A Comprehensive Security Architecture for SDN

Zhiyuan HU Mingwen WANG* Xueqiang YAN Yueming YIN Zhigang LUO
Alcatel-Lucent Shanghai Bell Co., Ltd. Shanghai, China
{Zhyuan.Hu, Xueqiang.Yan, Emile.Yin, Zhigang.Luo }@alcatel-sbell.com.cn
*Alcatel-Lucent, Antwerp, Belgium, Mingwen.Wang@alcatel-lucent.com

***Abstract*—SDN enables the administrators to configure network resources very quickly and to adjust network-wide traffic flow to meet changing needs dynamically. However, there are some challenges for implementing a full-scale carrier SDN. One of the most important challenges is SDN security, which is beginning to receive attention. With new SDN architecture, some security threats are common to traditional networking, but the profile of these threats (including their likelihood and impact and hence their overall risk level) changes. Moreover, there are some new security challenges such as bypassing predefined mandatory policies by overwriting flow entries and data eavesdropping by inserting fraudulent flow entries. This paper is to design open-flow specific security solutions and propose a comprehensive security architecture to provide security services such as enforcing mandatory network policy correctly and receiving network policy securely for SDN in order to solve these common security issues and new security challenges. It can also help the developers to implement security functions to provide security services when developing the SDN controller.**

Keywords—SDN; Security Architecture

I. INTRODUCTION

Software-Defined Networking (SDN) is a network architecture framework that decouples the network control from underlying network forwarding and transporting infrastructure so as to enable the network control to become directly programmable [1]. This decoupling allows the underlying forwarding and transporting infrastructure to be abstracted for applications and network services. Through software-based SDN controllers, the network is centrally managed so that administrators could configure network resources very quickly and dynamically adjust network-wide traffic flow to meet changing needs. Such a controller serves as a type of operating system for SDN network. By separating the control plane from the network hardware and running the control plane instead as software, the controller facilitates automated network management, as well as integration and administration of applications and network services. The advantages of SDN as above have already been proven in academia and industry. However, there are some challenges for implementing a full-scale carrier SDN. Some of such challenges have been presented in [2]. One of the most important challenges is SDN security, which is beginning to receive attention from academia and industry.

Currently most research work focuses on exploiting SDN to improve network security by using monitoring systems with inserting security policies to dynamically detect and mitigate suspicious traffic during live network operations. Mehdi et al. [3] propose the value of using SDN to provide intrusion detection in a Home Office/Small Office environment. OpenSAFE [4] uses its ALARMS policy language to manage the routing of traffic through network monitoring devices. A Distributed DoS (DDoS) detection method based on several traffic flow features to identify abnormal flows is presented in [5]. CloudWatcher [12] controls network flows to guarantee that all necessary network packets are inspected by some security devices with introducing SDN in the cloud. This framework automatically detours network packets to be inspected by pre-installed network security devices. FleXam [13] enables the controller to access packet-level information for the detection and mitigation of botnet and worm propagation. FRESCO [9] presents an OpenFlow-enabled security application development framework designed to facilitate the rapid design and development of security modular for the detection and mitigation of network threats.

There are some researches on security issues of SDN itself [15][16][17]. The use of transportation layer security (TLS) to secure the communication between the controllers and their switches is described in OpenFlow™ protocol [6]. However, this security feature is optional, and how to implement TLS (e.g., the version of TLS and cipher suites) is not detailed. The lack of TLS use could lead to fraudulent flow entry insertion, flow entry modification and DoS attacks. NICE [7] uses symbolic execution to verify conformance of OpenFlow applications. However, such path exploration approaches do not scale well for large applications. FlowChecker [10] exploits FlowVisor [14] to detect inconsistencies in policies from multiple applications or installed across multiple devices. FLOVER [8] uses assertion sets and modulo theories to check whether new flow entries conflict with predefined non-bypass policies. FortNOX [11] handles possible conflicts of flow entry insertion with role-based OpenFlow application authentication through digital signatures. Although it provides numerous components to enforce security, the authors feel that much work is still needed to offer a comprehensive suite of applications.

This paper is to analyze new security challenges with SDN and design security solutions to solve them. A comprehensive security architecture is proposed to guide the developer to implement security functions and provide security services for SDN.

The rest of the paper is organized as follows. In section II, potential security issues and challenges of SDN itself are

978-1-4799-1866-9/15/$31.00 ©2015 IEEE

analyzed. In section III, security solutions are designed, and a comprehensive SDN security architecture is proposed. Evaluation of security solutions and the security architecture is discussed in section IV. Section V concludes the paper.

II. SECURITY CHALLENGES WITH SDN

In general, SDN security threats are common to traditional networking, but the profile of these threats (including their likelihood and impact and hence their overall risk level) changes with new SDN architecture. With a centralized SDN controller, the impact of DoS/DDoS attacks from southbound interfaces can be worse than that directed against a single router. A spoofed SDN controller could potentially take control of the whole network in which switches are managed by this spoofed controller, while a spoofed router could only attack the traffic routed through it.

There are some new security challenges with SDN. Applications are programs that explicitly, directly, and programmatically communicate their network requirements and desired network behavior (i.e. network policies) to the controller via northbound interfaces. The controller converts these network policies into flow entries and inserts them into the flow table. However, currently a flow entry in OpenFlow™ flow table does not distinguish the application generating the new flow entry from another application generating the old flow entry. So, it's possible that a new network policy generated by a general application can replace a non-bypass security policy predefined by the security administrator. In Fig. 1, security administrator proactively configures a non-bypass security policy as following: the packets must be sent to the Firewall for packet scan detection if these packets are delivered from Host A (172.0.0.1) to Host B (172.0.0.2), as data transportation path_1 in green *(the path_1: Host A -> SDN_Switch_1 -> SDN_Switch_2 -> Firewall -> SDN_Switch_3 -> Host B)*. Sometime later, the application App_X needs the shortest path for data transportation with low delay and generates the policy as following: the shortest transportation path will be selected if these packets are delivered from Host A (172.0.0.1) to Host B (172.0.0.2), as data transportation path_2 in brown *(the path_2: Host A -> SDN_Switch_1 -> SDN_Switch_3 -> Host B)*. According to the format of the flow entry in OpenFlow™ flow table, the controller will replace the former non-bypass security policy with the later shortest path policy. This offends security administrator intention and the mandatory security policy will be passed by.

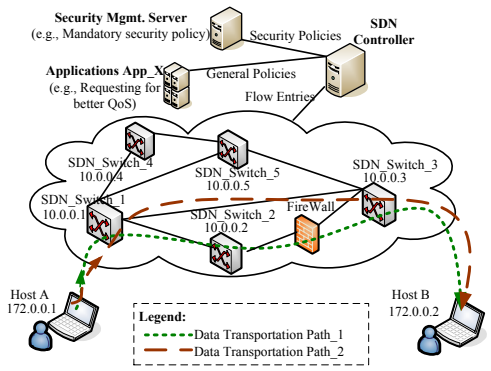

Figure 1. Bypassing a predefined mandatory policy

Another security challenge is that an attacker may hijack an application server and insert fraudulent flow entries to make data eavesdropping attacks. In Fig. 2, the attacker hijacks an application server to generate the policy as following: the packets are copied and forwarded to the attacker with IP address 192.0.0.10 if these packets are delivered from Host A (172.0.0.1) to Host B (172.0.0.2). This policy is converted into a flow entry then inserted into SDN_Switch_1 flow tables through the message *OFPT_Flow_MOD* from the controller. In this way, the attacker can easily intercept the packets from Host A to Host B.

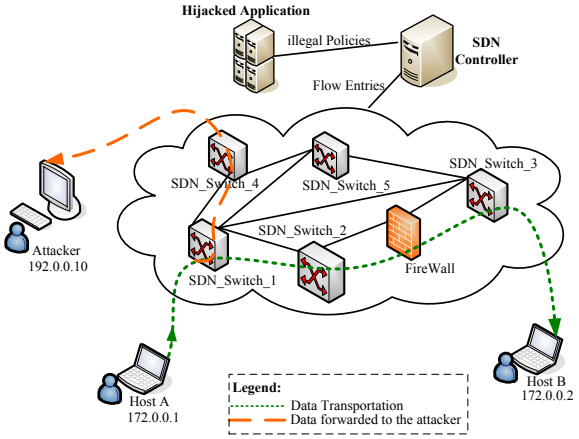

Figure 2. Data eavesdropping attacks by inserting fraudulent flow entries

SDN-oriented network is self-managed and self-controlled. It's important for the controller to detect attacks and mitigate them automatically. Currently OpenFlow™ protocol [6] as one of SDN southbound interface implementations only supports packet header scan detection. In the switches, flows are forwarded or dropped by matching packet header with Match Fields in OpenFlow™ flow tables. In order to detect some attacks such as worm or Trojan or spam information and mitigate them, packet data scan detection has to be done. However, the current OpenFlow™ protocol does not support packet data scan detection.

This paper is to design security solutions and propose a comprehensive security architecture to provide special security services (e.g., enforcing mandatory network policy correctly, supporting packet data scan detection, receiving network policy securely, converting network policy into flow entries precisely, etc.) and common security services (e.g., authentication, authorization, confidentiality, integrity, software patches, trusted computing) for SDN so that both common security issues and new security challenges as above can be solved.

III. A COMPREHENSIVE SECURITY ARCHITECTURE FOR SDN

When designing security architecture for SDN, existing security mechanisms are exploited to provide countermeasure for common security issues discussed in section 2. For those new security challenges like bypassing predefined mandatory policy, inserting fraudulent flow entries and packet data scan detection, new solutions are proposed and elaborated as below.

A. Solutions for new security challenges

1) Fine-grained naming scheme for the flow entry

Currently the components "match fields" and "priority" are taken together to identify a unique flow entry in the OpenFlow™ flow table. In this way, a flow entry in the flow table does not distinguish the application generating the new flow entry from another application generating the old flow entry. So, it is possible for an application server to create a new flow entry to replace the flow entry which reflects a mandatory policy predefined by a security administrator.

In order to design a fine-grained naming scheme for the flow entry, two new features will be added. One is the role of policy creator, the other is the security privilege level of role.

The role of policy creator is used to define the role of the administrator/application that creates a given network policy. The role of creator may be a security administrator, a general administrator, a user, or a guest.

The security privilege level of role is used to specify different security privilege levels for different roles of policy creators. Hierarchical levels from the highest to the lowest may be L5, L4, L3, L2, L1, and L0. The role with the relatively higher security privilege level is given more rights to access the SDN controller. For example, the policy created by the creator with relatively low security privilege level is replaced by the policy created by the creator with a higher level. In one illustrative embodiment, security privilege levels of roles can be set as follows: security administrator - L5 (highest); general administrator - L4; user - L2; and guest - L1. It is to be understood that, in this illustrative embodiment, there is no dedicated role for security privilege levels L3 and L0. However, the quantity and assignment of security privilege levels is open to the developer and can be instantiated based on the specific needs of the SDN network design. Overwriting policy by role level as above may not work for heterogeneous networks, since the definition of role level for those networks may be different.

In order to improve system performance without impacting the policies needing quick response, the third new feature i.e. the characteristic of network policy is added to describe a flow entry. The characteristic of network policy is used to describe whether new network policies are real-time policies or can be put into operation with some delay. For a real-time policy, the policy should be converted into flow entries, updated in flow tables, and then synchronized with the SDN switches without any delay. These real-time policies may react against security attacks detected online by security appliances (e.g., firewall, DPI, and IDP). For those policies which can be operated with some delay, they are converted into flow entries, updated in flow tables, and then synchronized with the SDN switches in batches periodically in order to improve system performance.

With above three new additional features to describe a flow entry defined in SDN controller, the new flow entry can be inserted into the flow table correctly since the flow entry created by the security administrator with a high security privilege level cannot be replaced by the new flow entry created by an application server with a low security privilege level. In this way, mandatory network policies will not be overwritten and bypassed.

The flow entry in the flow table of switches is not impacted since these three features are stored in the controller to help the controller make decision when updating flow entries.

2) Packet data scan detection

Packet data scan detection has to be supported in order to detect and mitigate some attacks like worm and Trojan and spam information. The administrator configures the policies to randomly detect some packets of the flow instead of all the packets of the flow for higher performance. There is one possible schema of packet data scan detection: to select the first m consecutive packets from each flow for packet data scan detection. This schema can be designed for all flows or for only flows meeting some conditions, such as packets from a certain source IP address and/or to a certain destination.

OpenFlow™ protocol [6], as one of SDN southbound interface implementation, may be extended in order to support packet data scan detection. Firstly, two more additional features may be added into the format of the flow entry. These updates have to be reflected into both the controller and switches. One of them is the schema which describes the schema of packet data scan detection. The other is the condition which describes the flows meets some conditions configured by the administrator or applications. Secondly, an optional action (OFPAT_DETECTION) should be added in section of "5.12 Actions" of [6] as following texts in *italic: Optional Action: the Detection action forwards a packet to a specified OpenFlow™ port then to security appliances (e.g., FW, IDP, DPI, etc) for further data scan detection.* This new action is similar to the action OFPAT_OUTPUT in OpenFlow™ protocol. Finally, action structures should be updated in section "7.2.4 Action Structures" of [6] as below with texts in *italic*:

```
enum ofp_action_type {
OFPAT_OUTPUT = 0,                   /* Output to switch port. */
OFPAT_DETECTION = XX (a given number),  /*Output to switch port */
OFPAT_COPY_TTL_OUT = 11,            /* Copy TTL "outwards" – from
                                       next-to-outermost to outermost */
OFPAT_COPY_TTL_IN = 12,             /* Copy TTL "inwards" – from
                                       outermost to next-to-outermost */
OFPAT_SET_MPLS_TTL = 15,            /* MPLS TTL */
OFPAT_DEC_MPLS_TTL = 16,            /* Decrement MPLS TTL */
OFPAT_PUSH_VLAN = 17,               /* Push a new VLAN tag */
OFPAT_POP_VLAN = 18,                /* Pop the outer VLAN tag */
OFPAT_PUSH_MPLS = 19,               /* Push a new MPLS tag */
OFPAT_POP_MPLS = 20,                /* Pop the outer MPLS tag */
OFPAT_SET_QUEUE = 21,    /* Set queue id when outputting to a port */
OFPAT_GROUP = 22,                   /* Apply group. */
OFPAT_SET_NW_TTL = 23,    /* IP TTL. */
OFPAT_DEC_NW_TTL = 24,    /* Decrement IP TTL. */
OFPAT_SET_FIELD = 25, /*Set a header field using OXM TLV format*/
OFPAT_PUSH_PBB = 26,    /* Push a new PBB service tag (I-TAG) */
OFPAT_POP_PBB = 27,    /* Pop the outer PBB service tag (I-TAG) */
```

```
OFPAT_EXPERIMENTER = 0xffff
};
A Detection action uses the following structure and fields:
/*Action structure for OFPAT_DETECTION which
sends packets out 'port'.*/
struct ofp_action_detection {
uint16_t type;          /* OFPAT_DETECTION. */
uint16_t len;           /* Length is 16. */
uint32_t port;          /* Output port. */
uint16_t schema;        /* One possible schema is: to select the first m
                           consecutive packets from each flow. */
uint32_t condition;     /* One possible condition: packets
                        of the flow to a certain destination . */
};
OFP_ASSERT(sizeof(struct ofp_action_output) == 10);
```

3) Network API authorization

In order to protect an application server from being hijacked by an attacker to insert fraudulent flow entries to make attacks, network API authorization has to be supported. The authorization framework for network API can refer to IETF OAuth2.0 [18] and its implementation based on RESTful API can refer to OMA Autho4APIv1.0 [19]. With implementing authorization for northbound interfaces of SDN controller, it is possible to mitigate some attacks such as inserting fraudulent flow entry from hijacked application servers, and spoofing the SDN controller.

B. Overview of SDN Security Architecture

According to above basic ideas description, an overview of SDN security architecture is showed in Fig. 3.

The SDN controller receives network policies from the logical functions like automated security management and general applications and load balance via northbound interface, in addition to the administrator management via management interface. In general, the management interface is an internal interface to SDN controller and may be more trustable than northbound interface. Two new logical functions designed by this paper are automated security management which supports SDN controller detecting and mitigating security attacks automatically, and network security monitor which supports administrators in monitoring security status of the entire SDN network.

The SDN controller synchronizes flow entries with switches through southbound interface to route the flows correctly. The SDN controller may exchange some control information of the switches on the border with another SDN controller through east-west interface.

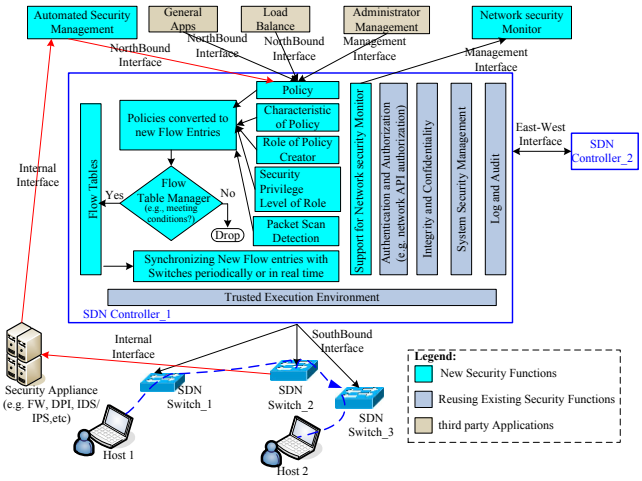

Figure 3: A comprehensive Security Architecture for SDN

The SDN controller is designed to provide security services mentioned in section II to solve common security issues with some changes and to handle some new security challenges. To provide such security services, the SDN controller comprises two kinds of modules. One kind of modules is new and designed or modified by this paper, such as a policy module, a characteristic of policy module, a role of policy creator module, a security privilege level of role module, a packet scan detection module, a policy conversion module, a flow table manager, flow tables, a flow entry synchronization module, a network security monitor support module. Another kind of modules is to reuse existing security mechanisms, such as an authentication and authorization module (with supporting network API authorization), an integrity and confidentiality module, a system security management module, a log and audit module, and a trusted execution environment.

The policy module is configured to receive one or more policies (including network policies, especially security policies) from its northbound interface (i.e., from automated security management module, general applications module, and Load Balance module) and its management interface (i.e., Administrator Management module). In order to guarantee that received policies are secure, policy creators (e.g., automated security management module, general applications, load balance module, and administrator management module) are authenticated and authorized by authentication and authorization module of the SDN controller. Moreover, confidentiality and integrity checks are provided by integrity and confidentiality module for these policies in transportation.

The characteristic of policy module is configured to specify whether or not the policy will be sent from the SDN controller to the SDN switches in real-time or periodically. For example, the attribute may specify an urgent condition, a general condition, or a default condition. The attribute urgent specifies that this policy has to be synchronized with the switch in real-time without delay. For instance, the policy may be a security countermeasure to stop or to mitigate security attacks detected by firewall or DPI functionality of a security appliance.

The role of policy creator module is configured to specify the role of a policy creator. As mentioned above, policy

creators may be automated security management module, one or more general applications, load balance module, and administrator management module. Thus, the role of creator may be set as a security administrator, a general administrator, a user, or a guest. In one example, the role of the automated security management module is a security administrator, the role of the load balance module is a general administrator, the role of the one or more general applications is user, and the role of one or more third party applications (not expressly shown) is guest.

The security privilege level of role module is configured to specify the security privilege level given to each role set by module. As mentioned above by way of example, the attribute for the security privilege level of roles can be set to one of the security privilege levels L5, L4, L3, L2, L1, and L0, with L5 being the highest security privilege level and L0 being the lowest security privilege level. In one illustrative embodiment, security privilege levels of roles can be set as follows: security administrator - L5 (highest); general administrator - L4; user - L2; and guest - L1. The role with higher security privilege level is given more rights to access the SDN controller. For example, the policy created by the creator role of guest can be replaced by the policy created by the creator role of security administrator.

The packet scan detection module is configured to specify the detection scheme for those flows meeting some conditions. As for packet header scan detection, there is no impact on OpenFlow™ protocol. But for packet data scan detection, two more additional features may be added into the format of the flow entry. One is the schema which describes the schema of packet data scan detection, the other is the condition which describes the flows meets some conditions configured by the administrator or applications. Moreover, an optional action (OFPAT_DETECTION) should be added and reflected into OpenFlow™ protocol.

The policy conversion module is configured to convert the policy received from the northbound interface and/or the management interface into new flow entries combined with characteristic of policy, role of policy creator, security privilege level, the schema and the condition.

The flow table manager is configured to manage searching, adding, updating, and deleting of flow entries in flow tables. The flow table manager confirms that new flow entries are correctly inserted into the flow tables. For example, for a specified data flow, the flow table manager checks if a new flow entry is in conflict with an old (previous) flow entry before inserting new flow entries. If yes, only new flow entries set by a creator with a relatively high security privilege level are permitted to replace the conflicting old flow entries set by a creator with a relatively low security privilege level.

The flow tables store flow entries for the SDN switches. The format of the flow entries in SDN controller are modified with adding five new features i.e. characteristic of policy, role of policy creator, security privilege level, the schema and the condition. These new features help SDN controller to convert network policies into flow entries precisely and insert them to the flow tables in SDN controller correctly. The first three new features are not reflected into the flow entries of the flow tables in switches since there is no change for OpenFlow™ protocol. The last two features impacts OpenFlow™ protocol and reflected into the flow entries of the flow tables in switches.

The flow entry synchronization module synchronizes new flow entries with the SDN switches periodically or in real-time depending on the attribute specified by characteristic of policy module. In order to improve the performance of communication between the SDN controller and a given SDN switch, flow entries with attributes “general” or “default” are synchronized periodically. However, flow entries with attribute “urgent” are pushed to switches as soon as those flow entries are inserted into flow tables.

The network security monitor support module is configured to support administrators in monitoring the entire SDN network, particularly with respect to security issues. For example, assume a security administrator wants to know the details regarding a security policy for a specified data flow through network security monitor module. The network security monitor module sends a request to network security monitor support module. The network security monitor support module retrieves related information (e.g., flow entries, flow entries generators, security policies, etc.) from flow tables, analyzes the retrieved information and responds to network security monitor module. The network security monitor module generates a report and displays the report to the security administrator.

The authentication and authorization module is configured to provide authentication and authorization services for the SDN network. For example, The authentication and authorization module provides for authorization and mutual authentication: between the SDN controller and each of switches over the southbound interface; between the SDN controller and the one or more general applications over the northbound interface; between the SDN controller and the administrator management module over the management interface; and between SDN controller and other SDN controllers over east-west interface. There are some available security mechanisms including, but are not limited to, OAuthv2.0 [18], Autho4API [19], transport layer security (TLS), access control lists (ACL), and role based access control (RBAC). In this way, the possibility of the attacks such as DoS/DDoS attacks from southbound interface and spoofing SDN controller is reduced.

The integrity and confidentiality module is configured to provide security functions including encryption/decryption, digital signature, and integrity validation for the communication via northbound interface, southbound interface, management interface and east-west interface. Available security mechanisms are transport layer security (TLS) and Internet Protocol Security (IPsec).

The system security management module is configured to provide functions including, but not limited to, installation of software patches, denial of service (DoS) attack detection and mitigation, and anti-virus and malware detection and mitigation.

The log and audit module is configured to provide functions including, but not limited to, capturing administrator activity (e.g., login activity, file accesses, commands run, etc.)

and capturing security policy creator activity (e.g., the time to set security policy, the creator, the role of the creator, etc.).

The trusted execution environment is configured to ensure that sensitive data (e.g., SDN controller operating system, software, flow tables, a characteristic of the security policy, a role of the creator of the security policy, a security privilege level of the role of the creator of the security policy, authentication credentials, and authorization policies) is stored, processed and protected in a trusted environment.

C. *Use Case: automatic and prompt reaction to new security attacks*

Security appliances (e.g., firewall, intrusion detection system (IDS), intrusion prevention system (IPS), deep packet inspection (DPI)) deployed in a data forwarding plane of a network can detect security attacks and report them to automated security management functions. In order to react against these attacks, automated security management functions immediately define countermeasures which will be converted into flow entries in the SDN controller then synchronized with SDN switches. Thereafter, those attack packets are dropped or forwarded to a security appliance for further inspection before forwarding them out of the network. In this manner, the SDN controller can automatically react to security attacks and mitigate them.

According to SDN security architecture in Fig. 3, flows of automatic and prompt reaction to new security attacks can be described as following.

Step 1: detecting security attacks and generating security policies.

The security administrator configures security policies to detect attacks. According to predefined security policies, switches forward the packets of the flows meeting the conditions to security appliances for packet scan detection. Once suspicious packets are detected, security appliances report the characteristics of these suspicious packets to automated security management module. After analyzing these suspicious packets, automated system management module generates countermeasures and security policies based on knowledge learned from its analysis

Step 2: sending security policies to SDN controller securely.

Before automated security management module sends those policies to the SDN controller via northbound API, network API authorization mechanisms referring to IETF OAuth [18] and OMA Autho4API [19] should be performed. Then automated security management module sends those security policies to the SDN controller with confidentiality (encryption) and integrity protection (message/data signature) referring to security mechanisms such as TLS. After that, SDN controller checks if automated security management module has the right to set policies for the SDN network. For example, this policy is dropped if automated security management module is in a black list or has no such right to set policies. It's possible that the module is hijacked by the attackers.

Step 3: converting security policies into new flow entries precisely

The SDN controller receives these security policies and converts them into new flow entries according to OpenFlow™ protocol [6] with the following additional features:

- Characteristic of policy: real-time (meaning this policy has to be applied in the network without any delay);
- Role of the policy creator: security administrator; and
- Security privileged level of role: L5 (the highest level).

Thus, the current flow entry format of the flow table in the SDN controller is extended in order to reflect at least the above three features. However, there is no change in the format of flow table in any of the SDN switches.

If security policies include packet data scan detection, two more additional features may be added into the format of the flow entry. One is the schema which describes the schema of packet data scan detection; the other is the condition which describes the flows meets some conditions configured by the administrator or applications. These updates have to be reflected into the flow entries of flow tables in both the controller and switches.

Step 4: inserting new flow entries to flow tables in SDN controller without conflict

The flow table manager is triggered to insert new flow entries to flow tables. For each flow entry, the flow table manager check if there is an existing or available flow entry in the current flow tables for the specified data flow (for example, identified by source Internet Protocol (IP) address/port and destination IP address/port) in flow tables. Then, flow table manager checks and determines whether the security privilege level of the new flow entry is higher than that of the old flow entry. If the security privilege level of the new flow entry is higher than that of the old one, the new flow entry will be inserted into flow tables and overwritten the old one.

Step 5: synchronizing new flow entries with SDN switches timely and securely

For a real-time policy, corresponding flow entries are synchronized with the SDN switches without delay. For those policies which can be operated with some delay, they are synchronized with the SDN switches in batches periodically in order to improve system performance.

Before receiving new flow entries, each SDN switch mutually authenticates with the SDN controller. Then SDN controller sends flow entries to each switch with confidentiality (encryption) and integrity protection (message/data signature) referring to security mechanisms such as TLS in OpenFlow™ protocol [18]. So, SDN switches obtain new flow entries and insert them into flow tables.

According to these new flow entries reflecting new security polices, SDN switches deal with (e.g., dropping, forwarding) those packets meeting the conditions accordingly. Therefore, those attacks detected by security appliances can be prevented or mitigated automatically.

IV. EVALUATION

A. Security

The following main goals are to be achieved when developing and deploying SDN according to the proposed security architecture.

- **Goal 1:** comprehensive security architecture for SDN as a development guide

A comprehensive security architecture in Fig. 3 is designed for SDN controller to provide special security services (e.g., enforcing mandatory network policy correctly, supporting packet data scan detection, receiving network policy securely, converting network policy into flow entries precisely, etc.) and common security services (e.g., authentication, authorization, confidentiality, integrity, software patches, trusted computing) for SDN so that both common security issues and new security challenges can be solved. Moreover, it helps the developer to implement security functions when developing SDN controller.

- **Goal 2:** enforcing predefined mandatory network policies correctly

With a fine-grained naming scheme for the flow entry proposed in this paper, SDN controller can distinguish an old flow entry in the flow table generated by one application from the new flow entry generated by another application for the same specified flow. Then the old flow entry created by one application with a high security privilege level cannot be replaced by the new flow entry created by another application with a low security privilege level. In this way, mandatory security policies are overwritten and bypassed.

- **Goal 3:** Supporting packet data scan detection to mitigate some attacks like worm and Trojan

With the method of packet data scan detection designed in this paper, SDN controller can mitigate the attacks like worm and Trojan and spam information, in addition to those attacks by doing packet header scan detection. Thus, SDN controller can do well in detecting security attacks and mitigating them automatically.

- **Goal 4:** Receiving network policies from northbound API securely

After implementing network API authorization with reference to [18][19], SDN controller receives network policies from northbound API securely and correctly since only authorized application has the right to configure the network. So, the attacks of the application server hijacked by the attacker to insert fraudulent flow entries can be prevented or mitigated.

Some other goals are also to be achieved, such as southbound API security, trusted execution environment for SDN controller, installing patches securely, etc.

B. Feasibility and scalability

Fine-grained naming scheme for the flow entry is to help the controller manage the flow entry in the flow table precisely. Three new features are added to the format of the flow entry only in the flow table of the controller. With these three new features and some service logics, the controller can convert network policies into flow entries correctly then synchronize them to the flow table of switches without bypassing any mandatory policies. This fine granular naming scheme is independent of SDN southbound interface implementations and good for scalability.

A schema of packet data scan detection is to support mitigating some attacks like worm and Trojan. As one of SDN southbound interface implementation, OpenFlow™ Switch Specification [6] may be updated to support packet data scan detection. Two more additional features should be added into the format of OpenFlow™ flow entry. One is the schema, another one is the condition. An optional action i.e. Detection action and its corresponding action structures should be also added. These changes are not difficult to be implemented.

Existing security mechanisms IETF OAuth2.0 [18] and OMA Autho4APIv1.0 [19] are used for implementing network API authorization. Their feasibility and scalability have already been proven in other applications like GSMA RCS. Other security mechanisms such as TLS and trusted computing have already been used widely in industry.

V. CONCLUSION

With SDN, some security threats are common to traditional networking, but the profile of these threats (including their likelihood and impact and hence their overall risk level) changes. Moreover, there are some new security challenges such as bypassing predefined mandatory policies by overwriting flow entries and data eavesdropping by inserting fraudulent flow entries. Security solutions are designed in this paper to solve above security issues. Moreover, a comprehensive security architecture for SDN is also proposed to provide security services such as enforcing non-bypass network policies correctly, supporting packet data scan detection to mitigate some attacks like worms, and receiving network policies from northbound API securely, etc. It can also help the developer to implement security functions when developing the SDN controller. However, how to integrate this security architecture into current commercial networks such as carrier networks and mobile networks is not discussed. It needs further studies in the future. We also plan to submit our proposals to the ONF Security Group in the future.

REFERENCES

[1] "Software-Defined Networking (SDN) Definition", Open Networking Foundation. Available: https://www.opennetworking.org/sdn-resources/sdn-definition

[2] S. Sezer, S. Scott-Hayward, P. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for SDN? Implementation challenges for software-defined networks," Communications Magazine, IEEE, vol. 51, no. 7, 2013.

[3] S. A. Mehdi, J. Khalid, and S. A. Khayam, "Revisiting traffic anomaly detection using software defined networking," in Recent Advances in Intrusion Detection. Springer, 2011, pp. 161-180.

[4] J. R. Ballard, I. Rae, and A. Akella, "Extensible and scalable network monitoring using opensafe," Proc.INM/WREN, 2010.

[5] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in IEEE 35th Conference on Local Computer Networks (LCN). IEEE, 2010, pp. 408-415.

[6] "OpenFlow Switch Specification Version 1.4.0", Open Networking Foundation. Available: https://www.opennetworking.org/sdn-resources/onf-specifications/openflow

[7] M. Canini, D. Venzano, P. Peresini, D. Kostic, and J. Rexford, "A NICE way to test OpenFlow applications," in Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, 2012.

[8] S. Son, S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Model Checking Invariant Security Properties in OpenFlow." Available: http://faculty.cse.tamu.edu/guofei/paper/Flover-ICC13.pdf

[9] S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, and M. Tyson, "FRESCO: Modular composable security services for software-defined networks," in Proceedings of Network and Distributed Security Symposium, 2013.

[10] E. Al-Shaer and S. Al-Haj. Flowchecker: configuration analysis and verification of federated openflow infrastructures. In Proceedings of the 3rd ACM workshop on Assurable and Usable Security Configuration, 2010.

[11] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu, "A security enforcement kernel for OpenFlow networks," in Proceedings of the first workshop on Hot topics in software defined networks. ACM, 2012, pp. 121-126.

[12] S. Shin and G. Gu, "CloudWatcher: Network security monitoring using OpenFlow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?)," in 20th IEEE International Conference on Network Protocols (ICNP). IEEE, 2012, pp. 1-6.

[13] S. Shirali-Shahreza and Y. Ganjali, " Efficient Implementation of Security Applications in OpenFlow Controller with FleXam", in 21st IEEE Annual Symposium on High-Performance Interconnects. IEEE, 2013, pp. 49-54

[14] R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Flowvisor: A network virtualization layer," OpenFlow Switch Consortium, Tech.Rep, 2009.

[15] D. Kreutz, F. Ramos and P. Verissimo, "Towards secure and dependable software-defined networks", in Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking. ACM, 2013, pp. 50-60.

[16] S. Scott-Hayward, G. O'Callaghan and S. Sezer , "SDN security: A survey", Future Networks and Services (SDN4FNS), IEEE SDN, 2013, pp. 1-7.

[17] K. Benton, L. Jean-Camp, and C. Small, "Openflow vulnerability assessment", in Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking. ACM, 2013, pp. 151-152.

[18] D. Hardt, "The OAuth 2.0 Authorization Framework", October 2012,IETF RFC 6749, Available: http://www.ietf.org/rfc/rfc6749.txt

[19] "Authorization Framework for Network APIs", Open Mobile Alliance?, OMA-ERP-Autho4API-V1_0, Available: http://www.openmobilealliance.org/