sustainability

MDPI

Article

FS-OpenSecurity: A Taxonomic Modeling of Security Threats in SDN for Future Sustainable Computing

Yunsick Sung [1], Pradip Kumar Sharma [2], Erik Miranda Lopez [2] and Jong Hyuk Park [2,*]

[1] Faculty of Computer Engineering, Keimyung University, Daegu 42601, Korea; yunsick@kmu.ac.kr
[2] Department of Computer Science and Engineering, Seoul National University of Science and Technology, Seoul 139-743, Korea; pradip@seoultech.ac.kr (P.K.S.); erik.miranda@seoultech.ac.kr (E.M.L.)
* Correspondence: jhpark1@seoultech.ac.kr; Tel.: +82-2-970-6702

Academic Editor: Marc A. Rosen
Received: 7 June 2016; Accepted: 5 September 2016; Published: 9 September 2016

Abstract: Software Defined Networking (SDN) has brought many changes in terms of the interaction processes between systems and humans. It has become the key enabler of software defined architecture, which allows enterprises to build a highly agile Information Technology (IT) infrastructure. For Future Sustainability Computing (FSC), SDN needs to deliver on many information technology commitments—more automation, simplified design, increased agility, policy-based management, and network management bond to more liberal IT workflow systems. To address the sustainability problems, SDN needs to provide greater collaboration and tighter integration with networks, servers, and security teams that will have an impact on how enterprises design, plan, deploy and manage networks. In this paper, we propose FS-OpenSecurity, which is a new and pragmatic security architecture model. It consists of two novel methodologies, Software Defined Orchestrator (SDO) and SQUEAK, which offer a robust and secure architecture. The secure architecture is required for protection from diverse threats. Usually, security administrators need to handle each threat individually. However, handling threats automatically by adapting to the threat landscape is a critical demand. Therefore, the architecture must handle defensive processes automatically that are collaboratively based on intelligent external and internal information.

Keywords: FS-OpenSecurity; SDN architecture; security model

1. Introduction

Future Sustainability Computing (FSC) is a principle that embraces a range of procedures, policies, programs and approaches that encompass all uses of information technology for an abundant life. The main idea is to address the sustainability problems in different information processing and computing environments.

Software Defined Networking (SDN) is an advanced approach to management and networking with the most centralized intelligence in an SDN controller component. The SDN proficiently centralizes the handling of the network by splitting the control module between off-device assets. It also has the potential to modernize legacy data centers by rendering a flexible way to control networking. Therefore, it is possible to realize the envisioned versions of storage and computing today. FSC includes reliable and efficient processes for delivering IT services. It is about how to deal with performance and doing what is essential to keep the service running efficiently, which includes keeping current versions, system recovery planning, and constant security. Consecutive generations of computing technology have enhanced network reliability, cost effectiveness, and speed, but they have not kept pace with human expectations for a change in the inbuilt shape of the network. SDN is about realizing many changes regarding interaction processes, and between systems and humans, easily adapting new techniques, and increasing the speed of network configuring. Although IT environments quickly

change, security threats are becoming more diverse and complex. The rapid improvement of this development is exponentially creating new types of attacks, such as the aggregation of identified and unidentified threats, increasing vulnerabilities on "Zero-day" and the use of unknown documents, websites, networks, and hosts with malware inside. Today, the world is highly dependent on infrastructures and networks. The boundaries are not clearly defined and threats are becoming smarter. It is necessary to identify the most appropriate approaches for protecting the network architecture in a landscape of evolving threats. Although there are many security products available, they are reactive in nature instead of possessing the strategic quality of oriented architecture. Recently, companies require a single architecture that brings together high-security devices on a high performance network performance with proactive enforcement protections.

The reality of the SDN-enabled host is rapidly being produced and developed. The programmability and functionality of classified data and control planes in the network have long been the topic of research. It has also been commercially applied to virtualization and cloud computing approaches. SDN has advantages in several scenarios, including data centers and enterprises, and they have already been applied throughout various backbone networks. For example, Google is already using SDN networks, called Google B4 [1]. Nevertheless, arguments against SDN exist for a full-scale carrier network. Some of these challenges have been shown in [2]. Security in SDN is one of the critical areas vying for attention.

The SDN controller lies at the core of the SDN architecture, which is logically placed between SDN applications and network elements to provide an interface among them. Its centralized location enables a global overview of what is happening in the network for other SDN components, and it configures in real-time to find the best path for traffic. Unlike conventional networks where control is distributed, the SDN controller is centralized. However, the centralized location of the controller becomes the main surface for attack.

SDN architecture is implemented to improve the network with the provision of security administration, analysis, and a highly responsive feedback system. To produce security-related data, anomaly detection and traffic analysis methodologies need to be installed in the network. This safety-related information is transferred to the central controller. Applications are run on the controller to examine and relate the input from the whole network. Based on the investigation from the security-related data, new or updated security policies in the form of flow rules are put in place throughout the network. This methodology effectively improves the control of the network and provides protection against security threats.

Nevertheless, the same properties of programmability and centralized control associated with SDN architecture raise the issue of network security approaches. Due to the centralization of the controller and flow-table restriction in network devices, SDN is susceptible to Denial-of-Service (DoS) or Distributed-DoS (DDoS) attacks. An additional issue caused by the network-based open programmability is trustworthiness, both among controllers and network devices, and controllers and applications.

Some solutions have been proposed in the literature to address these SDN security challenges, ranging from imitation controller through conflict resolution of policy to validation mechanisms. Analysis and a new architecture model for SDN security challenges are introduced in this paper. Depending on the SDN layer affected or targeted, the security issues are categorized on a case-by-case basis.

The remainder of the paper is organized as follows. In Section 2, security threats, attacks, requirements for SDN and previous research works are introduced. Section 3 presents a new proposed FS-OpenSecurity model for SDN as well as the analysis and discussion. Finally, we conclude our research in Section 4.

2. Related Works

The main idea of using an SDN architecture is to provide security at the enterprise level. This section highlights the enhancements in security and describes current work using SDN architecture.

2.1. Security Threats in SDN

In this section, we analyze the category of threats that the SDN network is exposed to. Threats are classified into three categories: based on SDN main functional components; the nature and type of attacks; and based on the resource types. An example is attacks that are focused on forwarding flow tables of devices where the flow tables contain information associated with switching, routing, network management, and accessing. SDN can focus on the control logic because it is the crucial component for control and management. The channel among the forwarding devices and the controller is another real target when it carries crucial information that is captured. The controller communicates using a standard interface such as REST with high-level applications at the top level. Such an interface can also be attacked to permit malicious logics to connect and communicate with the network, the controller, and its activity.

Spoofing: This indicates the method where system resources, such as ARP, MAC, and IP, are faked purposely to hide the source of the attack. For example, to gain the right to use network resources, clients can make use of spoofed IP addresses. To initiate DDoS attacks, spoofed addresses are also part of a zombie or a botnet network. At present, spoofing threats mainly incorporate IP spoofing and ARP in SDN. ARP Spoofing includes connecting to a valid IP address via an attacker MAC address. The attack by ARP spoofing can lead traffic to be diverted from the intended receiver and, as an outcome, an authentic host or user is removed from the system. IP mapping tables for MAC are used to detect ARP spoofing. IP Spoofing is utilized as an opening to different sorts of security attacks, for example, Domain Name System (DNS) amplification or tampering. A DNS is an index that links space names to IP address. Attackers may misrepresent the DNS index to redirect activity to unauthenticated sites. What all spoofing techniques have in common is that they attempt to divert activity to unauthenticated sites. Man in the Middle (MIM) attacks are also considered. An appropriate authentication scheme mitigates spoofing.

Tampering: This refers to the intentional and unauthorized destruction or modification of network data, for example, topology, access lists, flow tables, and policies. An intruder may attempt to change the rules of the flow. To allow illegitimate or deny legitimate hosts, they may change the firewall rules or flow tables. To cause some traffic to be diverted, intruders can also attempt to alter topology. Different controllers can intercommunicate vital information with each other in an SDN controller distribution. It is essential to protect the channel medium from being tampered with or hijacked.

Repudiation: Non-repudiation is conceived as a lawful instead of a technical concept. It attempts to ensure that reputation does not happen. Reputation is the nullification in the communication by one of the entities required in a communication that is has previously taken part in, either partially or fully. The sender must confirm that the packets sent to the genuine receiver enclosed in the packet header and the receiver requests to validate those packets are sent by the genuine sender enclosed in the packet header. Non-repudiation is often associated with responsibility, i.e., keeping the persons or entities responsible for their actions.

Information Disclosure: Data uncovering attacks have no immediate reason to disturb or destroy the network but to keep an eye on its data. Apart from this, they will at first attempt to sniff system data, for example, the characteristics of nodes, topology, or the details of the communication between nodes. The impact of these attacks on SDN architecture are integrated. To control all network switches, the controller is in a focal area. The attacker has an enormous network access to invade the controller. Conversely, information is detached from the controller, in contrast with conventional switches where the information is situated within the switch the control. In SDN, the flow tables of the device contain flow rules. If intruders are able to access and alter flow rules in the switches, they can redirect the traffic to go to wrong destinations. If they manage to detach a forwarding device to communicate

with the controller, they take for granted control due to significant miss-management of traffic. If they could take over traffic from a genuine host, they could imitate that host and connect to the network as a spy. MIM attack is an attack involving disclosure of information, communication and transit, TLS is optional. Additionally, in the northbound interface, communication with the controller is not yet matured.

Denial of Service: The most grievous threats that affect network performance are: drop of legitimate packets and increase latency, which may prevent the network from working or halt the entire system. As there is a nonstop flow among the switches and controller, DOS is more destructive to OpenFlow networks. The uninterrupted communication among the switches and controller lures the attackers to disrupt the normal network activity by pushing the flows between switches and the controller. DNS amplification and flooding are considered as stream level data attacks since stream level data is sufficient for identifying such attacks. To detect most types of DoS attacks, information about the flow level is generally sufficient. Usually, at the flow level, flow header traffic data are gathered. Methods of detecting a flow attack that rely only on header information network face the following threats: scans, DoS, botnets, and worms. These four sorts of attacks have a several familiar signs. Significant traffic usually comes in. However, in the event that the local machine is a victim or a botnet, it sends a huge traffic in bulk. Port numbers are also vital data in these sorts of attacks where there are identified ports to be widely utilized. Different sorts of system attacks may require packet level data to be considered. Data taken from flow headers are useful for detecting DoS.

Elevation of Privilege: To access applications and assets that require special permissions, the hackers try to elevate after entering the system. An intelligent and robust process is required to identify the privilege elevation attacks. The biggest problem with auditing or logging approaches is scalability since they store an enormous amount of information, which may influence memory, bandwidth, and storage. Since privileges are set into access controls or authorization modules, attacks frequently target those modules and attempt to modify data in those modules.

2.2. Security Requirements for SDN

By giving a general clarification of threats to vital areas of SDN, as shown in Figure 1, the requirements for novel responses to impendent threats to the network are seen. The vector of the threat to these areas is: the false/falsified flow traffic, attacks on the control plane communication interfaces, attacks on vulnerabilities forwarding devices, trust among applications and the controller, attacks on the vulnerable controllers, trust assets for remediation and forensics, and vulnerabilities in management.

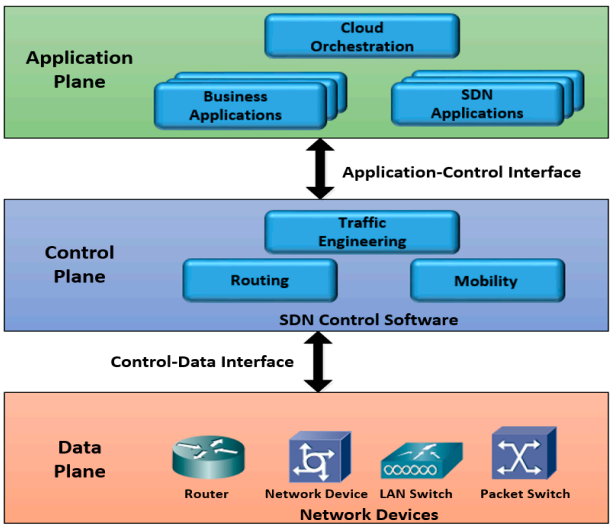

Figure 1. SDN architecture illustrating the application, control and data layers and interfaces.

The fundamental attributes of a secure communications network are authentication, non-repudiation, confidentiality, availability of information, and integrity [3], which we discuss in detail below.

Confidentiality: Confidentiality means avoiding delicate data like flow table or device data being revealed to unauthorized parties at the switch level. Attackers may attempt a variety of techniques such as scanning, hijacking, and spoofing to gather interesting information. For controller related systems, confidentiality means to avoid delicate data such as policies, rules, and controller data being disclosed to unauthorized parties. For the linked channel, confidentiality means ways to allow for safe communication among the switches as well as controllers, and protect against the disclosure of delicate data to unapproved parties.

Integrity: Integrity refers to the defense of the information elements to be altered by unapproved third parties, where components here consist of hosts, flow table and switch at the switch level. The open-flow permits a switch to upgrade its flow table by removing and adding flow policies. However, the control strategy is typically not a specific implementation and is improved in this system, which can open a gap for the attackers. Regarding the controller, integrity attempts to defend the controller's data from being altered by controller's policy, or being harmfully modified by unapproved parties. For the channel related networks, integrity refers to making sure that the information transmitted is accurate, steady, cannot be repudiated, and is trustworthy.

Availability: For a data plane at the switch level, availability refers to guaranteeing that approved parties are capable of accessing the forwarding device as well as associated data when desired. In an OpenFlow switch, the flow tables are a vital component that holds flow policies to identify the activities that take place. Invaders can trade off it with a variety of intrusions like DoS attacks. For the control related case, availability refers to making sure that approved parties are capable of accessing the controller for demanded data when desired. For the channel level, availability refers to making sure that authorized parties like switches and controllers are capable of accessing each other when desired.

To provide a protected network from unintentional damage or malicious attack, security professionals have to secure the network resources, the data like the communication and devices links across the network. To make sure that network security is constant, the modifications to the network architecture brought in by SDN must be evaluated. Casado et al. [4] exclusively measured the security parts of a different forwarding and control model. In 2006, SANE architecture was proposed, which focused on a centralized controller that is accountable for policy enforcement and authentication of hosts. The authors thought it would bring drastic changes to the networking architecture and end-hosts; however, SANE is not mature enough for most enterprises. Ethane [5] used a technique that required modification to the initial system. It controlled the network using two components: ethane switches, which are based on rules in a flow table forward packets, and a centralized controller accountable for applying global policy. This change in system control permitted the control as well as data plane to be divided to take into account more flexible programming.

The particular security issues with SDN from the point of view of the SDN skeleton are shown in Figure 1. The difficulties associated with each layer of the system such as data, control and application planes and the interfaces between these layers are recognized. Recently, some security analysis has been carried out. These analyses have determined that the modified components or relationship among these components in the SDN system expose new weaknesses that were not there earlier. Kloeti [6] the researchers carried out an investigation of the OpenFlow protocol with the STRIDE threat analysis technique [7]. Here we focus on the execution of data leakage, information disclosure, DDoS and traffic hijacking or re-routing attacks. The author acknowledged it was possible to run successfully, although some mitigation methods are provided.

The OpenFlow switch specification [8] depicts the transport layer security (TLS) with reciprocal validation among the switches and the controllers. Still, the security aspect is non-compulsory, and TLS standard is not specified. The deficiency of TLS adoption by the vast majority of the sellers and the

opportunity of DDoS attacks are the center point of an OpenFlow weakness evaluation [9]. The authors identified that, due to the absence of TLS use, it could prompt standard alteration and fraudulent principle insertion.

Kreutz et al. [10] presented an analysis of the overall high level of security of the SDN. Because of the environment of the programmability and centralized controller of the network, they concluded that novel threats need new responses. They proposed some approaches to tackle the diverse kinds of threats, including diversity, protected components, and replication. Li et al. [11], the investigation system and ProtoGENI testbed was likewise considered. It is observed that when using the ProtoGENI network, flooding attacks to the broader Internet and several attacks with malicious propagation among users of the testbed were possible.

The outcome of these analyses shows a variety of security issues related to the SDN framework. An association is absorbed between the category of publishing/attack security and SDN/interface layer struck by the issue of security/attack. The layers of data and control are clearly recognized as attack targets, and the analysis focuses on security related issues associated with the data, control, and management layers.

2.3. Previous Research on SDN

The architecture of an SDN demonstrates the novel modernization in network utilization. This consolidates the view of the large network and programmability information gathering process from remaining Intrusion Prevention Systems (IPS) and Intrusion Detection Systems (IDS), such as that compiled by the analysis and programming of the central control of the network. This technique makes the SDN stronger against malicious attacks compared to traditional networks.

To offer network security features, conventional networks use devices (middle-box). Presently, there is a debate regarding the consolidation of security devices operating in SDN and the advantage of programmability for forwarding network traffic to devices. The Slick architecture [12] proposed a centralized controller. The centralized controller is accountable for installing and migrating features onto custom middle-boxes. To route specific flows taking into account security needs, applications such as Slick then guide the controller to install the essential functions.

Fayazbakhsh et al. [13], the FlowTags architecture proposed the use of modestly customized middle-boxes, which cooperate with an SDN controller through a FlowTags API. FlowTags, comprising of traffic flow information, are implanted in packet headers to give flow tracking and facilitate controlled routing of tagged packets. The disadvantage of this approach is that it only works with proactive policies and not reactive. In contrast to [12,13], the SIMPLE policy enforcement layer [14] uses SDN to control middlebox deployments which needs no modifications for SDN functionality. This approach is appropriate for legacy systems. According to these approaches, a simple method for providing network security would be to set up a medium box or a suitable controller and program the network to redirect traffic designated by the devices. However, the variety of attacks threatening the network cannot be completely handled by the middle-box. As such, to make network security solutions, in particular, exploit the SDN architecture, a series of solutions have been proposed beyond the middle of the boxes.

To discover vulnerable targets in the network, attackers use different filtering techniques. Protection has been shown to prevent these attacks using random addresses such as virtual Internet Protocol (IP) using SDN [15]. This methodology utilizes the OpenFlow controller to manage a pool of virtual IP addresses and is decided to be inside the network, erasing the genuine IP addresses from the outside world. It uses a target of protection that is a type of adaptive cyber security. It is required to utilize Monitoring Systems to defend the network from attack. The authors in [16] presented a DDoS detection technique that employs numerous traffic flow features and monitors NOX switches at regular time interval and uses self-organizing maps to recognize abnormal flows. The OpenSAFE [17] used its ALARMS policy language to deal with the routing of traffic throughout network monitoring devices. Shin and Gu in [18] presented a very similar design focused on SDN in the cloud. In [19],

they proposed using SDN to provide intrusion detection in a small office/home. The opportunity for the improvement and simplification of system security using the SDN architecture is manifested in this body of research. Similar approaches are commercially available for a variety of SDN security products at different stages of development.

2.4. Security Attacks in SDN

While security as improved SDN architecture was recognized, solutions to deal with the challenges of securing the SDN are fewer. To address the security issues and ensure the correct functioning of an SDN, three key network properties are needed: management plane, data plane forwarding, and network topology must always be preserved. Here, the following scenarios are shown: delegated possible attacks on the network topology and data forwarding plane plunged compromised hosts and switches. The mechanisms of poisoning the opinion of the network controller remains essentially the same. In addition, to shield from a known security threat, as we discussed, we will not be able to transfer the conventional solution to SDNs directly. Some adaptations of these solutions try to employ defense mechanisms considering known or particular vulnerabilities, like selective signature-based approaches; however, they suffer from the same limitations as anti-virus solutions as they are not effective against unknown attacks.

Attacks by Host and Switch: If the packets do not match a flow rule, OpenFlow mandates that packets must be sent to the controller by the switch. This agreement opens new opportunities for malicious hosts corrupting the transfer of the data plane and the network topology, both of which are essential to the correct portion of the SDN. Notably, malicious hosts first build the data packet. The built packet data as PACKET_IN messages would then be communicated by the switches, and afterward processed by the controller. Secondly, DoS/DDoS attacks on the switches and the controller, and thirdly, they empower side-channel techniques to gather information about rules of the flow. Matured switches do not just mitigate all the host-based attacks, they additionally generate dynamic attacks on traffic flows pass through the switch, causing traffic hijacking or re-routing and network DoS/DDoS.

Network Topology: Various packets of the protocol like LLDP, IGMP, and ARP sent by switches is monitored by SDN controllers as OpenFlow PACKET_IN messages to build its view of the topology of the network. LLDP messages are handled by controllers to discover the topology and IGMP messages to support multicast groups; then it transmits the ARP requests and responses for end hosts to build ARP caches, facilitating a communication network. Matured hosts spoof the above messages to corrupt the controller topology and instal flow rules to launch a variety of attacks on the network.

Data Plane Forwarding: By flooding the traffic in the network to random hosts, malicious switches and hosts launch DoS/DDoS by exhausting the resources of the switches and the controller completely.

Traditional Attacks Manifest in SDNs: Many attacks that affect conventional networks also strike SDNs. Nevertheless, the adaptation of conventional protections for such attacks in SDN is not insignificant. This is due to conventional intelligence frequently trusting switch knowledge to implement safeguards against identified security attacks. However, SDN switches are straightforward sending entities with no insight.

Although it is possible to patch the SDN against specific known vulnerabilities, it is not a complete solution for all attacks, as described below.

Rosemary: Although the control plane works as the critical middleware facilitator between the network applications and data plane, even tiny and basic faults in network applications lead to the breakdown of the control plane and failure of network functionality. Rosemary [20] employs numerous approaches like sandboxing, application containment, and resource management to build a high-performance, secure, and robust network OS.

AVANT-GUARD: An openFlow network is vulnerable to a saturation attack control plane that operates the chokepoint that emerges between the control plane and the data plane. In addition, OpenFlow provides partial support for the effective monitoring of the network, which is an essential feature for the implementation of various security services. AVANT-GUARD [21] performs data

plane extensions that enable the managing of vigilant and scalable switching flow to facilitate the development of services that are more flexible and, eventually leading to scalable security SDN.

FRECSO: Developers face numerous troubles to create security application on SDN because it needs a complex rationale to be executed. For developers who want to design network security applications, FRESCO [22] attempts to provide an effective program framework, that can easily and efficiently implement a variety of security detection modules and mitigation.

SE-Floodlight: This is an improved security version of Floodlight [23]. Its main important components include flowing rule conflict resolution, an authorization model for the data plane access, role-based authorization, authentication, and more. It gives a security enforcement kernel to the data and control plane to provide the characteristics listed previously.

Security-Mode ONOS: The primary objective is to offer a safe application SDN achieving environment for Open Network Operating System (ONOS). In systems controlled by ONOS, it is conceivable to convey different ONOS applications to empower an assortment of system control capacities by utilizing the effective APIs offered by ONOS stage. Meanwhile, the ONOS applications with such a dominant authority can also be misused and become the root of safety problems. To remove these opportunities to misuse or abuse the system, the authors of [24] impose security policies to restrict ONOS applications.

3. FS-OpenSecurity Model for SDN

To address the security problems as we discussed in the previous chapter, we introduce a FS-OpenSecurity architecture for SDN. FS-OpenSecurity is a lightweight, efficient, scalable and protocol-independent defense framework for SDN networks to prevent all three planes saturation attack. In this chapter, we will present the detailed design of FS-OpenSecurity.

3.1. Infastruecture Overview

We have divided the security attacks into two categories: northbound and southbound. To handle these attacks we have proposed FS-OpenSecurity, a taxonomic modeling of security threats in SDN. In the OpenSecurity model, we have proposed a two-layer model: software defined orchestrator (SDO) and SQUEAK detect and prevent northbound and southbound attacks as shown in Figure 2. SDO and SQUEAK will handle security attacks at a different level. SDO mainly works at the application or management layer, application-controller interface, and control layer. SQUEAK will work at the data layer, data-controller interface, and control layer. This architecture must shield organizations of every size at any location like branch offices and headquarters as well as mobile devices using roaming or cloud environments. By adding the high-performance, fast-evolving and dynamic SDO and SQUEAK based layer models, the FS-OpenSecurity architecture provides not only operational flexibility but also delivers proactive and reactive incident prevention for the ever changing threat landscape. The protections should automatically adapt to threats without the need for security administrators or policy makers to follow thousands of recommendations and opinions manually. The protections should be used seamlessly in a large IT environment and architecture must provide a defensive force that operates in collaboration with intelligent internal and external sources.

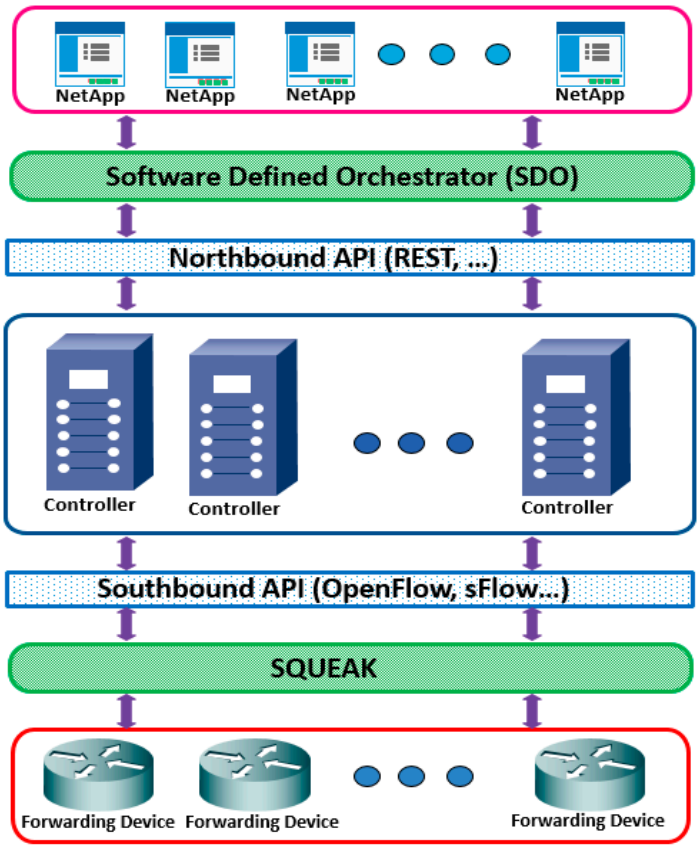

Figure 2. FS-OpenSecurity architecture.

3.2. Software Defined Orchestrator (SDO)

The main responsibility of SDO is to provide the protection software defined and allow for execution to the application of the appropriate application layer. SDO might be implemented with high performance and used as software based on the host or used as dedicated hardware in the network, in the cloud or on mobile devices. Protection categories include data protection, access control, and threat prevention as shown in Figure 3. These plans vary in the field of basic knowledge of the security policy system.

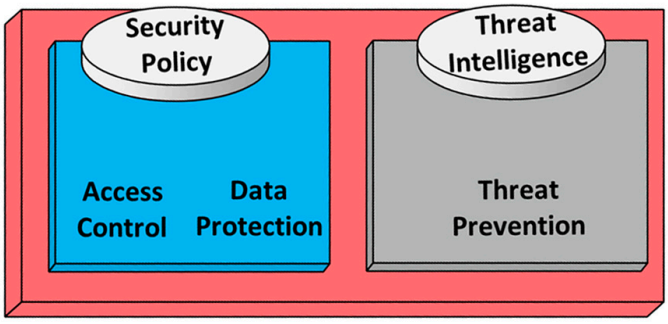

Figure 3. Software defined orchestrator.

Data Protection: Focuses on the categorization of data rather than the interactions and behaviors. Layer management decides the data flow policies in the system.

Access Control: A security protocol model of authorized communications is imposed among assets and users by the Management Layer.

Threat Prevention: Understanding of threat and threat activities are described. It is similar to a cop that acts on collaborative threat intelligence in real time obtained from the community.

SDO provides the desired level of flexibility to manage various types of threats and changes in network configurations. The application layer provides a dynamic system running protection measured at points of application across the enterprise. Since the software manages the protection, essential equipment does not need to be exchanged in these reinforcements as new attacks and threats are exposed, or when new technologies are developed within the organization. Protections should be automatically added to the landscape without requiring manual threats followed by thousands of recommendations and opinions. All this is done without human intervention. However, the human decision making may be necessary when the threat pointer offers low assurance as to the identification of an attack or threat.

Data Protection: Protections must follow data, in motion and in storage. The cryptographic control model is applied to deny access to unauthorized users and to defend data outside and within the organization. Data flows are scrutinized to identify and prevent data loss by classifying the data through enterprise information categorizations considering its attributes, ownership, and content depending on the security strategy for data watermarking and categorization. Data signatures are generated based on the sensitivity of the data and are used to prevent unauthorized data leakage from exploiting any place or host. Cryptographic techniques, digital signatures, and data encryption should be applied to the data in storage to prevent unauthorized changes and access. These techniques provide continuous protection, even when the data is copied on the outer surface of the controlled system. Encryption is particularly important for storage on removable media, mobile devices, cloud computing and shared storage environments. Cloud based or local key management architecture are essential to managing keys and efficiently accessing the encrypted data. Encryption is also used to ensure that data is safely disposed of through the key revocation.

Access Control: Access control has generally been central to the implementation of trade security policy and is still today the basis of any security architecture. Access facilitates business process control with essential interactions given the data and users within the commercial network. It applies the lowest level necessary to meet the standard of least privileged security. Interactions that are not clearly authorized are regarded as unauthorized and should be blocked. Access control defenses depend on custodians that describe the specific policy to business enterprise, roles, users, assets and applications and security policies are classified for all communications between the same authorized users, assets, and applications. For example, access control is used to decide whether a user is authorized to access sensitive corporate services of authorized hosts, locations, time of day or other approved attributes. These protection controls are divided into incoming and outgoing control assemblies. Inbound, all segments are expected to defend their assets against external attacks. Strict application of the Lease Privilege standard reduces the attack surface privileges. For example, an application within the segment might have a security breach, but since the access control rule denies access to the application, the vulnerability cannot be exploited. The Least Privilege standard also determines that customers within the protected segment should be allowed to simply access external services that directly or indirectly hold the company. Outgoing checks are essential to meet that standard. Monitoring and traffic analysis are adaptively handled in context. For example, in the case of Internet influx, the control layer may check with a cloud database for the latest protocols and authorized applications; while in the case of internal inflow, it may authorize the use of appropriate behavior of the protocol or application used by the organization. Further, the control layer is aware of network changes and definitions used in other computer systems. Examples could be the depositing of user changes by automatically applying security to a new virtual machine or allowing access to a new host classified in a Domain Name Server. In SDN, the control layer also redirects network traffic flows through appropriate reinforcements, in this way shaping the network to comply with the company's segmentation model and Security Policy.

Threat Prevention: Preventing the backup of threats blocks attackers and denies the abuse of vulnerabilities and delivery of malicious payloads. Deterrence threat strategy is simple: "All threats must be banned". This policy requires little customization for each company, but is a little generic and should be applied to all companies. Prevention backup threats are separated into two sections: pre-infection and post-infection. Pre-infection protection provides proactive detection and prevention of threats that attempt to exploit vulnerabilities in protocols and internal applications or try to deny service to authorized applications. Post-infection protection offers agile defenses that notice, deactivate and contain the threats once they have successfully corrupted one or more network entities. In some cases, a particular security decision provides low confidence in the continuation of a threat. The prevention module of the threat of the control layer compares the results of many engines—behavior, signatures, emulation malware, reputation and human validation—to achieve a higher level of confidence. Further, the control layer uses foreign assets to generate significant defense security. For threat prevention controls to be effective, they need a reliable and wide threat intelligence. Organizations should anticipate a steady stream of information about threats to transfer to their security environment without manual intervention being required.

Threat Intelligence is achieved using internal and external sources of threat data as shown in Figure 4. Preferably, these sources should consist of public security information, such as security Computer Security Incident Response Teams (CSIRTs) and Computer Emergency Readiness Teams (CERTs), different security product vendors, security analysts and other organizations within the security community. In addition, threat intelligence is generated within the company through sandboxing techniques, data analysis and research malware security events collected from reinforcements. Threat Intelligence shows the threat agents, their intended targets, known tactics, Techniques and Procedures (TTP) and attack campaigns. Use of information on threats, threat prevention, controls to interpret large data security in action intelligence activity in the form of descriptions and leading indicators. These indicators are the logic that forms the layer that executes judgments of performance and permits organizations to predict attacks and identify their importance when it is detected in the network.

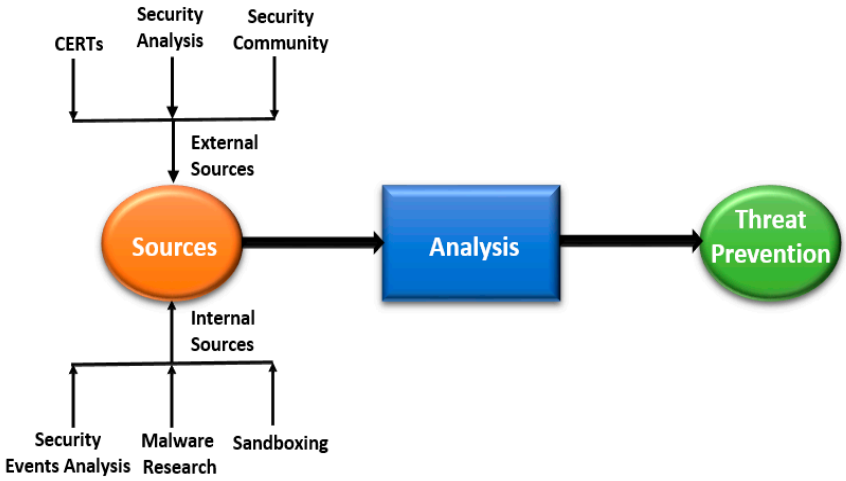

Figure 4. Threat intelligence.

For example, collecting raw intelligence detected a striker that targets the economic sector targets. The goal of the attacker is to attach malware exploiting vulnerabilities reading documents and spread it over a variety of channels such as USB drives, email, and hijacked websites. When the user runs the unsuspicious malware, the malware uses a Remote Access Tool (RAT) to give the right of entry to the internal network of the attacker. To generate usable information, the sandbox application layer performs a process and checks whether the document contains malware that infects by inserting a file on the local file system. The sandbox calculates the unique hash of the file and document and presents

these indicators as an action to SDO. Based on these indicators, then the SDO provides security against attacks and automatically distributes reinforcements. Indicators are shared with other organizations in the community. Big data analysis for evidence of network data and host file systems is done to limit the damage post-infection by generating hashes of files and documents. This facilitates the identification of the compromised hosts, and automatically or manually guarantees to offer a policy for these hosts by controlling their access privileges on the network. The analysis of security events and logs can help identify a statistical connection between specific outgoing connections and suspicious hosts.

Threat indicators are especially effective when they identify threat patterns of behavior that are relatively expensive to change from attack to attack. For example, there is no benefit to the initiator blocking the address of an intrusion if the address is forged by the attacker and is stochastically connecting to the connection. However, connecting a bot server is more difficult to change because the attacker must define a new server every time the former is blocked. Advanced attacks have yet to find a complex corresponding threat indicator. For example, at present malware could generate a random server URL, which enables it to use a large number of potential servers. The analysis of this algorithm generates an equivalent complex indicator that can recognize all the URLs that are used in the attack.

With the recognition of abnormal and malicious sources, threat indicators are generated within the business. Internal sources may include the logic to control the security of the application layer to carry out checks in the sandbox environment. This produces threat indicators for any misconduct detected on the applications and documents that may potentially have malware. Furthermore, analysis of security events obtained from the application layer assists in recognition of attacks and anomalies. After these threats are identified, threat indicators are established to provide containment for entities that compromise and block new attacks. Forensic analysis of hosts and networks by security analysts produce threat indicators and provide SDO for distribution to reinforcements. Honeypots are used to lure attackers into thinking they accessed the internal network. Then, the protective analyzes TTPs attackers create appropriate threat indicators and block the attack.

Nowadays, attackers are targeting corporate resources by exploiting potential vulnerabilities in the system. As we explained earlier, they prevent threats by identifying threats and control by preventing their behavior. However, system vulnerabilities are known and exploited by attackers before they are published and known by the owners of the community and the security system. These vulnerabilities are known as "Zero-day". In a sense, Zero-day attacks cannot be stopped because no direct intelligence of threat exists before. There are some approaches of protection to protect against zero-day attacks—sandboxing attack of the surface, minimizing behavioral controls and anomaly detection, the man in the loop. Moreover, patching vulnerable applications and use of IPS controls based on intelligence threats reduce the window of exposure for the identified vulnerabilities, which means that exploits are mitigated as soon as vulnerabilities are revealed. Although this practice does not prevent zero-day exploits, the great majority of sophisticated attacks use known but unpatched vulnerabilities as leverage.

3.3. SQUEAK

Attackers often spread through the network to make use of the insider's vantage points, and afterward, commence attacks on the internal network. Therefore, the objective is to confirm the attacks begun on data plane forwarding, network topology and identify policy breaches inside the SDN. This makes our threat model ideal to detected attacks launched from inside the SDN. Thus, SDNs are modeled as an unopened system. Restriction elimination on the unknown external communication allows our system to focus on OpenFlow control messages inside the SDN because the SDO in the FS-OpenSecurity model handles all these issues.

SDN setup is considered without traffic through the OpenFlow and non-OpenFlow network entities. We believe that the trust controller is essential for the accurate operation of the network. The switches can carry their own identity as the switches are connected to the control unit via separate TCP connections, usually with TLS is enabled. Notwithstanding this, we accept that the

greater part of switches in the network are trusted. Since the majority of SDN systems operate as modules under control of the bit, they are also dependable, as long as the controller is itself confident. The above assumptions mean that the communication OpenFlow controller to the switches is trustworthy. However, switches to the controller are unreliable, and can be produced by a malicious or switch.

SQUEAK brings up two serviceable functions to OpenFlow existing architecture: an analysis function of the flow rule and a packet migration function. The analyzer module maintained the principle usefulness of the system foundation when the immersion attack occurred. The packet migration module sends out information to the network flow OpenFlow controller with no overload. An abstract architecture of SQUEAK is shown in Figure 5. The flow rule analyzer module functions as a controller on top of the controller. The packet migration module also performs as a controller.

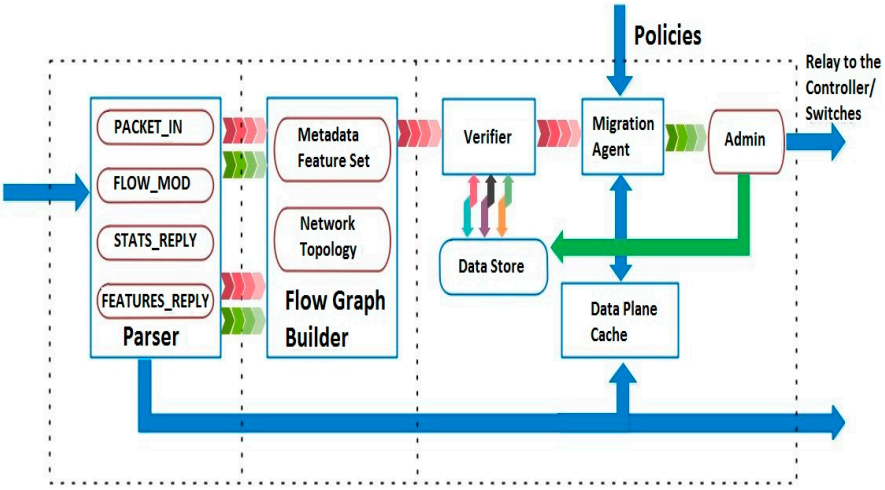

Figure 5. SQUEAK flow diagram.

Figure 5 shows SQUEAK's workflow, which has three stages. First, SQUEAK monitors and passes all communications and recognizes the appropriate OpenFlow messages essential to building an overview of the network. Second, the analysis SQUEAK OpenFlow messages and extracts the metadata and the forwarding topological state to build a network of the incremental graph with the traffic flow. Particularly, SQUEAK maintains the topological state of metadata and registers forwarding from incoming packet headers such as OpenFlow, outbound flow path configuration guidelines and concrete measures of traffic flows on network connections, respectively. Third, this SQUEAK valid metadata flow of a set of allowable values of metadata is collected during the life of a flow and its administrative policies. SQUEAK flags known attacks by the policies specified administrator, even though it is the most specific flow activity achieved over time to detect unanticipated and potentially malicious activity. SQUEAK does not produce warnings when it discovers a new flow behavior. Alternatively, SQUEAK triggers alerts when it detects untrusted entities that trigger modification of the behavior of existing flow, or if the flow defies any specified security policy administrator. For example, SQUEAK does not trigger alarms when a switch discovers its neighbors. However, if one of the neighbors alters on a switch port, SQUEAK immediately reports a happening, because it changes the network topology and therefore the flow graph. In addition, SQUEAK will not generate alerts on redirected flows because they are generated by the messages of the confident FLOW_MOD controller. This significantly reduces the alarm if the detection of each new behavior is marked, which is increasingly likely in networks. These suppression alerts also mean that any malicious movement that causes a flow authentic behavior will be supported by the findings of SQUEAK activities and avoid immediate detection. Nevertheless, malicious activity will be detected when looking back later at Squeak flags authentic activities as suspect, only to be disabled by the administrator.

However, two or more malicious end hosts or switches can still poison the controller notice by generating a false bidirectional connection, probably changing the shortest path routing between other hosts on the network. SQUEAK identifies such false links validating metadata forwarding data plane flows, which records the patterns of real network traffic flow along a path in the flow graph. Especially, SQUEAK applies a custom algorithm to monitor and observe the flow of bytes statistics by grabbing STATS_REPLY messages to each switch in the flow path, and deciding whether the switches are accounting discordant values of bytes transmitted.

SDNs offers three main features that facilitate SQUEAK to detect security threats in real time with precision. Ease of analysis: SDN are as dynamic as the Internet, and OpenFlow is less complicated than conventional communication protocols. All brains are centralized in the controller, where the flow of all the networks to inform is recognizable. This makes the analysis of control messages inside SDN substantially easier. Action ascription: Action ascription in SDNs is much lighter than in conventional networks, due to the centralized control unit which has a universal visibility and a lot of statistics obtained from the controller. Domain knowledge: If we do not believe SDN to be a black box, we influence the domain knowledge regarding OpenFlow to build a small but communicative feature set that records the essence of all network communications. This helps to effortlessly notice changes in the control message patterns.

SQUEAK aims to provide precise and real-time authentication behavior provided by three main features. Firstly, it monitors and observes all appropriate OpenFlow control messages from the controller or switches. Then, it uses a set of concise functionality that allows the appropriate authentication and verification of these messages. Then it applies a custom algorithm for rapid validation of network updates as they are litigated by the controller.

Intercept OpenFlow Packets: SQUEAK must capture each network information to be able to identify abnormal behavior. SQUEAK is a mediator between switches and the controller. A switch or an end host only abuses a subset of all messages in OpenFlow to destroy or poison the view of the network controller. These messages include PACKET_IN, FLOW_MOD, STATS_REPLY, and FEATU-RES_REPLY. However, the confidence controller makes use of FLOW_MOD messages to direct the switches to establish connectivity between end points. Thus, only SQUEAK dynamically monitors these OpenFlow messages to extract the relevant metadata. All other messages are simply passed through.

Build Incremental Flow Graphs: SQUEAK analyses control messages OpenFlow as previously described to build and modify the flow graphs associated with flows in the network. It then identifies violations or attacks in the security policies recognizing the actual change in the topological transfer metadata and network data plan associated with each flow graph. There are three entities in an SDN environment that accurately illustrate the metadata for each flow in the network: streams, switches, and end hosts. SQUEAK extracts and stores metadata information associated with each entity to perform a set of features. MAC/IP source and destination connections have a match for all hosts on the network. The MAC/port bindings exclusively recognize a flow between the ends. The flow corresponds with the port and switches off when it decides that all waymarks flow in the data plane. Finally, flow statistics provide packets/bytes transferred for each flow.

Moreover, SQUEAK absorbs and maintains this logical and physical topology flows, and FLOW_MOD transmission status messages under each intermediate switch in the flow path, to discover the malicious updates metadata. SQUEAK relies on four messages from OpenFlow PACKET_IN, FLOW_MOD, STATS_REPLY FEATURES_REPLY and it extracts all the appropriate metadata as observed in a port and a particular switch. In particular, SQUEAK discovers the appearance of flows and topological information such as host MAC-port, IP-MAC binding or port switch when the switches generate PACKET_IN. Preferred paths to be taken by the flow, and subsequent updates, are resolved when the controller generates an FLOW_MOD for switches. SQUEAK operates STATS_REPLY, which sometimes receives switches, to extract the statistical level of

flow in the data plane, including bytes/packets transferred. SQUEAK catches FEATURES_REPLY to gather switching configurations such as port status when a switch first connects to the controller.

Validate Network Behavior: Substantiation of restrictions on network features, resources and flow properties is carried out by the engine of SQUEAK policy. SQUEAK speedily validates the different effects of all network updates on individual flows by crossing the flow graph and examining the related metadata for compliance with the administrator or the specified application protection properties. Treating the flow graph during each update of the network is the time to do the job. Thus, SQUEAK caches the current path of milestones to decide whether the update provides constraints or not. In case the network updates change the existing track structure, such as migration of virtual machines in the multi-tenant data centers, SQUEAK rejects the waymarks that have been cached, reconstitutes the current path and the cross for marking consistency as dependencies, and any specified security policy administrator. Incremental flow graphs and flow metadata ensure that the validation process is faster since during each update SQUEAK must only think about metadata for a specific network connection for a single flow. This design dies not only check constraints extremely fast, but it also develops an easier assignment action and has high accuracy.

We present a finite state machine to deal with the data plan cache in the saturation attacks in SQUEAK. The state machine is shown in Figure 6. Before all states, SQUEAK undergoes some preparation such as symbolic execution to create a set of road conditions for each function PACKET_IN manager of all the applications. Compared to the conventional approach of symbolic execution, not only are the input variables served, but also the global variables used in the PACKET_IN handler function are signified. After the preparation work, SQUEAK starts from the resting state. First, if there is no attack, both the analyzer of traffic rules and packet migration agent modules stay at rest. When a saturation attack is detected, SQUEAK comes to the state Init. The migration agent module starts transmitting the table packet miss in the data plan cache in the state Rule Generation. The flow rule analyzer will track running control systems and switching conditions of a road that are pre-set before the flow rules. At the same time, the cache module data plane starts to process the packets cached and generated PACKET_IN messages to the controller. When the flow rules are ready, SQUEAK updates the state of the flow rules. The analysis module squarely installs the rules to the data plane switches and maintains the updating that these rules flow. Once all the rules are installed, SQUEAK begins to apply the rules updated in the mitigation attack state. The data plan cache continues handling unprocessed packets. When the cache complete data plane processing of all the packets is cached, and the attack is identified as over, SQUEAK stops. It sends a reset signal to the migration agent module, which stops the migration table packet miss and instigates a state of rest again.

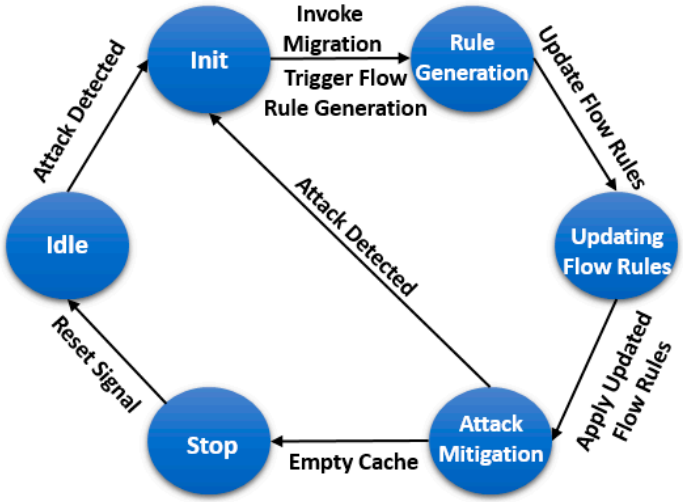

Figure 6. The states of SQUEAK's workflow.

Verifier: The primary purpose is to change the host IP addresses that were recurrently allocated and create vIPS (virtual-IP) guests with wide levels of transparency and unpredictability of end hosts. The genuine IP addresses are only accessible by authoritative entities.

$$\forall \left(\sum_{1 \le i \le n} c_{ik} R_i \right) \times T \le \sum_{1 \le j \le m} b_{jk} \left| r_j \right| \tag{1}$$

Ranges must be allocated to subnets proportional to their total required alteration rate. Equation (2) defines the volatility constraint and set up that the ranges must be allocated to subnets in proportionality to their total required alteration rate.

$$\forall k, P_k = \frac{T \times \sum_{1 \le i \le n} c_{ik} R_i}{\sum_{1 \le j \le m} b_{jk} \left| r_j \right|} \tag{2}$$

Where P_k is the total essential alteration of subnet n_k during time T on total range size assigned to it and P_a as given in Equation (3) is the total essential alteration of all hosts on the total size of vacant address ranges.

$$P_a = \frac{T \times \sum_{1 \le i \le n} R_i}{\sum_{1 \le j \le m} \left| r_j \right|} \tag{3}$$

The unpredictability constraint is represented as $\forall k, P_k \cong P_a$. The constraint means that the total size of allocated address spaces for a subnet should be proportionate to the requirements of the subnet. To address the constraint with SMT, we rewrite it as $\forall k, |P_k - P_a| < \sigma$, σ is a constant value.

A verifier invalidates the information gathering of external scanners, and it also saves the network hosts from even zero-day unknown attacks.

We initially divide the entire process into two steps. First, our input is a PACKET_IN based manager, and our desired output conditions are a path. First there is the input variables PACKET_IN and global variables that are used in the PACKET_IN. After that, we will symbolize the two variables and use a symbolic classic algorithm to cross all possible branches and collect all road conditions. On'the basis of this we will create the conditions of the path. Since this step is relatively long, let us treat this step, an offline, symbolic component, ahead of the runtime, which means it will not increase the overhead in our system. All of these steps are summarized in Algorithm 1.

Algorithm 1 Path Conditions Generation

Input: *IN*: PACKET_IN handler function
Output: *PC*: Set of path conditions
Begin
  input is NULL, *global* is NULL
  input = *findInputVariables*(*IN*)
  global = *findGlobalVariables*(*IN*)
  IN' = *symbolic*(*IN*, *input*, *global*)
  PC = *symbolicExecutionEngine*(IN')
  return *PC*
End

The next step is dynamic analysis and converting it into the reactive flow rules. It will keep track and allot the current value of the global variables to the condition path. Then, only input variables are symbolized in the path conditions and the reactive flow rule dispatcher components are used to analyze each condition path. For paths, the final decision (handling a small set change to generate

a status message) is considered. Finally, the reactive flow rules that we want are defined. All these steps are summarized in Algorithm 2.

Algorithm 2 Converting to Reactive Flow Rules

```
Input: PC: Set of path conditions,
            global': Global variables with real-time assigned value
Output: FR: Reactive flow rules
Begin
  FR is NULL
  paths = assignedValue(PC, global')
  For each path condition pc ∈ paths do
    If pc.check = mutateState Then
      FR.add (convert(pc.pathCondition))
    End If
  End For
  return FR
End
```

Migration Agent: The migration agent module is the core of SQUEAK. Based on the types of alters, it identifies attacks and makes decisions. When receiving DOS/DDoS attack changes, it will trigger the state transition in the state machine explained above. Another task of the migration agent module is to migrate the table packet miss in the data cache map. When the saturation attack is identified, it will change the system state to state Init, which triggers both the analyzer flow rule for generating new rules and the data plan cache. The migration agent module installs an entry of generic flow rule that has a modest priority and, before this, all the table packets miss the data plan cache. Consequently, flood packets will not flood the controller or overload the switch. The data plan cache will store all the time table packet misses. Some packages will be given priority to generate messages in the PACKET_IN data plan cache. While the data plan cache creates PACKET_IN messages, it cannot immediately send the messages to the controller. The control platform differentiates the various data paths (switches) from different connections. If the map data cache immediately sends messages to the controller, the data plan cache will be recognized as a new data path. Thus, another feature of the migration agent module solves the problem. The cache data plane PACKET_IN sends messages to the migration agent module. Then, the migration agent will start a PACKET_IN procedure with the original data path information to other applications in the controller. In addition, the migration agent load flow related data plan cache is based on the use of system resources and the flow of incoming messages of PACKET_IN.

The cache data plan is a machine that, for now, cached table packet misses when attacking at a point of saturation. The majority of flood traffic is redirected to the data plan cache as an alternative to flood the OpenFlow architecture during data attacks, which masters the saturation. The data plan cache has three components: a packet buffer queue, a packet classifier, and packet in the generator. When a table-miss packet is transferred between the data plan cache, the packet classifier analyses the header of the packet and attaches to its corresponding buffer queue. Above all, there are four packet buffer queues contained in aircraft cached data based on the packet protocol such as ICMP, TCP, UDP, and by default, which is based on First-in, First-out. Between these four packet buffer queues, we use the round-robin scheduling algorithm to choose the header packet queue for future PACKET_IN generating messages. The sending rate of PACKET_IN generator is remotely limited by the migration agent in the OpenFlow controller.

3.4. FS-OpenSecurity Service Scenarios

Centralized Structure: For a FS-OpenSecurity model in the centralized structure, the standard arrival rate of flow requests is $\Psi_c = N \times (N-1) \times \Psi$. It has a Poisson distribution and has an exponential distribution processing time of controller. The scale of the network N is inversely proportional to the average processing rate $\delta_c(N)$. Then, $\delta_c = \frac{K}{g(N)}$. Therefore, the average flow initiation response time is

$$E\{T_c(N)\} = \frac{1}{\delta_c - \Psi_c} \tag{4}$$

Decentralized Structure: The decentralized structures have numerous controllers, let assume the number of controllers is m_D and each of the controllers manages a local network. In FS-OpenSecurity, we manage the decentralized structures in two different ways. One way is the local view scheme, and another way is the global view scheme. In the local view scheme, every controller has its local network aspect and all of its neighboring local networks are considered as a logical node. The flows are separated into two types: local flows and global flows. If the same controller controls the flow's source and destination host, then it is the local flow, otherwise it is the global flow. The initiation request of a global flow is processed by multiple controllers, whereas the initiation request of a local flow is processed by only single controllers. When a controller gets a global flow induction request, then the flow request will be divided into two initiation requests. One request is a local flow induction request for the controller, and another is either a global or local flow initiation request for each one of its neighboring controllers. The flow request is treated as above as far as there are no global requests. Let us suppose the average controller distance is d_H, then each global flow initiation request will be divided into $d_H + 1$ local requests on average. The total of the reply time for these local requests is the response time for the global flow initiation.

There are flows $N^2 - \frac{N^2}{m_D}$ global and $\frac{N^2}{m_D} - N$ local flows among the $N \times (N+1)$ flows. Thus, the advent rate of flow induction request at each controller is:

$$\Psi_{D,l} = \Psi \times \frac{\left(N^2 - \frac{N^2}{m_D}\right) \times (d_D + 1) + \left(\frac{N^2}{m_D} - N\right)}{m_D} \tag{5}$$

Each controller has to control a topology with less than $\frac{N}{m_D} + m_D - 1$ nodes. As a result, the average service rate of the controller is assumed to be:

$$\delta_{D,l} = \frac{k}{g\left(\frac{N}{m_D} + m_D - 1\right)} \tag{6}$$

The queue length for each controller as per Little's law is:

$$L_{D,l} = \frac{\Psi_{D,l}}{\delta_{D,l} - \Psi_{D,l}} \tag{7}$$

The average response time of each request is:

$$E\{T_{D,l}\} = \frac{L_{D,l} \times m_D}{N \times (N-1) \times \Psi} = \frac{1 + \frac{N \times (m_D - 1)}{(N-1) \times m_D} \times d_D}{\delta_{D,l} - \Psi_{D,l}} \tag{8}$$

In the global view scheme, each controller has a global aspect of the network, and every controller processes all the flow induction requests produced by its local networks. The controller average service rate is $\delta_{D,g} = \frac{K}{g(N)}$. For the request at each controller, the average arrival rate of initiation is $\Psi_{D,g} = \frac{\Psi \times N \times (N-1)}{m_D}$. Thus, the average response time of flow initiation requests is:

$$\mathrm{E}\left\{\mathrm{T}_{\mathrm{D,g}}\left(\mathrm{N}\right)\right\} = \frac{1}{\delta_{\mathrm{D,g}} - \Psi_{\mathrm{D,g}}} \tag{9}$$

Attack Scenarios: Figure 7 depicts the attacker exploring for potential components of SDN to compromise. To build a complete secure SDN environment, it is necessary to ensure the security of every component on SDN. To secure the SDN key components, FS-OpenSecurity took all the essential steps. In Figure 8, we depict how the system is put together in FS-OpenSecurity to build a complete secure SDN environment.

Whenever a new packet arrives at the switch, the system checks whether it belongs to an existing flow or not. If so, it updates the flow statistics. Otherwise, it is sent to the controller via SQUEAK. There is no direct communication from data plane to controller. The packets that do not have any entries in switch flow tables are marked as suspicious packets. When SQUEAK receives such packets, it builds incremental flow graphs and validates the network behavior. If necessary, SQUEAK will communicate and synch with the controller or SDO. Once SQUEAK validates the packets, and if the validation result indicates attacks, then it will issue an alert and take necessary steps. In addition, it will communicate the same to the controller to take essential steps to mitigate the attack.

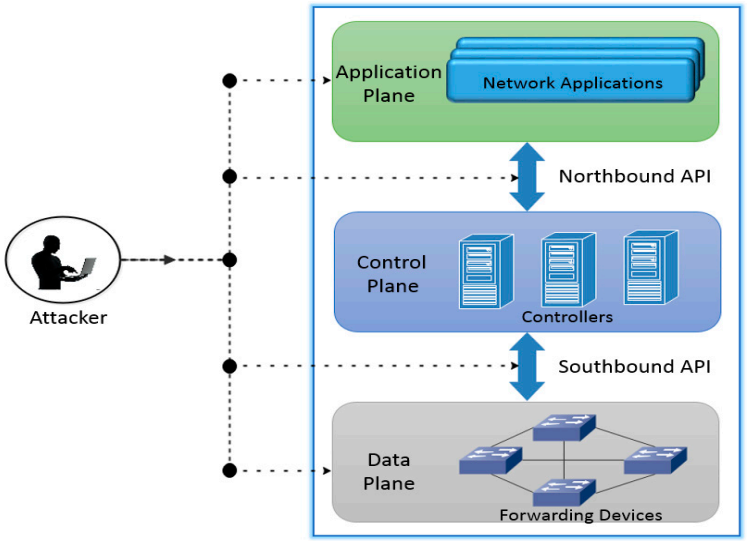

Figure 7. Attacker searching for potential targets.

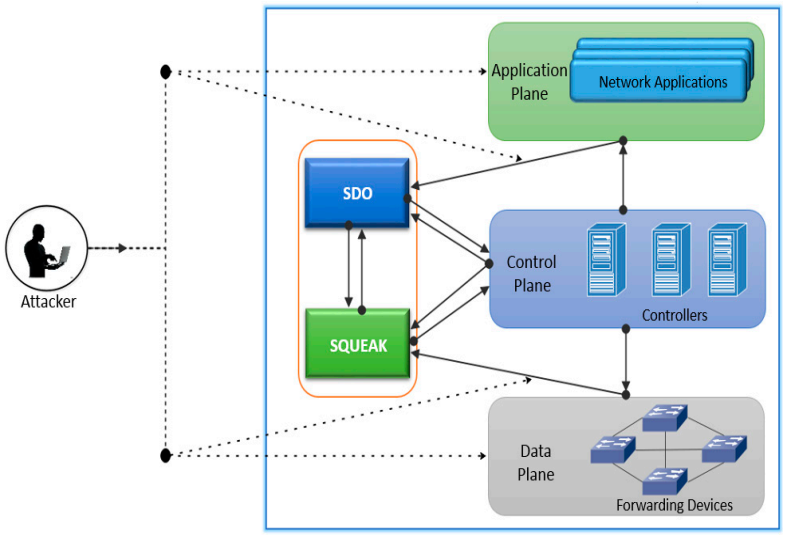

Figure 8. FS-OpenSecurity's workflow to secure SDN environment.

An attacker may be able to gain unauthorized access to the SDN components by exploiting software vulnerabilities at the application or management layer. APIs and communication channels can also be potential targets for the attackers. Generally, the control plane offers APIs for applications to query SDN system information and to monitor and respond to network changes. The applications or APIs built using APIs may contain vulnerabilities that could permit an attacker to perform different information disclosure attacks. To handle these issues, FS-OpenSecurity provides security at the application layer as well as at communication channels. There will be no direct communication between the application plane and control plane. Each and every request will go to the controller via SDO. SDO will act as the brain of the controller. SDO will also sync with SQUEAK to monitor and take necessary mitigation steps in case of any DDoS or similar attacks.

Because of its dynamic and flexible architecture and no direct communications with the central controller, FS-OpenSecurity monitors and handles the attacks at all planes as well as at communication channels or at the interface level.

3.5. Analysis of FS-OpenSecurity

In this section, we analyze the FS-OpenSecurity model to compare it against approaches that are proposed in earlier research. The analysis is based on security issues—Confidentiality, Integrity, and Availability—in an SDN environment. We have categorized the SDN security countermeasures into three different levels: OpenFlow switch, controller, and channel.

Switch-related Security: As mentioned above, the OpenFlow switch is vulnerable to a variety of attacks such as hijacking, spoofing, scanning, DoS/DDoS, forgery, and so on. To address these security concerns, the FS-OpenSecurity model proposed in this paper enhances security at the switch with, e.g., the implementation of the policy and authorization. A list of safety efforts related to switching is given in Table 1. Most of the time, the certificate of software, flow verification, and trust management methods will handle all security issues at the switch.

Table 1. Security comparison at switch level.

Technique	Year	Confidentiality	Integrity	Availability
FlowChecker [25]	2010	√	√	
NPoL [26]	2014	√	√	
OpenRouter [27]	2011	√	√	√
OF-RHM [15]	2012	√	√	
FlowMon [28]	2015	√	√	
VeriFlow [29]	2012	√	√	
OpenSec [30]	2014	√	√	√
FlowNAC [31]	2014	√	√	√
AnonyFlow [32]	2012	√	√	
OF-GUARD [33]	2014	√	√	√
FS-OpenSecurity	2016	√	√	√

Controller-related Security: To defend the SDN controller from a variety of threats such as DoS/DDoS attacks, validation mechanisms and trust models should be added to protect this key element. An FS-OpenSecurity model performed the validation of source address and took a countermeasure against attacks exhaustive resources such as DOS/DDoS. It also handles the overflow of the buffer and the message PACKET_IN fall for network control. Several proposed approaches are summarized in Table 2. In order to overcome this kind of security problems, it is measured as the application of security policy, trust models, and monitoring tools.

Channel-related Security: To mitigate these security problems, a secure and clean channel should be protected. Apparently, encryption is essential and the first step to achieving this goal. For example, the SSH application, TLS or other techniques are used to secure the management of the controller and communication to the north. Then communications services and applications that require data

or services from the controller must be secured using the methods of encryption and authentication. The SDO layer in the FS-OpenSecurity proposed model handles all these security issues and provides highly secure and efficient communication dynamics. Several approaches proposed in the literature are listed in Table 3. Credential Verification, encryption, and trust mechanisms are applied to improve the security of the channel between the switches and controller, and between the controller and applications.

Table 2. Security comparison at controller level.

Technique	Year	Confidentiality	Integrity	Availability
NICE [34]	2012	√	√	
OPERETTA [35]	2015			√
TopoGuard [36]	2015	√	√	√
FlowGuard [37]	2014	√	√	
ARM [38]	2012			√
FortNOX [39]	2012	√	√	√
VAVE [40]	2011	√	√	√
CONA [41]	2010		√	√
HyperFlow [42]	2010			√
FloodGuard [43]	2015			√
PermOF [44]	2013	√	√	
FS-OpenSecurity	2016	√	√	√

Table 3. Security comparison at channel-related.

Technique	Year	Confidentiality	Integrity	Availability
AVANT-GUARD [21]	2013	√	√	√
HIP [45]	2014	√	√	√
FlowVisor [46]	2010	√	√	
FS-OpenSecurity	2016	√	√	√

3.6. Discussions

Different protections are required on various enforcements. To select protection category, the user authorizations, segment assets, the threat environment operational constraints and system performance should be considered. The SDO's role selects the suitable security control logic. The appropriate logical security control will be applied to limit the application of each segment to enforce data protection policies and access control and challenge the identified threats. Similarly, SQUEAK is a controller tool that takes advantage of flow graphs to identify security threats on data plane forwarding and network topology originating within SDNs. SQUEAK is built incrementally and uses flow graphs with compendious data for network flows and uses both probabilistic and deterministic control.

Security controls are selected as below. First, a risk analysis is performed for the segment or a group of individual segments. The risk is the intensity of the impact and the likely presence of security incidents on resources or functions of an organization. The events of this security event consist of threats, breaches of security policy, and the flow of incorrect data. Understanding the risks provides a priority framework for security checks. Different risk categories are examined for each communication that crosses a boundary of the outer segment. A systematic risk is based on the impact, probability of success and potential damage. Risks can be designed to a high standard or can be well prepared for potential attack techniques, as described in Table 4. A set of security controls is determined to mitigate each risk, reducing exposure to a level that suits the company. A simplified summary of observation risk mapping defenses is described in Table 5. Each row illustrates an elaborated attack method like malware sent as a link in an email.

Table 4. A Sample Risk Categorization.

Risk	Risk Description
Insider	An authorized user performs communication that the temperaments of security policy
External Attack	An external entity attempting to gain unauthorized access to resources or services
Data Access	An attacker reads or modifies data in the storage architecture or data at rest by accessing the network or transit
Data Leakage	Sensitive data is carried out unauthorized users or written to the flash media
Exploit	An attacker performs a protocol violation causing the system to malfunction
Malware	Malicious code injected on the network or I/O devices removable wrong impacts of business resources
Denial of Service	An interaction consumes extreme amounts of storage or network capacity, treatment, denial of service for authorized interactions
Traffic Hijacking or Re-Routing	Illegitimate takeover of routing group addresses by corrupting the routing tables

Table 5. Mapping FS-OpenSecurity controls to risks.

Risk	Access Control		Threat Prevention		Data Protection	Network Topology/ Data Plane Forwarding
	Inbound	Outbound	Pre	Post		
External Attack	√		√	√	√	√
Insider Attack	√	√		√	√	
Traffic Hijacking or Re-Routing						√
Denial of Service	√		√			√
Malware	√	√	√	√		
Exploit	√		√			
Data Leakage		√		√	√	
Data Access					√	

Risk mapping protections help ensure that all risks are sufficiently mitigated to find out what reinforcements should be used to control security during the examination of communications that go throughout several reinforcements. Mapping protections also discover the residual risk and regulating security checks when a given control starts to fail, requires excessive resources or is too expensive. The FS-OpenSecurity controls enforcements so that any risks related to all communication is controlled along the complete path interaction. The following are the general approved rules for where to apply each key control:

Pre-infection Threat Prevention security controls and Inbound Access Control like user identification, IPS, and firewall should be enforced as close to the resources as possible. This provides more granular controls tailored to the specific resources and reduces the risk of a security bypass.

DoS/DDoS controls should be put into action at the organizational level due to higher chances, risks and the motivation of these hacker attacks.

Pre-infection anti-malware controls should be implemented at the organizational boundary as they are created by external entities. In addition, anti-malware controls are put into service on mobile devices that treat documents and on end hosts that can contain malware. Selection of enforcement should take data encryption and performance issues into account, e.g., encrypted messages must be decrypted before being considered for malware.

Post-infection threat prevention controls need to limit the right to use external applications in the organizational perimeter. Cooperative intelligence is applied to explore the high-risk and target

applications. To provide containment capability threats, outbound network access can also be set on end hosts.

Network-level data loss prevention controls are put into operation by rights to the classification scheme. When data is exported outside the organization, internal information must be controlled. However, the departmental data should be illegal within the corporate segments. In addition, to protect against threats to data access, encryption controls must be installed on mobile devices, end hosts, and cloud computing environments.

4. Conclusions

In this paper, FS-OpenSecurity, an architecture model that makes use of network monitoring and SDN control functions for developing secure model architecture was presented. FS-OpenSecurity aspires to improve network security by cutting down the overhead on SDN controllers through the dissociating of control and monitoring functions.

The role of the FS-OpenSecurity architecture is to provide protection at each layer and interface. The protections included: data protection, access control, prevention of threats, and protection that campaigns against network attacks such as DDoS, ARP spoofing or hides poising and detects security threats on the flight return data and the network topology. By mapping these analytical protection controls to the risk in each segment and its resources, an enterprise applies a robust multi-layer defense of any attacks, including advanced persistent threats (APTs). To deploy the appropriate protections, the FS-OpenSecurity is based on the repositories of data that consist of knowledge of data resources and their assortments, knowledge of the organization and its information systems, and knowledge of threats. In conclusion, it is also essential for businesses today to carry out the risk analysis of each segment, map risks assessed for appropriate security controls and analyze communication channels to protect the more for every application.

Acknowledgments: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2016R1A2B4011069).

Author Contributions: Yunsick Sung: Research for the related works, analyzing of the proposed model and drafting the article. Pradip Kumar Sharma: Acquisition of data, analysis, and interpretation of related works, conception, and design and meliorating the complete model. Erik Miranda Lopez: Research for the related works, and readability, grammar, and spelling checks of the article. Jong Hyuk Park: Total supervision of the paperwork, review, comments, assessment, etc.

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

References

1. Jain, S.; Kumar, A.; Mandal, S.; Ong, J.; Poutievski, L.; Singh, A.; Venkata, S.; Wanderer, J.; Zhou, J.; Zhu, M.; et al. B4: Experience with a globally-deployed software defined WAN. *ACM SIGCOMM Comput. Commun. Rev.* **2013**, *43*, 3–14. [CrossRef]
2. Sezer, S.; Scott-Hayward, S.; Chouhan, P.K.; Fraser, B.; Lake, D.; Finnegan, J.; Viljoen, N.; Miller, M.; Rao, N. Are we ready for SDN? Implementation challenges for software-defined networks. *IEEE Commun. Mag.* **2013**, *51*, 36–43. [CrossRef]
3. Douligeris, C.; Serpanos, D.N. *Network Security: Current Status and Future Directions*; John Wiley & Sons: Hoboken, NJ, USA, 2007.
4. Casado, M.; Garfinkel, T.; Akella, A.; Freedman, M.J.; Boneh, D.; McKeown, N.; Shenker, S. SANE: A Protection Architecture for Enterprise Networks. Available online: https://www.usenix.org/legacy/event/sec06/tech/full_papers/casado/casado_html/ (accessed on 8 September 2016).
5. Casado, M.; Freedman, M.J.; Pettit, J.; Luo, J.; McKeown, N.; Shenker, S. Ethane: Taking control of the enterprise. *ACM SIGCOMM Comput. Commun. Rev.* **2007**, *37*, 1–12. [CrossRef]
6. Kloeti, R. OpenFlow: A Security Analysis. Available online: ftp://ftp.tik.ee.ethz.ch/pub/students/2012-HS/MA-2012-20.pdf (accessed on 8 September 2016).
7. Hernan, S.; Lambert, S.; Ostwald, T.; Shostack, A. *Threat Modeling-Uncover Security Design Flaws Using the Stride Approach*; MSDN Magazine: Louisville, KY, USA, 2006; pp. 68–75.

8. OpenFlow Switch Specification Version 1.3.2, Open Networking Foundation. Available online: https://-www.opennetworking.org (accessed on 8 September 2016).
9. Benton, K.; Camp, L.J.; Small, C. Openflow Vulnerability Assessment. In Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, Hong Kong, China, 12–16 August 2013; pp. 151–152.
10. Kreutz, D.; Ramos, F.; Verissimo, P. Towards Secure and Dependable Software-Defined Networks. In Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, Hong Kong, China, 12–16 August 2013; pp. 55–60.
11. Li, D.; Hong, X.; Bowman, J. Evaluation of Security Vulnerabilities by Using ProtoGENI as a Launchpad. In Proceedings of the Global Telecommunications Conference (GLOBECOM 2011), Houston, TX, USA, 5–9 December 2011; pp. 1–6.
12. Anwer, B.; Benson, T.; Feamster, N.; Levin, D.; Rexford, J. A Slick Control Plane for Network Middleboxes. Open Networking Summit. 2013. Available online: http://nextstepesolutions.com/Clients/ONS2.0/pdf/2013/researchtrack/posterpapers/final/ons2013final51.pdf (accessed on 8 September 2016).
13. Fayazbakhsh, S.; Sekar, V.; Yu, M.; Mogul, J. FlowTags: Enforcing Network-Wide Policies in the Presence of Dynamic Middlebox Actions. In Proceedings of the Second Workshop on Hot Topics in Software Defined Networks, Hong Kong, China, 12–16 August 2013.
14. Qazi, Z.A.; Tu, C.C.; Chiang, L.; Miao, R.; Sekar, V.; Yu, M. SIMPLE-fying middlebox policy enforcement using SDN. *ACM SIGCOMM Comput. Commun. Rev.* **2013**, *43*, 27–38. [CrossRef]
15. Jafarian, J.H.; Al-Shaer, E.; Duan, Q. Openflow Random Host Mutation: Transparent Moving Target Defense Using Software Defined Networking. In Proceedings of the First Workshop on Hot Topics in Software Defined Networks, Helsinki, Finland, 13–17 August 2012; pp. 127–132.
16. Braga, R.; Mota, E.; Passito, A. Lightweight DDoS Flooding Attack Detection Using NOX/OpenFlow. In Proceedings of the 2010 IEEE 35th Conference on Local Computer Networks (LCN), Washington, DC, USA, 10–14 October 2010; pp. 408–415.
17. Ballard, J.R.; Rae, I.; Akella, A. Extensible and Scalable Network Monitoring Using OpenSAFE. In Proceedings of the INM/WREN, San Jose, CA, USA, 28–30 April 2010.
18. Shin, S.; Gu, G. CloudWatcher: Network Security Monitoring Using OpenFlow in Dynamic Cloud Networks (or: How to Provide Security Monitoring as a Service in Clouds?). In Proceedings of the 20th IEEE International Conference on Network Protocols (ICNP), Austin, TX, USA, 30 October–2 November 2012; pp. 1–6.
19. Mehdi, S.A.; Khalid, J.; Khayam, S.A. Revisiting Traffic Anomaly Detection Using Software Defined Networking. In *Recent Advances in Intrusion Detection*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 161–180.
20. Shin, S.; Song, Y.; Lee, T.; Lee, S.; Chung, J.; Porras, P.; Yegneswaran, V.; Noh, J.; Kang, B.B. Rosemary: A Robust, Secure, and High-Performance Network Operating System. In Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, 3–7 November 2014; pp. 78–89.
21. Shin, S.; Yegneswaran, V.; Porras, P.; Gu, G. Avant-Guard: Scalable and Vigilant Switch Flow Management in Software-Defined Networks. In Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, Berlin, Germany, 4–8 November 2013; pp. 413–424.
22. Shin, S.; Porras, P.A.; Yegneswaran, V.; Fong, M.W.; Gu, G.; Tyson, M. *FRESCO: Modular Composable Security Services for Software-Defined Networks*; National Down Syndrome Society: New York, NY, USA, 2013.
23. Porras, P.A.; Cheung, S.; Fong, M.W.; Skinner, K.; Yegneswaran, V. *Securing the Software Defined Network Control Layer*; NDSS (National Down Syndrome Society): New York, NY, USA, 2015.
24. Lee, S.; Lee, C.; Jo, H.; Kim, J.; Lee, S.; Nam, J.; Park, T.; Yoon, C.; Kim, Y.; Kang, H.; et al. A Playground for Software-Defined Networking Security. Available online: http://nss.kaist.ac.kr/papers/sosr-demos15-final3.pdf (accessed on 8 September 2016).
25. Al-Shaer, E.; Al-Haj, S. FlowChecker: Configuration Analysis and Verification of Federated OpenFlow Infrastructures. In Proceedings of the 3rd ACM Workshop on Assurable and Usable Security Configuration, Chicago, IL, USA, 4–8 October 2010; pp. 37–44.
26. Bifulco, R.; Karame, G. Towards a Richer Set of Services in Software-Defined Networks. In Proceedings of the NDSS Workshop on Security of Emerging Technologies (SENT), San Diego, CA, USA, 23–26 February 2014.

27. Feng, T.; Bi, J.; Hu, H. OpenRouter: OpenFlow Extension and Implementation Based on a Commercial Router. In Proceedings of the 19th IEEE International Conference on Network Protocols (ICNP), Vancouver, AB, Canada, 17–20 October 2011; pp. 141–142.
28. Kamisiński, A.; Fung, C. FlowMon: Detecting Malicious Switches in Software-Defined Networks. In Proceedings of the 2015 Workshop on Automated Decision Making for Active Cyber Defense, Denver, CO, USA, 12–16 October 2015; pp. 39–45.
29. Khurshid, A.; Zou, X.; Zhou, W.; Caesar, M.; Godfrey, P.B. Veriflow: Verifying Network-Wide Invariants in Real Time. In Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13), Lombard, IL, USA, 2–5 April 2013; pp. 15–27.
30. Lara, A.; Ramamurthy, B. Opensec: A Framework for Implementing Security Policies Using Openflow. In Proceedings of the Global Communications Conference (GLOBECOM), Austin, TX, USA, 8–12 December 2014; pp. 781–786.
31. Matias, J.; Garay, J.; Mendiola, A.; Toledo, N.; Jacob, E. FlowNAC: Flow-Based Network Access Control. In Proceedings of the Software Defined Networks (EWSDN), Third European Workshop, Budapest, Hungary, 1–3 September 2014; pp. 79–84.
32. Mendonca, M.; Seetharaman, S.; Obraczka, K. A Flexible in-Network IP Anonymization Service. In Proceedings of the IEEE International Conference Communications (ICC), Ottawa, ON, Canada, 10–15 June 2012; pp. 6651–6656.
33. Wang, H.; Xu, L.; Gu, G. Of-Guard: A Dos Attack Prevention Extension in Software-Defined Networks. In Proceedings of the Open Network Summit (ONS), Santa Clara, CA, USA, 3–5 March 2014.
34. Canini, M.; Venzano, D.; Perešíni, P.; Kostić, D.; Rexford, J. A NICE way to test OpenFlow applications. In Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12), San Jose, CA, USA, 25–27 April 2012; pp. 127–140.
35. Fichera, S.; Galluccio, L.; Grancagnolo, S.C.; Morabito, G.; Palazzo, S. OPERETTA: An OPEnflow-based REmedy to mitigate TCP SYNFLOOD Attacks against web servers. *Comput. Netw.* **2015**, *92*, 89–100. [CrossRef]
36. Hong, S.; Xu, L.; Wang, H.; Gu, G. *Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures*; NDSS: National Down Syndrome Society: New York, NY, USA, 2015.
37. Hu, H.; Han, W.; Ahn, G.J.; Zhao, Z. FLOWGUARD: Building Robust Firewalls for Software-Defined Networks. In Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, Chicago, IL, USA, 17–22 August 2014; pp. 97–102.
38. Matias, J.; Tornero, B.; Mendiola, A.; Jacob, E.; Toledo, N. Implementing Layer 2 Network Virtualization Using OpenFlow: Challenges and Solutions. In Proceedings of the Software Defined Networking (EWSDN), European Workshop, Darmstadt, Germany, 25–26 October 2012; pp. 30–35.
39. Porras, P.; Shin, S.; Yegneswaran, V.; Fong, M.; Tyson, M.; Gu, G. A Security Enforcement Kernel for OpenFlow Networks. In Proceedings of the First Workshop on Hot Topics in Software Defined Networks, Helsinki, Finland, 13–17 August 2012; pp. 121–126.
40. Yao, G.; Bi, J.; Xiao, P. Source Address Validation Solution with OpenFlow/NOX Architecture. In Proceedings of the 2011 19th IEEE International Conference Network Protocols (ICNP), Vancouver, BC, Canada, 17–20 October 2011; pp. 7–12.
41. Suh, J.; Choi, H.; Yoon, W.; You, T.; Kwon, T.; Choi, Y. Implementation of Content-Oriented Networking Architecture (CONA): A Focus on DDoS Countermeasure. In Proceedings of the 1st European NetFPGA Developers Workshop, Cambridge, UK, 9–10 September 2010; pp. 1–5.
42. Tootoonchian, A.; Ganjali, Y. HyperFlow: A Distributed Control Plane for OpenFlow. In Proceedings of the Internet Network Management Conference on RESEARCH on Enterprise Networking, San Jose, CA, USA, 27 April 2010; p. 3.
43. Wang, H.; Xu, L.; Gu, G. FloodGuard: A Dos Attack Prevention Extension in Software-Defined Networks. In Proceedings of the 45th Annual IEEE/IFIP International Conference Dependable Systems and Networks (DSN), Rio de Janeiro, Brazil, 22–25 June 2015; pp. 239–250.
44. Wen, X.; Chen, Y.; Hu, C.; Shi, C.; Wang, Y. Towards a Secure Controller Platform for Openflow Applications. In Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, Hong Kong, China, 12–16 August 2013; pp. 171–172.

45. Liyanage, M.; Ylianttila, M.; Gurtov, A. Securing the Control Channel of Software-Defined Mobile Networks. In Proceedings of the 2014 IEEE 15th International Symposium World of Wireless, Mobile and Multimedia Networks (WoWMoM), Sydney, Australia, 16–19 June 2014; pp. 1–6.
46. Sherwood, R.; Gibb, G.; Yap, K.K.; Appenzeller, G.; Casado, M.; McKeown, N.; Parulkar, G.M. Can the production network be the testbed? *OSDI* **2010**, *10*, 1–6.