node using merging configurable method.

Figure 10 shows the pseudocode of the reordering configurable method. This algorithm deploys virtual security devices close to the entrance of the attack flows to create efficient security networks. Actually, the location close to the attacking sources includes a number of normal packets [37]. Moreover, the location can largely reduce the damage done to the entire network system, because the ultimate goal of DDoS defense is to rapidly catch and remove anomalous flows. To be more specific about the code, the Reordering () function is called automatically when the current network needs to relocate. The Reordering () function calculates the total weight value using the weight () function. If the current weight value is higher than the preset optimal weight value α, a rearrangement process is performed. The optimal weight α value is calculated based on one hop between the node and the node according to the network topology configuration. We get α that weight bounding value multiplying the number of virtual equipment by half of the number of physical equipment links. For example, if there are four links of physical equipment and three pieces of virtual equipment, $2 \times 4 = 6$ is α. The weight value in Figure 11 becomes $3 + 4 \times 2$ and becomes 11. In this case, since the value of 6 is exceeded, it becomes a reordering target.

```
Function Reordering()
Input: attack flow: vNetwork
If vNetwork.weight (vNetwork) < α then
  vNetwork.reordering ()
end if
```

Figure 10. The pseudocode of the reordering configurable method.

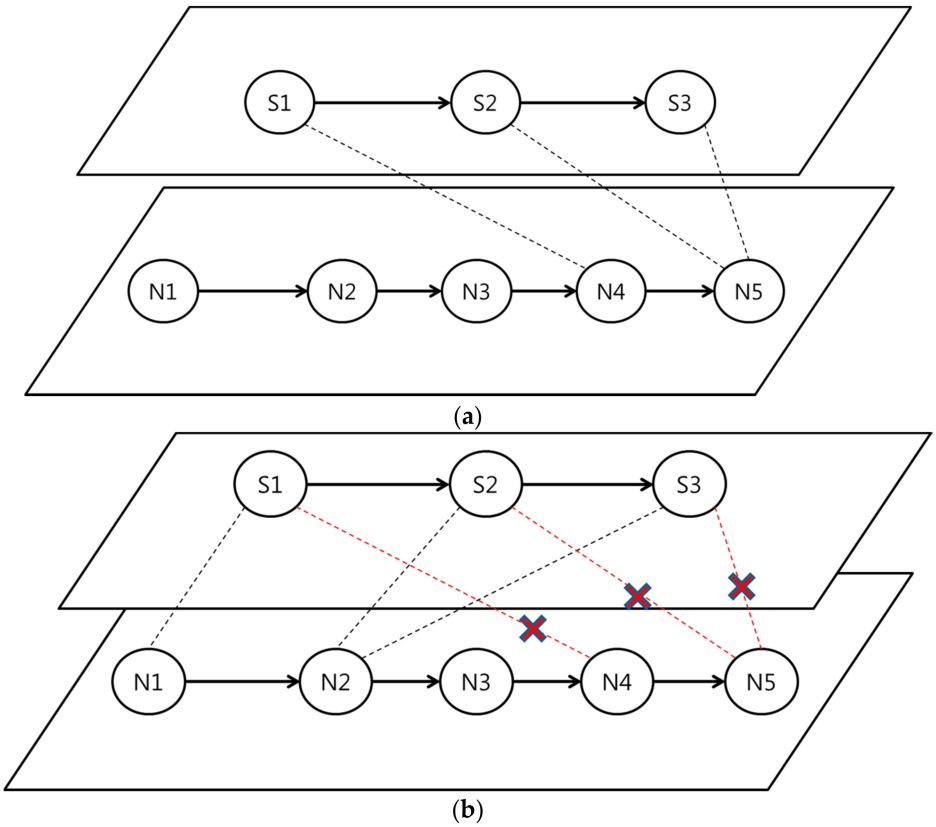

Figure 11. The example of the reordering configurable m. (**a**) Inefficient network flow path; (**b**) Rearranged network path.

Figure 11 presents a path reordering example, in which an inefficient path becomes an efficient path. As seen in Figure 11a, S1–S3 are deployed on the end of the network path. However, this is not a good way to detect attack flows because most attack flows attack the front network nodes. If the front network nodes are broken, the back nodes can become obsolete. As seen in Figure 11b, the reordering configurable method changes the back paths to front paths to increase network security.

7. Theoretical Analysis

This section describes a network environment with a physical layer that does not have security processing capability, and a virtual layer that can efficiently process attack flows using a security policy scheme. Two networks with an existing environment are analyzed. For this analysis, it is assumed that half of the data packets are flowing through the attack flow. For example, in the case of 100 data packets, the flow is 50/50, 50 normal packets and 50 attack packets.

The theoretical analysis results are presented in Figure 12. The black line shows the cost of the data packet, in which the device is placed in a simple physical layer in which no security policy is applied. The red line is added to the virtual layer, and the security policy is applied. In that scenario, the security policy is applied and 50 attack flows are processed. Then, the cost is recalculated, so that the cost to the simple physical layer can be reduced.

The graph shows that the overall physical layer results in an excessively high cost based on 0.8. Thus, the cost increases sharply based on the value 1 (Table 1). However, if a device with a security policy is deployed in the virtual layer, the cost is uniformly lower than 70 in all cases. This is because no matter how large the data size, the overall cost is reduced due to the security policy. For example, in the case of the load for the last 1.2, the value of 1.2 is 120/100, but when the virtual security policy is applied, the cost is 3, because 60/100 is 0.6.

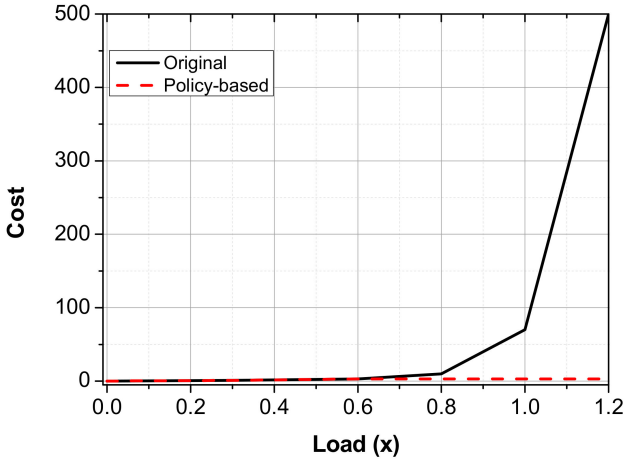

Figure 12. The relationship between the costs and loads.

8. Experimental Result through Simulation

This section compares and analyzes the simulation results to demonstrate the performance of the proposed method. For the simulation, we developed a virtual network configuration module. For the performance evaluation, we calculated the current cost by calculating the cost based on 100 data packets. The experiments were performed based on four hypothetical scenarios, and the experimental results were analyzed. The four scenarios are:

1. A network with only physical equipment that does not have any security policy;
2. A network environment where the attack flow is cleaned based on the security policy;
3. A network environment that processes attack packets through a security chain based on the number of attack flows divided by 5;
4. A comparison of a network with and without a security policy, in an environment where the normal packet/attack packet changes randomly over time.

First, Figure 13 shows the resulting graph for Scenario 1, in which the security policy is not applied. The x-axis of the graph represents the number of sensors, and the y-axis represents the cost. As shown in Figure 13, the cost increases proportionally as the number of sensors in the graph increases. This is because as the number of devices increases, the number of attack packets does not decrease; only the cumulative cost of the network device increases.

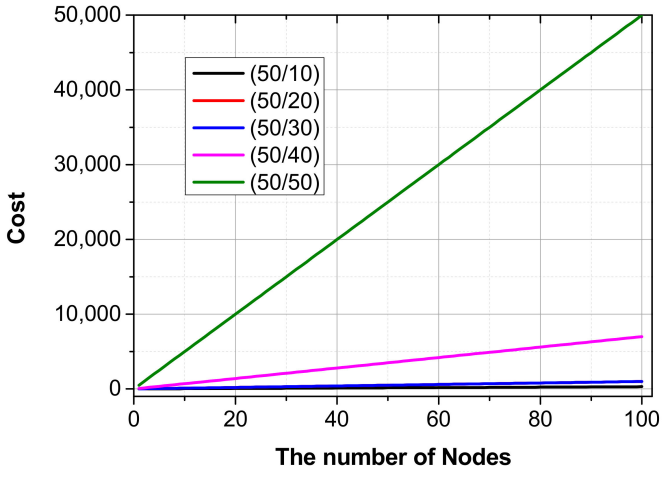

Figure 13. The cost of physical layer devices without security policy.

Figure 14 shows the resulting graph for Scenario 2. Unlike Scenario 1, the cost remains almost at 300 or less, regardless of the number of attack and normal packets, with little increase in cost. This occurs because, when a security device is encountered, the security policy processes all the attack flows.

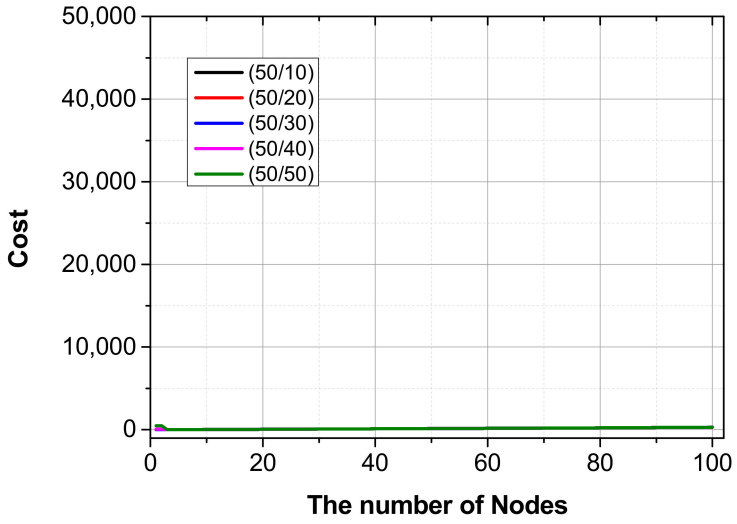

Figure 14. The cost of physical layer devices with a security policy scheme at the first node.

Figure 15 shows the results for Scenario 3. Unlike Scenario 2, a device with a security policy does not process the attack flow all at once; rather, it processes it while decreasing it by -5. The cost is not fixed as a whole because of the scenarios in which the security devices process according to the security chaining policy. Since the devices to which the security policy applies are randomly applied as attack packet/5 and (attack packet/5)/number nodes, they are shaken because they are not applicable devices.

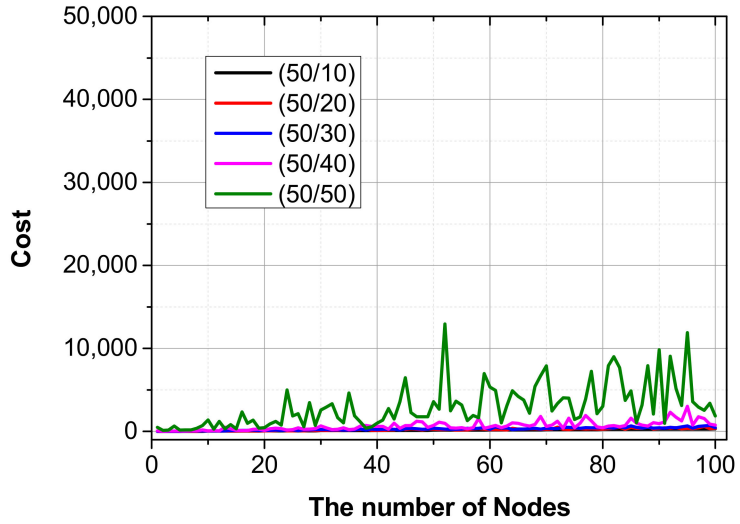

Figure 15. The cost of physical layer devices with a security policy scheme at a random device.

Finally, Figure 16 shows a graph in which the normal and attack packets are randomly changed from 0 to 50 in a network environment where 100 devices are scattered, and the cost is accumulated over time. If the security layer is not applied over time, and only the physical layer is present, this type of network environment is always worse than an environment where security is applied. However,

if the cost of the security policy is reduced in the virtual network environment, the cost is effectively reduced and decreased.

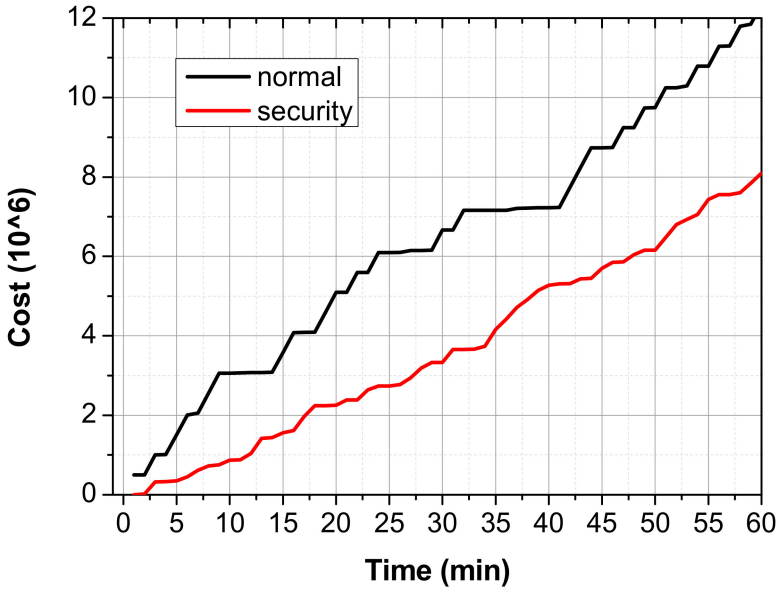

Figure 16. Normal scheme vs. policy security scheme.

9. Conclusions

In this paper, we proposed a policy-based path resetting method in an SDN environment. The specification and the mathematical analysis of the proposed numerical code and logic hierarchy were presented, and the simulation results were analyzed.

We then compared the proposed policy configurable methods with the previous system in terms of costs. The results show that the proposed policy configurable methods are better than the previous system that exists in the physical layer that cannot efficiently process attack flows. The reason is that policy configurable methods efficiently process the attack flows by reflecting the policy according to the attack types, whereas the previous system has only the fixed path, and when a new type of attack flow comes, it cannot process it due to the various attack types.

Especially, if the load is 1, the gap between the systems is high. The proposed SDN policy-based security architecture provides four policy configurable methods: separating, chaining, merging, and reordering. Security system managers can easily manage the entire security system using these methods. Moreover, the security policy configurable methods are not dependent on vendors due to independent policies because, in the proposed SDN policy-based security architecture, all vendors have the same policies.

In future work, we will consider whether the proposed SDN policy-based security architecture with the policy configurable methods can be deployed in real commercial devices to create an efficient security system.

Acknowledgments: This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2017R1C1B5016444).

Author Contributions: This paper is a joint work between Woosik Lee and Namgi Kim, from the initial design of the experiment to the analysis of the results and the preparation of the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bradai, A.; Singh, K.; Ahmed, T.; Rasheed, T. Cellular software defined networking: A framework. *IEEE Commun. Mag.* **2015**, *53*, 36–43. [CrossRef]
2. Bo, H.; Gopalakrishnan, V.; Lusheng, J.; Seungjoon, L. Network function virtualization: Challenges and opportunities for innovations. *IEEE Commun. Mag.* **2015**, *53*, 90–97. [CrossRef]
3. Lee, W.; Choi, Y.; Kim, N. Study on virtual service chain for secure software-defined networking. *Adv. Sci. Technol. Lett.* **2013**, *29*, 177–180. [CrossRef]
4. Cisco. Cisco Nexus 1000V Switch and Cisco vPath 2.5 Virtual Services Ecosystem. *White Pap.* **2014**. Available online: http://www.cisco.com/c/en/us/products/collateral/switches/nexus-7000-series-switches/white-paper-c11-730475.html (accessed on 13 June 2017).
5. Quinn, P. Service Function Chaining Creating a Service Plane Using Network Service Header. *Computer* **2014**, *47*, 38–44.
6. Jiang, Y.; Li, H.; Wei, H. An architecture of service chaining. *Internet-Draft* **2013**. Available online: https://tools.ietf.org/html/draft-jiang-service-chaining-arch-00 (accessed on 13 June 2017).
7. Wan, K.; Chang, R. Engineering of a global defense infrastructure for DDoS attacks. In Proceedings of the 10th IEEE International Conference on Networks, Singapore, Singapore, 27–30 August 2002; pp. 419–427.
8. Khatua, S.; Ghosh, A.; Mukherjee, N. Optimizing the utilization of virtual resources in Cloud environment. In Proceedings of the 2010 IEEE International Conference on Virtual Environments Human-Computer Interfaces and Measurement Systems (VECIMS), Taranto, Italy, 6–8 September 2010; pp. 82–87.
9. Alicherry, M.; Lakshman, T.V. Optimizing data access latencies in cloud systems by intelligent virtual machine placement. In Proceedings of the IEEE INFOCOM, Turin, Italy, 14–19 April 2013; pp. 647–655.
10. Fischer, A.; Botero, J.F.; Beck, M.T.; Meer, H.; Hesselbach, X. Virtual network embedding: A survey. *IEEE Commun. Surv. Tutor.* **2013**, *15*, 1888–1906. [CrossRef]
11. Chowdhury, N.M.M.K.; Rahman, M.R.; Boutaba, R. Virtual network embedding with coordinated node and link mapping. In Proceedings of the IEEE INFOCOM, Rio de Janeiro, Brazil, 19–25 April 2009; pp. 783–791.
12. Chowdhury, M.; Samuel, F.; Boutaba, R. PolyViNE: Policy-based virtual network embedding across multiple domains. *J. Internet Serv. Appl.* **2010**, *4*, 49–56. [CrossRef]
13. Jo, J.Y.; Lee, S.Y.; Kong, J.U.; Kim, J.W. A centralized network policy controller for SDN-based service overlay networking. *J. Korea Inf. Commun. Soc.* **2013**, *38*, 266–278. [CrossRef]
14. Sazena, M.; Kumar, R. A recent trends in software defined networking (SDN) security. In Proceedings of the Computing for Sustainable Global Development, New Delhi, India, 16–18 March 2016; pp. 16–18.
15. Satasiya, D.; Raviya, R.; Kumar, H. Enhanced SDN security using firewall in a distributed scenario. In Proceedings of the Advanced Communication Control and Computing Technologies, Ramanathapuram, India, 25–27 May 2016; pp. 25–27.
16. Shin, S.; Xu, L.; Hong, S.; Gu, G. Enhancing network security through software defined networking (SDN). In Proceedings of the International Conference on Computer Communication and Networks, Waikoloa, HI, USA, 1–4 August 2016; pp. 1–4.
17. Husssein, A.; Elhajj, I.H.; Chehab, A.; Kayssi, A. SDN security plane: An architecture for resilient security services. In Proceedings of the IEEE International Conference on Cloud Engineering Workshop, Luxembourg, Luxembourg, 12–15 December 2016; pp. 4–8.
18. Cox, J.H.; Clark, R.J.; Owen, H.L. Leveraging SDN for ARP security. In Proceedings of the SoutheastCon, Norfolk, VA, USA, 30 March–3 April 2016; pp. 1–8.
19. Ranjbar, A.; Komu, M.; Salmela, P.; Aura, T. An SDN-based approach to enhance the end-to-end security: SSL/TLS case study. In Proceedings of the IEEE/IFIP Network Operations and Management Symposium, Istanbul, Turkey, 25–29 April 2016; pp. 25–29.
20. Yao, L.; Dong, P.; Zheng, T.; Zhang, H.; Du, X.; Guizani, M. Network security analyzing and modeling based on petri net and attack tree for SDN. In Proceedings of the International Conference on Computing Networking and Communications, Kauai, HI, USA, 15–18 February 2016; pp. 15–18.
21. Pisharody, S.; Chowdhary, A.; Huang, D. Security policy checking in distributed SDN based clouds. In Proceedings of the IEEE Communications and Network Security, Philadelphia, PA, USA, 17–19 October 2016; pp. 17–19.

22. Martins, J.S.B.; Cmpos, M.B. A security architecture proposal for detection and response to threats in SDN networks. In Proceedings of the IEEE ANDESCON, Arequipa, Peru, 19–21 October 2016; pp. 19–21.
23. Patel, P.; Tiwari, V.; Abhishek, M.K. SDN and NFV integration in openstack cloud to improve network services and security. In Proceedings of the International Conference on Advanced Communication Control and Computing Technologies, Ramanathapuram, India, 25–27 May 2016; pp. 25–27.
24. Rupasinghe, P.L.; Kulatunga, K.M.D.S.B.; Murry, L.; Keseva, K. SDN based security solution for legislative email communications: Safe guarding communication. In Proceedings of the International Conference on Computing, Communication and Automation, Greater Noida, India, 29–30 April 2016; pp. 29–30.
25. Ammar, M.; Rizk, M.; Hamid, A.A.; Seoud, A.K.A. A framework for security enhancement in SDN-based datacenters. In Proceedings of the IFIP International Conference on New Technologies Mobility and Security, Larnaca, Cyprus, 21–23 November 2016; pp. 21–23.
26. Liu, Y.; Guo, Z.; Shou, G.; Hu, Y. To achieve a security service chain by integration of NFV and SDN. In Proceedings of the International Conference on Instrumentation and Measurement, Computer, Communication and Control, Harbin, China, 21–23 July 2016; pp. 21–23.
27. Wilczewski, D. Security considerations for equipment controllers and SDN. In Proceedings of the IEEE International Telecommunications Energy Conference, Austin, TX, USA, 23–27 October 2016; pp. 23–27.
28. Bull, P.; Austin, R.; Popov, E.; Sharma, M.; Watson, R. Flow based security for IoT Devices using an SDN gateway. In Proceedings of the IEEE 4th International Conference on Future Internet of Things and Cloud, Vienna, Austria, 22–24 August 2016; pp. 22–24.
29. Gonzaiez, C.; Charfadine, S.M.; Flauzac, O.; Nolot, F. SDN-based security framework for the IoT in distributed grid. In Proceedings of the International Multidisciplinary Conference on Computer and Energy Science, Split, Croatia, 13–15 July 2016; pp. 13–15.
30. Huq, S. Hardening the SDN optical transport network security—Is it a pleonasm or oxymoron? In Proceedings of the Optical Fiber Communications Conference and Exhibition, Anaheim, CA, USA, 20–24 March 2016; pp. 20–24.
31. Basit, A.; Ahmed, N. Path diversity for inter-domain routing security. In Proceedings of the International Bhurban Conference on Applied Sciences and Technology, Islamabad, Pakistan, 10–14 January 2017; pp. 10–14.
32. Rawat, D.B.; Reddy, S.R. Software defined networking architecture, security and energy efficiency: A survey. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 325–346. [CrossRef]
33. Dacier, M.C.; Konig, H.; Cwalinski, R.; Kargi, F.; Dietrich, S. Security challenges and opportunities of software-defined networking. *IEEE Secur. Priv.* **2017**, *15*, 96–100. [CrossRef]
34. Li, C.; Qin, Z.; Novak, E.; Li, Q. Securing SDN infrastructure of IoT-Fog network from MitM Attacks. *IEEE Internet Things J.* **2017**. Available online: http://ieeexplore.ieee.org/document/7883928/ (accessed on 13 June 2017).
35. Koo, S.K. Cyber security in South Korea: The threat within. *The Diplomat* **2013**. Available online: http://thediplomat.com/2013/08/cyber-security-in-south-korea-the-threat-within/ (accessed on 13 June 2017).
36. Fortz, B.; Thorup, M. Internet traffic engineering by optimizing OSPF weights. *INFOCOM* **2000**, *2*, 519–528. [CrossRef]
37. Zargar, S.T.; Joshi, J.; Tipper, D. A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks. *IEEE Commun. Surv. Tutor.* **2013**, *15*, 2046–2069. [CrossRef]