Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/120401

New SDN-Oriented Distributed Network Security System

Fahad Nife[1,2], *Zbigniew Kotulski*[1] *and Omar Reyad*[3,*]

[1] Faculty of Electronics and Information Technology, Warsaw University of Technology, Poland
[2] Faculty of Science, Al-Muthanna University, Iraq
[3] Faculty of Science, Sohag University, Egypt

Received: 2 Mar. 2018, Revised: 20 Apr. 2018, Accepted: 24 Apr. 2018
Published online: 1 Jul. 2018

Abstract: Software-Defined Network (SDN) is a network technology aimed to open new possibilities in network management and orchestration. This is important in future (especially mobile) networks, where virtualization of resources and network functions is the basic paradigm. SDN has been proposed to programmatically control networks, facilitating deployment of new applications and services, as well as tuning network policy and performance. It represents an important change in the way networks are architected, built, and managed. In this new networking paradigm, a network control plane is physically decoupled from a forwarding plane and is directly programmable. In SDN networks, the control plane supports a logically centralized controller which has a global view of the entire network; it gathers information from the data plane to be processed by the management tasks which are implemented as applications running on the top of the controller. Based on the global view, these applications make packets processing decisions and distribute them to the data plane via the controller. However, security of such networks with their programmability and centralized points of control is not currently ensured on a sufficient level. In this paper, we present the concept of a new security system for SDN-based networks, which can be easily integrated with the existing network infrastructure as well as can provide security of all network components. It consists of two main subsystems: the network authentication and access control system to protect the network control and the distributed firewall system to protect data transmission. Such a system enables creating additional boundaries within the network to provide a multi-plane system of defense, solves the problem of a single point of failure, and makes it easy to protect the network from external attacks as well as from internal malicious users.

Keywords: SDN, Network Security, Stateful Firewall, 802.1x, Access Control Mechanism

1 Introduction

In todays modern business climate, computer networks become an essential element to provide important communication links for an organization to run its applications, deliver services, and become more competitive for data security mechanisms [1]. To meet modern IT requirements, a traditional network should overcome the substantial limitations in its architecture which include the complexity in addressing new business requirements, where the features are vendor-specific and implemented through proprietary commands, the inconsistent policies, where a manually configured or scripted configuration across hundreds or, maybe, thousands of network devices that make the policy change extremely complicated, and the inability to scale, where the static provisioning in the traditional network meet an increase in the number of endpoints and services, or the necessary bandwidth, which requires an authentic planning and redesigning the network. SDN is a new open technology in industry promising to solve known limitations in traditional networks and opening new frames of networks designing, configuring, controlling and operating [2].

The SDN architecture decouples the control plane from the forwarding (data) plane, logically centralizes the network intelligence, and abstracts the underlying network infrastructure from the applications, which lead to more innovations, greater scalability, and highly flexible solutions. The point of SDN is to give open interfaces that empower the ability to develop software that can control the network resources connectivity as well as traffic flow over them [3]. Nevertheless, the centralized hierarchy that poses a single point of failure and risk of DoS attacks, and the network programmability are attractive for attackers, so securing the SDN networks

* Corresponding author e-mail: ormak4@yahoo.com

is a real challenge and can be considered the key for success or failure of such technology.

In this paper, we extend our previous work in [4,5] and propose a solution that takes the advantages of the programmability and the centralized control with its global view, it consists of two main subsystems. The first subsystem tries to prevent any access attempts from unauthorized users by verifying a host identity upon connection to the network. Moreover, a different level of privilege is implemented for each host based on its authentication credentials. The second subsystem presents a reactive distributed stateful firewall solution, in which the security policy is centralized in the firewall application which runs on the top of the controller, and implemented by the underlying data plane to present distributed checkpoint that provides a defense-in-depth and can be considered the best solution for those networks with different levels of trust. The implementation of our proposed system does not need any new protocol supports or host configuration, as well as it does not enforce any change in terms of behavior or design for current OpenFlow-based SDN paradigm

The rest of the paper is organized as follows. The theoretical background of firewalls and 802.1x technologies are described in Section 2. In Section 3, we present the state of the art of current SDN security solutions, which are analyzed and compared later with our proposed solution. In Section 4, the overview of the proposed system's architecture is described. In Section 5, we showed the details of its implementation and the results of the performance tests while in Section 6, we conclude the paper with a summary of the proposed solution and its future enhancements.

2 Background

In the following, we relate our proposed system to other existing network solutions and give the overview of 802.1X architecture, and firewalls technologies.

2.1 802.1*x Architecture*

IEEE 802.1x [6] is a standard technology that defines a port-based network access control mechanism for wired Ethernet networks. The standard specification of 802.1x relies on the EAP protocol [7] to verify a user's identity, it includes three major components: supplicant (host), authenticator, and authentication server. It defines an EAP over LAN (EAPOL) protocol to transport the EAP messages between the supplicant and the authenticator as shown in Figure 1.

To ensure that only authorized persons can access the network, all traffic is blocked except EAPOL. Once the user identity is successfully verified, his/her other traffic is permitted. In the 802.1x specification, EAP gives a way for the supplicant and the authentication server to negotiate an EAP authentication method. The EAP method is utilized to determine the credential type and how the credentials are provided from the supplicant to the authentication server. EAP is extensible by adding new EAP methods. Different EAP methods are available for use within IEEE 802.1x. The common EAP methods used in 802.1x are EAP-MD5, EAP-TLS [8], and PEAP-MSCHARPv2. The supplicant is an entity, which must provide the proper credentials to the authenticator to gain access to the network. Usually, it is an end-user workstation, but it can be another device seeking network services, such as a switch or router. The authenticator is an entity that controls the access to the network. Usually, it is a switch acting as a proxy between the supplicant and the authentication server. Based on the client identification data and the decision retrieved from the authentication server, it either grants or does not grant the client to access the network. The authenticator logically characterizes its ports as a controlled port and an uncontrolled port. The access to the whole network resources is provided by the controlled port, while uncontrolled port allows the access only for the EAPOL traffic between the supplicants and the authentication server. At the beginning, the authenticator opens only its uncontrolled port for EAPOL traffic. Once the supplicant is successfully authenticated, the controlled port is opened, so the supplicant can access different network resources. The authentication server is an entity that provides the authentication service to the authenticator; it validates the client and specifies whether the supplicant can access available services on the network. Typically, the authentication server supports RADIUS and EAP protocols. All authentication and authorization policies are located on the authentication server. RADIUS [9] or Diameter [10] protocols are used to transfer EAP frames between the authenticator and the authentication server. The authentication process started either by the supplicant requesting access to the network or by the authenticator when it detects a port status change or when it receives a packet with a source MAC address not included yet in its MAC address table. If the authenticator starts the negotiation, it sends an EAP-request/identity packet, but in case the supplicant starts the negotiation, it sends an EAPOL-start packet, to which the authenticator answers with an EAP-request/identity packet. The supplicant responds with an EAP-response/identity packet to the authentication server via the authenticator, in which the authenticator plays its role as a proxy between the supplicant and the authentication server. The authentication server responds it with an EAP-request packet to be forwarded to the supplicant via the authenticator. The supplicant now replays with an EAP-response. After that, the authentication server answers with either an EAP-success packet approving the client identity or with an EAP-reject packet to the supplicant, which means that its traffic will not forwarded as shown in Figure 2.

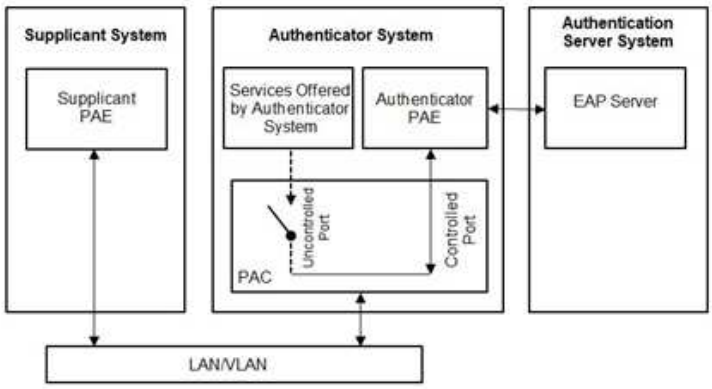

Fig. 1: IEEE 802.1x framework

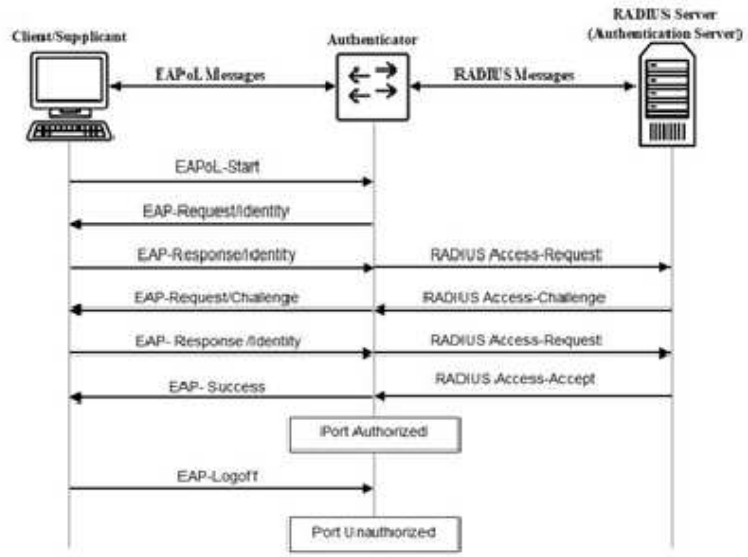

Fig. 2: Message exchanged during authentication

2.2 Firewall

In order to control the flow of information between designated sources and destinations devices, individuals and organizations, managing networks within information systems and between interconnected systems, commonly utilize flow control policies and enforcement mechanisms [11,12]. Flow control is based on the characteristics of the packet header and/or the information included. Enforcement occurs, for instance, in boundary protection devices, such as gateways or firewalls that employ sets of rules that restrict the system services by utilizing a packet-filtering capability based on header information, or message-filtering capability based on message content.

Firewalls are very important security mechanisms which are used to control the access from and to the network by filtering incoming or outgoing traffic, based on a set of filtering rules (firewall policies) reflecting and enforcing the organization's security policy. A Firewall is a checkpoint usually employed in the boundary between an authorized system and external environments to compare each and every incoming packet with its policy rules. It decides whether to allow or to deny a packet from passing through it. The firewall policy is a set of rules defining how the firewall handles the incoming and outgoing traffic for some specific IP addresses range of addresses, protocols, applications, and packet content [13]. According to the firewall policy, the packets flowing through a firewall can have one of three outcomes: accepted, dropped, or rejected.

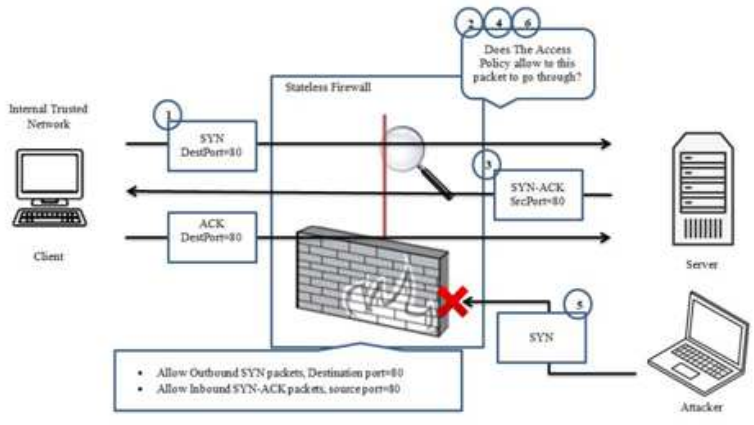

Fig. 3: Stateless firewalls

Firewalls can be implemented as software- or hardware-based and can be classified according to deferent criteria, such as their functionality or the layer (ISO/OSI reference model and TCP/IP model) where they are implemented (e.g., network, session, transport or application layer). For instance, stateless and stateful firewall are the most common firewalls that work on the network and transport layers. The stateless firewall enforces the network security policies by employing a filtering process for incoming and outgoing packets by comparing the IP address of the incoming packet against predetermined security rules set, to decide whether to pass or drop it [14]. It treats each packet separately, so it does not need to save the context ("state") about a packet already used for processing the next packet in the same flow, see Figure 3. The stateful firewall attempts to track the state of network connections and can tell if the packets are parts of current opened sessions originating within a trusted network. Generally, it maintains a table called the state table to store information about each active connection [15]. This state table holds entries that represent specific information (e.g., source and destination IP addresses, port numbers, flags, sequence and acknowledgment numbers) that uniquely identify the communication session, see Figure 4. For each received packet, the device will check the state table to see, if this packet belongs to any existing connection [16]. If it is, the action will be applied (accepted or denied) based on the policy associated to such a connection. In case it is the first packet of the connection, there are no matches with the state table entries, so it should be checked against the policy rules. According to the predefined decision, if it is accepted, a new entry will be created for this connection. But, if it is denied, creating a new entry for this packet is optional

3 SDN Security: State of The Art

In this section, we show an overview of some proposed solutions concerning the security of SDN networks. Our review is organized into two groups. The first group is focused on available efforts to provide network access control solutions. The second group presents the SDN-based firewalls approaches. Both groups are analyzed and compared with our proposed solution. One of the earliest solutions to provide network access control is Ethan [17]. The aim of Ethan project is to provide a user identity-based network access control system to the centralized controller architecture. It defines a global policy located on the centralized controller's side and relies on building a strong binding between the user's attributes, machine, and traffic (IP, MAC, and Ports). In order to authenticate and register different entities to the system, the switches use certificates to verify themselves to the controller through an SSL secure channel, the hosts authenticate using their previously registered MAC addresses, while the users authenticate by introducing their usernames and passwords to a website end to the Kerberos authentication server. Resonance [18] follows Ethan and extends it to provide the real-time monitoring ability. While both, Resonance and Ethane, are similar in the terms that they are employing the MAC address, and they redirected users to a website for authentication, their way of policy update is quite different (static vs. dynamic). Ethan and Resonance projects are followed by different solutions that adopt the IEEE 802.1x standard to provide an access control system for SDN networks. The authors in [19,20] adopt the IEEE 802.1x standard to provide a network and access control solution based on the tight bind of the host to the switch port for OpenFlow-based SDN network. Similar to Ethan and resonance projects, this solution argues that newly connected hosts are redirected to a website for submitting their credentials. The authentication website is integrated with the RADIUS client and ends to the RADIUS

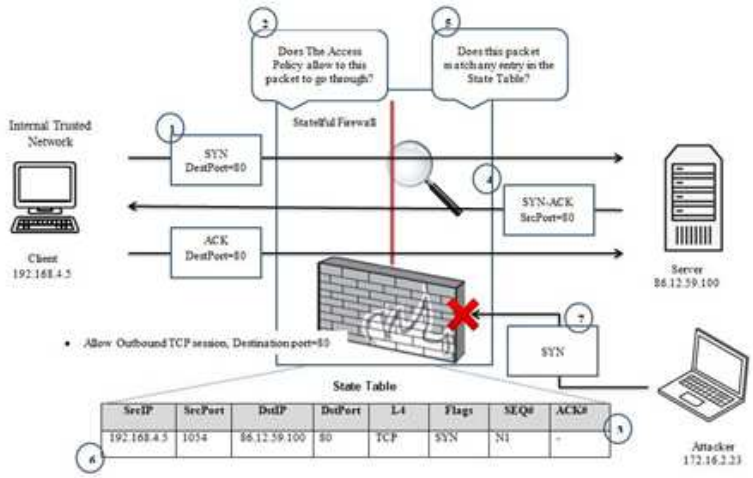

Fig. 4: Stateful firewalls

authentication server. The problem related with such solutions is that they require using the captive portal, which requires layer 3 configuration (e.g. DHCP) or even ephemeral private IP address assigned to the node to be able to access the website. Also, these approaches require the host to install an HTTPS-capable Web browser, which can be a real problem for virtual machines or other network devices that do not support web browser, such as printers, scanners, phones, or even some servers. AuthFlow [21] solution does not need any IP address assignment or layer 3 configurations; it tries to authenticate the hosts directly at layer 2 (based on MAC addresses). AuthFlow extends the RADIUS client authenticator hostapd [22] to establish secure SSL channel with the controller. FlowNAC [23] is a Flow-based network access control for SDN networks. It uniquely identifies the services and associates them with the flows; the decisions are based on, both, the user identity and the requested services. FlowNAC extends the policy to define the target service identifier, and implement a proactive manner of distributing the centralized policies. FlowNAC introduces the EAPoL_in_EAPoL encapsulation to help the user to access many services simultaneously. Such an encapsulation may enforce an additional overhead of updating the standard protocols, entities, and data models. FlowIdentity [24] adopt the IEEE 802.1x standard to provide a network access control solution and policy enforcement through a role-based firewall. The defined security policies are dynamically updated and directly enforced. While both, FlowNAC and FlowIdentity, argue to provide some service-based authentication, they differ only in the way that policy enforcement is achieved. Figure 5 shows the comparison between our proposed work and selected known SDN access control solutions.

Previous efforts to build firewall solutions to secure the SDN networks have been presented in many publications and projects deliverables. For instance, FortNox [25] is a security policy enforcement kernel implemented as a software extension to NOX controller. It prevents any application attempts to insert flow rules that may change the flow rules enforced by the security policy. FortNox is a real-time rule conflict detection engine that, using the rule-based authorization, either accepts or refuses a new rule. FRESCO [26] is a security application, which applies the OpenFlow framework consisting of reusable modular libraries that can be connected together to build more sophisticated security applications. FRESCO assists the developers to compose the necessary modules to produce the required security functions, like firewalls, IDS, and scan detections. FLOWGUARD [27] is a framework introduced to build robust SDN firewalls. FLOWGUARD provides a real-time violation resolution mechanism. Whenever the network states are updated, or the configurations are changed, it checks and compares the flow path spaces against the specified authorized space in the firewall, to detect firewall policy violations. FleXam [28,29] is a sampling extension method that gives the controller the ability to access packet-level information needed in different security applications. Easily the controller decides which part of the packet (e.g. headers/payload) and where they should be sent. FleXam provides two methods to sample the packet stochastically based on predetermined probability or deterministically based on a pattern. The stateful hardware firewall solution presented in [30], propose a prototype consisting of dump switches and the security rules specified in the controller. Unknown traffics are sent to the controller for inspection. The authors of [31] suggest a prototype for a reactive stateful firewall. It provides orchestration services for security policy management according to a holistic view, and reactive application that processes the state of the connection, In order to achieve the state aware, a generic

Publication	Authenticator	Layer Based	Behavior	Capabilities Required
Ethane	captive portal	Layer 3	Reactive	Web Browser Support
Resonance	captive portal	Layer 3	Reactive	Web Browser Support
AuthFlow	captive portal	Layer 3	Reactive	Web Browser Support
FlowNAC	hostapd	Layer 2	Proactive	EAPoL_in_EAPoL encapsulation
FlowIdentity	hostapd	Layer 2	Reactive	No
Proposed Solution	hostapd	Layer 2	Reactive	No

Fig. 5: Comparison of the proposed approach with other SDN-based access control solutions

algorithm is introduced for processing the Finite State Machine (FSM) of the TCP protocol. To summarize this draft analysis of security functions and performance, see Figure 6, which compares such functions.

The objective of our proposed solution is to secure the SDN network; it consists of two main subsystems. The network authentication and access control system to protect the network control by adopting the 802.1x standard with RADIUS server to authenticate, both, the users and hosts, it is based on the modified centralized authenticator which maintains a local database used to keep records of currently authenticated users as an introduced solution for the stateless property of the RADIUS protocol. The other subsystem is the distributed firewall to protect data transmission. Such a system creates an additional boundaries within the network to provide a multi-plane system of defense, solves the problem of a single point of failure, and protect the network from external attacks as well as from internal malicious users.

4 System Architecture

4.1 Architecture Overview

The main idea behind our proposed work is to protect the SDN network from external attacks and internal malicious users by developing the network access control and authorization mechanism as well as the distributed stateful firewall. Figure 7 shows the main components of the proposed system: the authentication server, the centralized authenticator, and the system functionality, that is executed by different modules running on the top of the SDN controller, which in turn maintain some introduced tables in the controller and on the data planes sides to provide the required security functions protecting the network. These tables help to reduce the lookup process time needed to handle a packet and improve the overall system performance. The Stable table implemented in the data plane provides the flow state-awareness and holds entries used to keep track for each specific flow.

4.2 Access Control and Policy Enforcement Subsystem

The authentication functionality is composed of five components: supplicant, OpenFlow-enabled switches, controller, authenticator, and the authentication server, see [5]. The authentication process of comparing the credentials, provided by the user with the predefined authorized users information database to validate the users, is done in the authentication server (RADIUS). If the credentials successfully match, the process is completed, and the user is granted authorization for the access. The permissions returned defined the network resources that the user can access. The authenticator is centralized and connected to all the OpenFlow-enabled switches. For security reasons, (to isolate the controller to protect the network from vulnerabilities and attacks like SYN flooding on the controller, which may lead to destructing the whole network [32,33], and to reduce the load on the controller), the authenticator is separated from the controller. The authenticator implements RADIUS client functionality, it forwards the message between the supplicant and the authentication server. The final decision, whether the user is authorized or not, is from the authentication server. It is forwarded to the controller via a secure encrypted channel, using SSL 3.0 standard, established between the controller and the authenticator. Reactive mode is implemented; it reacts to the new users access requests, approves the identity and accordingly installs the appropriate rules to manage the connection. However, two entries are installed in advance in the switch flow table to forward all EAPOL (Ethernet type set to 0x88E) to the authenticator and drop other packets.

Publication	Problem approached	Proposed solution	Behavior	Extra Memory	Capabilities Exploited	Performance
Collings, J. et al. [22]	Dynamic packet filtering	Stateful H/W Firewall	Reactive	No	Centralized control	High communication overhead
Zerkane, S. et al. [23]	DoS attack mitigation	Stateful/ Stateless	Reactive	Yes	Centralized control	High communication overhead
Suh, M. et al. [26]	Block unnecessary mitigation	Stateless firewall	Reactive	No	Centralized control	High Performance
Pena, J. G. et al. [27]	Packet Filtering	Stateless firewall	Proactive	Yes	Distributed control	High Performance
Jeong C. et al. [15]	Malicious traffic inspection	IDS	Proactive	Yes	Centralized control	High Performance
Proposed solution	Dynamic packet filtering	Stateful firewall	Reactive	Yes	Centralized control	High Performance

Fig. 6: Comparison of the proposed approach with other SDN based firewall solutions

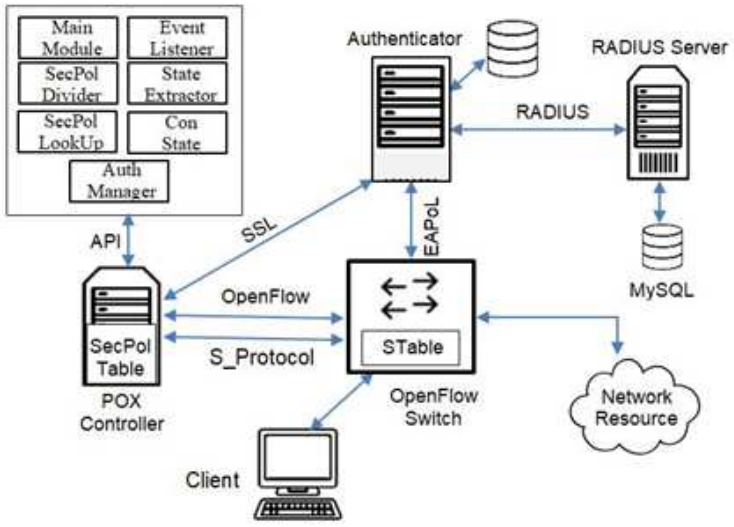

Fig. 7: The Architecture overview

Whenever a new 802.1x-aware workstation (supplicant) connects to the network, the authentication message starts exchanging between the supplicant and the authenticator. The supplicant starts the negotiation by sending EAPOL-start message to the edge switch, see Figure 8. Proactively, the switch is instructed to forward such frames to the centralized authenticator. The authenticator now will ask for the supplicant credentials using EAP-request/identity. The supplicant submits its credentials (username that uniquely defines this request for the client) using EAP-response/identity packet. After that, the authenticator decapsulates the message and checks it against the currently active user local database. If the provided credentials belong to a currently active user, the authenticator informs the controller application (auth_manager module); otherwise, it will forward it to the authentication server. The authentication server replays with RADIUS access/challenge, which the authenticator forwards to the client. The client now provides its identifying credentials using EAP-response/identity. The RADIUS server verifies these credentials, it either approves the identity of the user using an EAP-success (access-accept) packet or replies with EAP-reject (access-reject) packet, what it means the access is denied and the port is kept blocked. Accordingly, the authenticator will update its active-user database and will forward the authentication server's decision to the controller. Depending on the decision, the controller may install the new entry on the corresponding switch flow table and apply the predefined group policy for that client or may leave the ports blocked for that client.

Logically, authorization follows authentication. Despite the two terms are often used synonymously

(although they may often seem to be combined). However, they are two different processes. After the user is authenticated and his credentials are correctly matched, the authentication server returns a list of attribute-value pairs defining the user privileges to be forwarded to the authenticator, and finally to the Auth_manage module running over the controller, via the SSL channel. Such messages inform the controller about the authentication success and the identity of the supplicant (MAC address). The returned attributes may include service type, protocol type, access list to apply, or just static route to be installed in the OpenFlow table. Then, Auth_manage module translates this parameter into flow rules to be installed and applied to the corresponding switch port. Consequently, the switch can easily relate between the supplicant (MAC address) and its flow.

4.3 Distributed Stateful Firewall

Figure 4 shows an overview of the stateful firewall subsystem that is integrated into the SDN architecture, see [4]. The firewall functionality is provided by six modules which run on the top of the controller, which maintain tables located in the controller and the OpenFlow-enabled switches for processing the connections states. The main module is responsible for coordinating and managing other units. The Event_Listener module is triggered by some predefined events to be responded with an appropriate action. The role of State_Extractor module is extracting required information from the packet. The SecPol_Div module is responsible for initiation of SecPolTable table for every new switch connects to the controller. The other table is STable. It is located in the switch and is used to save the state of each flow passes via the switch. SecPol lookup module is used to check out the SecPolTable belonging to all switches in the whole path for the traffic, to install a new entry in the switches STable table located in the traffic path, Con_State module uses a S_PROTOCOLE_Add message; this approach helps avoiding unwanted delays (switch-controller communication), when the first packet arrives other switches in the path. The following describes the interaction between OpenFlow switches, controllers and the stateful firewall application, and show how they handle the traffic. When the switch receives a packet, first step and before checking the flow table, it will compare the packet's header against the STable entries for any match. If this packet matches with one of the STtable table entries, the switch will forward the packet to its destination as it belongs to a previously opened connection. Otherwise, it will match the packet against the Flow Table entries to perform the convenient action. In case that no entry defined in the OpenFlow table explains how to handle such connection, the switch either performs the default action or encapsulates the packet with packet-in message and forwarded it to the controller. Once the controller receives the packet-in message, it will consult the SecPolTable related for that switch, and respond with the proper action encapsulated with packet-out messages to install one or more appropriate entries in the flow table of requesting switch. The Event_Listener module is listening to the packet-out message to call three functions State_Extractor, SecPol_Lookup, and Con_State. The State_Extractor function extracts the header attributes needed to make up the communication session (e.g. IP source/destination addresses, port numbers, sequence/acknowledgment numbers, and flags) which together considered the fingerprint for any individual connection. The function SecPol_Lookup is responsible for checking out all SecPolTable tables belong to the switches located in the packet path to make sure it does not violate the network security policy. Finally the Con_State function to install entry in STable of corresponding switches using S_PROTOCOLE_Add message and all switches in the packet path, such approach helps avoid additional overhead (unwanted switch-controller communication delay) when this packet arrives for remaining switches in its path. One more important introduced mechanism done by the SecPol_Div, which for each data plane device join to the controller, this SecPol_Div object initiates SecPolTable for that new connected switch, this mechanism improves the overall performance by reducing time needed by the controller to decide how should that switch handle the packet if it does not match any entry in its flow table. When security policy updated by the network administrator, the SecPol_Div will rebuild all the SecPolTable and reinforce and dynamically propagate the security policies for every data plane. In order to maintain the STable located in the OpenFlow-enabled switch for adding, removing or even clear all entries, the designated communication protocol S_PROTOCOL between the controller and the switch is introduced. S_PROTOCOLE protocol defines three messages: S_PROTOCOLE_Add, S_PROTOCOLE_Remove, and S_PROTOCOLE_Clear, to enable state-based connection monitoring and to help the ConState module to maintain the STable by adding, removing and clearing the STable entries of the data plane.

5 Implementation and Functional Validation

5.1 Implementation

A proof-of-concept is constructed to validate and test the proposed subsystems functionality. We deploy and configure our virtualized test environment, see Figure 8, in which the firewall functionality is mostly implemented as interconnected modules running on the top of the controller (we chose a widely used POX controller), except for the state table placement, which uses a low-level, assembly-based language called NetASM [34],

which comes with a software switch capable of creating tables and defining layouts of the processing pipeline. The testbed consists of the several following machines:

1. The authentication server, which is Ubuntu server 16.04.3 LTS VM with the FreeRADIUS [35] server v2.1.12 installed along with MySQL database to save the authentication and authorization resources.
2. The authenticator, which is Ubuntu server 16.04.3 LTS VM running the modified version of open-source 802.1x hostapd [22]; it is modified to maintain a local database as well as establish a secure channel with the controller.
3. The POX [36] controller, which is installed on Ubuntu server 16.04.3 LTS VM along with two applications, forwarding.l2_learning module and our own application; it consists of modules performing the firewall functionality and a module collecting the result of the authentication process of the supplicant from the authenticator through the SSL channel (it must install required entries in the corresponding switch.
4. The switch, which is Ubuntu server 16.04.3 LTS VM running Mininet [37] network emulator with NetASM language to create OpenvSwitch [38], helping to connect different entities for testing purpose.
5. Finally, the supplicant, which is Ubuntu Desktop 16.04.3 LTS VM installed with wpa-supplicant for network clients.

5.2 Validation

To prove the effectiveness and behavior; and to analyze the performance overhead of our proposed solution, several experiment scenarios are conducted and tested, see Figure 8. In the first scenario, we tested the control access subsystem; Client-1 and Client-2 are trying to access the network. For the test purpose, Client-1 is authorized and its identification is previously saved in the authentication server, therefore as soon as the controller receives the notification of Client-1 of authentication success, it installs entries in the OpenFlow table explicitly reflecting the client privileges. The other unauthorized host is Client-2; its access request is rejected and the switch refuses any attempt to access the network resources. The second experiment was implemented to show the system efficiency of providing the different level of access (privileges) for the network users. OpenFlow-based SDN paradigm allows simple ”Match-Action” processing, which gives a great utility of traffic control, where the packets are filtered basing on the protocol, source/destination IP addresses, and source/destination port numbers. For example, we can easily block a specific host to open an HTTP session with a given server-1, while still allow this host to access other services on that particular server. Typically, port numbers are associated with applications/protocols, so we can easily determine which applications are allowed alongside with the hosts that can access them; allow or deny the access to the port associated with that specific application number determines if that application can be used, and which devices can access it. For this test, the authenticated Client-1 is allowed to open an HTTP sessions while denied to start Telnet sessions. The installed entries are to allow any Client-1 request with the TCP port (80/8080) and deny its request with the TCP port equal to 23. The last experiment is to validate stateful firewall solution. Client-1, which is connected to Switch-2, starts HTTP session with server-1 that is connected to Switch-3. When the first packet reaches to Switch-2, it is forwarded to the controller for consultation. The controller replies with packet-out message to install an entry in Switch-2, to tell forwarding such packets to Switch-3. Simultaneously, Con_State function installs new entries in the STable table for, both, Switch-2 and Switch-3, using a S_PROTOCOLE_Add message. The other packets belong to this flow and the reply from server-1 is compared against the STable that is used to keep track of this matter. In the other case, Client-3 wants to start a connection with server-1; so, Switch-1 will check the flow with its STable table, showing that there is no match with any previous connection, so the Switch-1 will drop the packets, as we implement the security policy to reject any HTTP sessions from out LAN.

6 Summary and Conclusions

Securing the SDN networks is a hard challenge; it is so important and is a key to the success of this technology. In this paper, we provided the security solution for SDN networks consisting of two subsystems: the network access control system and distributed reactive stateful firewall. The former subsystem adopts the 802.1x standard to secure the network by preventing any access attempts from unauthorized users, verifying the host identity upon connection to the network, and implementing different levels of privileges for each host, based on its authentication credentials which are saved in the authentication server. The latter subsystem presents a distributed stateful firewall solution, in which the security policy is centrally enforced by an application running on the top of the controller. It is implemented by the underlying data plane that makes use of state tables used to keep track of each flow in the network to provide stateful packet filtering.

The combination of these subsystems helps creation of additional boundaries within the network to provide multi-level of defense, solving the problem of single point of failure, and protect the network from external attacks as well as from internal malicious users. For both subsystems, the policy is centralized and dynamically enforced to the data plane(s), and the reactive mode is implemented to keep the data planes flow table small. We

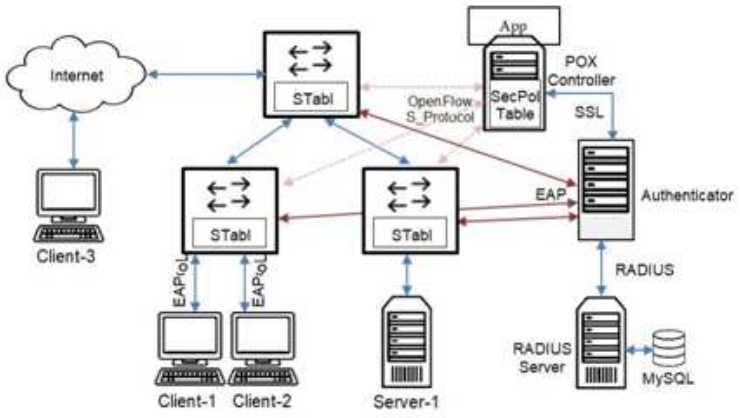

Fig. 8: Experiment testbed

reclaim to implement both reactive and proactive modes for both subsystems to improve the expected overall performance as a future enhancement.

References

[1] O. Reyad and Z. Kotulski, Image Encryption Using Koblitz's Encoding and New Mapping Method Based on Elliptic Curve Random Number Generator, In: A. Dzich et al. (eds) MCSS 2015, CCIS Springer, Heidelberg, Vol. 566, pp. 34–45 (2015).

[2] G. Pujolle, Software Networks Virtualization, SDN, 5G and Security, ISTE Ltd and John Wiley & Sons, Inc., Great Britain and the United States (2015).

[3] D. Kreutz, F.M.V. Ramos, P.E. Verissimo, C.E. Rothenberg, S. Azodolmolky and S. Uhlig, Software-defined networking: A comprehensive survey, Proc. IEEE 103, pp. 14–76 (2015).

[4] F. Nife and Z. Kotulski, Multi-level Stateful Firewall Mechanism for Software Defined Networks, In: P. Gaj et al. (Eds.): CN 2017, CCIS Springer, Vol. 718, pp. 271–286 (2017).

[5] F. Nife and Z. Kotulski, New SDN-Oriented authentication and Access Control Mechanism, In: P. Gaj et al. (Eds.): CN 2018, CCIS Springer, Vol. 860, pp. 74–88 (2018).

[6] IEEE Std., 802.1X-2010 IEEE Standard for Local and Metropolitan Area Networks - Port-Based Network Access Control (2010).

[7] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz, Extensible authentication Protocol (EAP), RFC 3748 (Proposed Standard) (2004).

[8] D. Simon, B. Aboba, and R. Hurst, The EAP-TLS authentication Protocol, RFC 5216 (2008).

[9] C. Rigney, S. Willens, A. Rubens, and W. Simpson, Remote authentication Dial In User Service (RADIUS), RFC 2865 (Draft Standard) (2000).

[10] V. Fajardo, J. Arkko, J. Loughney, and G. Zorn, Diameter Base Protocol, RFC 6733 (Proposed Standard) (2012).

[11] R.K. Sharma, H.K. Kalita and B. Issac, Different firewall techniques: A survey, 5th IEEE Int. Conf. on Comp. Commu. and Net. Tech., China, pp. 1–6 (2014).

[12] K. Scarfone and P. Hoffman, Guidelines on Firewalls and Firewall Policy, Gaithersburg (2009).

[13] A.X. Liu, Computer and Network Security, Firewall Design and Analysis, Vol. 4, World Scientific Publishing Co. Pte. Ltd. 5 Toh Tuck Link, Singapore (2011).

[14] Q. Duan and E. Al-Shaer, Traffic-Aware Dynamic Firewall Policy Management: Techniques and Applications, IEEE Comm. Magazine, Vol. 51, pp. 73–79 (2013).

[15] Z. Trabelsi, Teaching stateless and stateful Firewall Packet Filtering: A Hands-On Approach, In: 16th Colloquium for Information Systems Security Education, Florida, pp. 95–102 (2012).

[16] J.M. Kizza, Guide to Computer Network Security, Fourth Edition, Springer International Publishing AG (2017).

[17] M. Casado, M.J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, Ethane: Taking Control of the Enterprise, in ACM SIGCOMM, Kyoto, Japan (2007).

[18] A. Nayak, A. Reimers, N. Feamster, and R. Clark, Resonance: Dynamic Access Control for Enterprise Networks, In Workshop: Research on Enterprise Networking (WREN), Barcelona, Spain (2009).

[19] V. Dangovas and F. Kuliesius, SDN-driven authentication and access control system, in The International Conference on Digital Information, Networking, and Wireless Communications (DINWC), Society of Digital Information and Wireless Communication (2014).

[20] F. Kuliesius and V. Dangovas, SDN-enhanced campus network authentication and access control system, in 2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN), pp. 894–899 (2016).

[21] D.M. Ferrazani Mattos and O.C.M.B. Duarte, AuthFlow: authentication and access control mechanism for software defined networking, Annals of Telecommunications, pp. 1–9 (2016).

[22] K. Benzekki, A. El Fergougui and A.E. El Alaoui, Devolving IEEE 802.1X authentication capability to data plane in software-defined networking (SDN) architecture,

Security Comm. Networks, 2016 John Wiley & Sons, Ltd., pp. 4369–4377 (2016).
[23] J. Matias, J. Garay, A. Mendiola, N. Toledo, and E. Jacob, FlowNAC: Flow-based network access control, in European Workshop on Software Defined Networks (EWSDN), Budapest, Hungary (2014).
[24] S.T. Yakasai and C.G. Guy, FlowIdentity: Software-defined network access control, in IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN), pp. 115–120 (2015).
[25] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson and G. Gu, A Security Enforcement Kernel for Openow Networks, HotSDN'12 ACM Special Interest Group on Data Comm., Finland, pp. 121–126 (2012).
[26] S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu and M. Tyson, FRESCO: Modular composable security services for software-defined networks, In: NDSS 2013 Network and Distributed Sys. Security Symposium, San Diego, CA, pp. 1–16 (2013).
[27] H. Hu, W. Han, G.J. Ahn and Z. Zhao, FLOWGUARD: Building robust firewalls for software-defined networks, In: HotSDN'14, ACM, Chicago, USA, pp. 97–102 (2014).
[28] S. Shirali-Shahreza and Y. Ganjali, FleXam: Flexible sampling extension for monitoring and security applications in OpenFlow, In HotSDN'13 Hong Kong, China, ACM, pp. 167–168 (2013).
[29] S.S. Shahreza and Y. Ganjali, Efficient Implementation of Security Applications in OpenFlow controller with FleXam, IEEE 21st Annual Symposium on High-Performance Interconnects, pp. 49–54 (2013).
[30] J. Collings and J. Liu, An OpenFlow-based prototype of SDN-oriented stateful hardware firewalls, IEEE 22nd International Conference on Network Protocols, Raleigh, USA, pp. 525–528 (2014).
[31] S. Zerkane, D. Espes, L. Parc and F. Cuppens, Software Defined Networking Reactive Stateful Firewall, International Federation for Information Processing 2016, Springer, Vol. 471, pp. 119–132 (2016).
[32] D. Kreutz, F.M. Ramos, and P. Verissimo, Towards secure and dependable software-defined networks, Proceedings of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, pp. 55–60 (2013).
[33] B. Heller, R. Sherwood, and N. McKeown, The controller placement problem, Proceedings of the 1st Workshop on Hot Topics in Software Defined Networks, New York, USA: ACM, pp. 7–12 (2012).
[34] M. Shahbaz and N. Feamster, The case for an intermediate representation for programmable data planes NetASM, In SOSR (2015).
[35] D.V Walt, FreeRADIUS, Manage your network resources with FreeRADIUS, Packt Publishing Ltd. Birmingham B3 2PB, UK (2011).
[36] L.R. Prete, C.M. Schweitzer, A.A. Shinoda and R.L. Santos, Simulation in an SDN network scenario using the POX controller, IEEE Colombian Conference in Communications and Computing, pp. 1–6 (2014).
[37] R.L. Oliveira, C.M. Schweitzer, A.A. Shinoda, and L.R. Prete, Using mininet for emulation and prototyping software-defined networks, IEEE Colombian Conference on CC, pp. 1–6 (2014).
[38] B. Pfaff et al., The Design and Implementation of Open vSwitch, Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation, pp. 4–6 (2015).

Fahad Naim Nife is a PhD student at the Faculty of Electronics and Information Technology, Warsaw University of Technology, Poland. He received his MSc in Computer Science from Pune University, India. His main research interest is in Software-Defined Networking security.

Zbigniew Kotulski is a Professor and Head of the Security Research Group at the Faculty of Electronics and Information Technology, Warsaw University of Technology, Poland. He received his MSc in Applied Mathematics from Warsaw University of Technology and PhD and DSc degrees from the Institute of Fundamental Technological Research of the Polish Academy of Sciences. He is the author and co-author of five books and over 170 research papers on Applied probability, Cryptography, Cryptographic protocols and Network security.

Omar Reyad is a Lecturer of Computer Science at the Faculty of Science, Sohag University, Egypt. He received his PhD degree from the Faculty of Electronics and Information Technology, Warsaw University of Technology, Poland. He received his MSc in Computer Science from Sohag University. His main research interests are in Elliptic curve cryptography, Cryptographic protocols and Biometric security.