Secure Exchanges Activity in Function of Event Detection with the SDN

Salim Mahamat Charfadine(✉), Olivier Flauzac, Florent Nolot, Cyril Rabat, and Carlos Gonzalez

Université de Reims Champagne-Ardenne, Laboratoire CReSTIC, Reims, France
{salim.mahamat-charfadine, carlos.gonzalez-santamaria}@etudiant.univ-reims.fr, {olivier.flauzac,florent.nolot,cyril.rabat}@univ-reims.fr

Abstract. With the exponential evolution of the Internet of Things (IoT), ensuring network security has become a big challenge for network administrators. Network security is based on multiple independent devices such as firewall, IDS/IPS, NAC where the main role is to monitor the information exchanged between the inside and outside perimeters of the enterprises networks. However, the administration of these network devices can be complex and tedious if it is performed independently on each of them. In recent years, with the introduction of the Software Defined Networking concept (SDN) offers many opportunities by providing a centralized and programmable administration. In this article, we propose a distributed SDN architecture for IoT with a coupled controllers/IDS, by using APIs to dynamically analyze, detect and delete malicious flows. The management of network security is therefore simplified, dynamic and scalable with this approach. We also present the deployment of a real network to test our solution.

Keywords: IoT · SDN · Security · OpenFlow · Firewall · IPS/IDS · NAC

1 Introduction

With the emergence of a large variety of internet-connected devices which are used in many areas of everyday life, such as health, education, economy, transport and military, it raises new challenges related to the network security management and monitoring a high network traffic of end-users communication.

Nowadays, the most networks security systems are commonly based on traditional techniques such as firewall, intrusion detection system (IDS), intrusion prevention system (IPS) and network access control (NAC). These mechanisms are difficult to manage and need to evolve toward the next network generation architectures [1].

To simplify the management and to secure network traffic exchanges, the new concept of SDN was introduced in 2011. This technology has many and varied advantages presented in [2]. The SDN is a new approach for network architecture which consists in decoupling the control plane from the data plane[1,2], allowing the centralization of all control functions on an external node which is called SDN controller. The controller implements forwarding flow rules and it may be installed on one machine or several physical or virtual machines. It exists several types of SDN controllers such as ONOS[3], OpenDayLight[4], POX[5], RYU[6] and so on. Their fundamental differences are related to the programming language and the southbound supported protocol. For example, the OpenDayLight (ODL) controller that we use in our testbed platform, is designed in Java and Python including REST services.

OpenFlow is a standard protocol[7] that allows the communication between the SDN controller and network devices (switches, routers, etc.). The message exchanges are encrypted by the SSL protocol. OpenFlow protocol has the ability over the SDN controller to access and manipulate [3] forwarding rules of packets installed on an OpenFlow switch. ODL supports the OpenFlow version 1.3.

The enthusiasm of the principal technology companies such as Google and Microsoft [4,5] in the deployment of SDN at their datacenters create many opportunities for this new concept which can become a reality. Moreover, it has become an open solution and is begin to be generalized in small infrastructures.

However, IoT involves new challenges. The number of devices is greater, resulting in an increase of flows rules, security menaces and the overload on the controller. Classical architectures must therefore be adapted to take those issues into account. In this article, we propose a decentralized solution based on SDN controllers coupled with IDS. In a domain that we call a cluster, each network OpenFlow device is connected to a SDN controller. The IDS allows to monitor the flow of a cluster and, by communicating with the controller via the REST API, to block the malicious flows. In this case, it reduces the overload of the controller.

In order to analyze our solution, we deployed an architecture in a real test environment based on virtual machines. Thus, instead of using simulators or emulators, we can analyze our solution in a real case of use and highlight monitor OpenFlow messages exchanged.

In the next section, we present a state of the art on network security with SDN. Then, we propose our security approach that allows to detect and isolate a flow of malicious packets dynamically with the SDN concept. Finally we conclude by including some ideas for our perspectives of future works.

[1] https://www.opennetworking.org/sdn-resources/sdn-definition.
[2] https://www.opennetworking.org/sdn-resources/openflow.
[3] http://www.onosproject.org.
[4] https://www.opendaylight.org/.
[5] https://openflow.stanford.edu/display/ONL/POX+Wiki.
[6] https://osrg.github.io/ryu/.
[7] https://www.opennetworking.org/sdn-resources/openflow.

2 State of the Art

To secure a network, we have two main directions: The first option includes the use of the traditional solution based on specific security components (like firewall, IPS, IDS, etc.) and the second one uses the SDN architecture. Previously, we describe that the SDN simplifies the network management with a centralized global view allowing to program the security thanks to the different APIs provided by the controllers.

Software Defined Networking (SDN). To illustrate the operational principle of SDN, we propose a simple network with two hosts h1 and h2 connected through a switch managed by an ODL controller. Figure 1 shows the different phases of an ICMP packet exchanged between h1 and h2. First, the ICMP packet is sent to the switch (1) and it is transmitted to the controller (2). The controller analyses the ICMP packet and installs the flow on the switch (3). h2 receives the ICMP packet (4). Finally, the flow is installed and h1 can exchange ICMP packets with h2 (5).

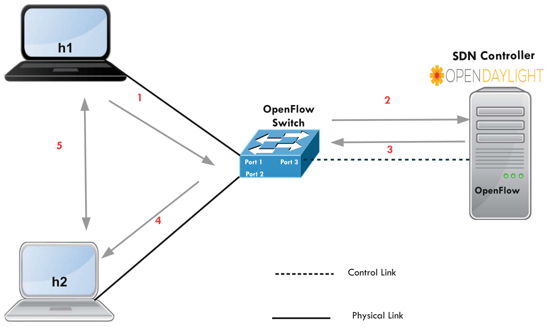

Fig. 1. Flow exchange in an OpenFlow network

When a flow rule is installed on the switch, the controller will not be contacted again to install the same flow. In this case if h1 or h2 are compromised, all the network security become vulnerable.

SDN and the Networks Security. In [6], the authors propose a level 2 centralized firewall based on MAC address filter and the SDN POX controller. But, the network attacks are more and more sophisticated, this kind of detection is not suitable. The authors in [7] describe a firewall application based on layers 2 to 4. They analyze network traffic and compare the packets headers received according to predefined rules. If the application detect a threat the packet is deleted, otherwise it is transferred to its destination. In these two works, the SDN controller is central point of detection. However, the centralization with only one controller is a critical point failure: if the controller is attacked and if an attacker takes its control, the security of the whole network is compromised.

In addition, if there are a large volume of traffic the controller will have an overload resources. The controller is no longer enough available to perform security and the analysis of network traffic.

In this context, in [8], the authors propose to deploy an SDN infrastructure with a single controller. The study overview the impacts of OpenFlow related to flow processing on communications latency. Using the Mininet network simulator and implementing ICMP traffic, they can verify that the filtered rules of the firewall reach the controller. Even though, the latency can increase due to a bottleneck using a single controller. The use of Mininet reflects only a part of the variability of a real network traffic which involve a large number of OpenFlow rules.

To improve this fact, Flauzac *et al.* propose in [9] a solution based on several controllers with the possibility to organize them into domains. The aim is to distribute the control network in multiple controllers. An other solution based on this architecture is described in [10]. The authors propose the use of several NOX controllers coupled to Hyperflow. Hyperflow is an application that allows the propagation of events detected by a controller to the neighbors controllers. Also, the proposed solution helps the management of a controller failure, redirecting the flows traffic to the other controllers.

Theses solutions reduce the bottleneck on a single controller, but the division into domains is not dynamical because the number of components increase and reduces the efficiency of each controller. Redirect the network traffic impacts on the network and controllers overload, increasing the latency.

A solution consists on separate a part of the network traffic analysis by using another security tool coupled to the controller: for instance, an IDS or an IPS. Several solutions has been proposed in [11–13] only tested on Mininet. Furthermore, the rules need to be specified on the controller and on the IDS which difficult the network management effectively.

To improve dynamicity, the authors of [14] and [15] use the machine learning and deep learning concepts coupled to the IDS. The neural network as well as known datasets, they show that their solution could have good performances in most of the environment test. However, a global architecture based on alerts detection linked to the SDN controller has not been proposed. Also, it may be difficult to adapt with several neural networks.

SDN and IoT. The authors in [16], provides a current status overview of the IoT networks as well as the security challenges such as object identification, privacy, integrity, authentication, authorization, and malicious software threats. Also, they propose a security architecture with an SDN-based security mechanism where an IoT controller exchanges messages with the IoT agents. The IoT controller is responsible for the transfer decisions based on information received from the IoT agents and then send the network policies rules through the SDN controller. Upon receiving the connection request from an IoT agent, the IoT controller establish the forwarding rules based on network protocols and communicate these rules to the SDN controller.

Bull *et al.* [17] provides a method to detect and mitigate the suspicious activity of a connected devices from an SDN-based IoT gateway. The IoT gateway has flows entries pre-installed to allow flow traffic analysis, detecting any suspicious behavior and it can associate to many actions. Three types of action have been defined: block, transfer or apply QoS rules. The issue of this proposed solution is the static configuration of the rules on the IoT gateway.

In [18], the authors propose a distributed security architecture for IoT by using the SDN architecture. Their solution secure the traffic monitoring of the entire network and the high availability with several SDN controllers synchronized and organized into domains. They also propose a multi-domain routing protocol in an SDN framework to secure the integrity of messages exchanges between the controllers with the different domains. Although, this solution has the advantage of a tested virtual environment. However, it does not have an intrusion detection system. Threats from inside/outside perimeter of the network could compromise the security management.

3 Collaborative Solution for Securing Network Exchanges

Our solution is inspired by the concept of grid of security [1] and the smart firewall approach [19] to improve security in a traditional network and extend these proposed solutions to IoT. In this approach, we propose a collaborative security solution with a distributed controller architecture coupled with an IDS as shown on Fig. 2. We divide the network into clusters. A cluster is a SDN-domain that use OpenFlow protocol for communication between the networking devices with the SDN controller, and an IDS to manage the security on domain which we call the trusted zone.

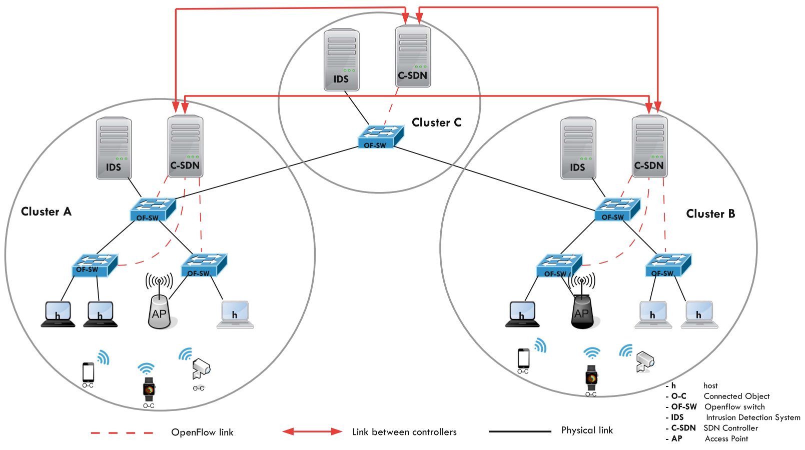

Fig. 2. Our SDN distributed cluster-based architecture

As we described previously, once a flow request is installed on an Openflow switch, the controller do not receives another request for the same forwarding rules which exposes the network to threats in case of compromise devices. Figure 3 shown how to proceed in three phases to solve this problem.

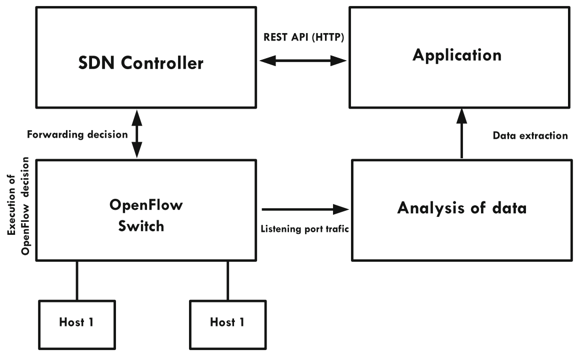

Fig. 3. Theoretical model of the approach

Network Data Collection. During this first phase, we collect the data to be analyzed and find a mechanism to take control over all of the data. For this, we use the port mirroring technique to track all network traffic flow through a particular port which is constantly analyzed.

Analysis, Detection and Generation of Alerts in Case of Threat. The second phase consists of analysis and detection of a threat, followed by generation of an alert in a directory of files logs. To realize this step, we use an intrusion detector system which can analyze, detect and generate log files on malicious flow.

Removal of the Malicious Flow. In the last phase, we developed an application that allows to extract and analyze the logs. Then, if a suspicious device is detected, the IDS send an instruction to the SDN controller to dynamically delete malicious flows via the REST API.

By following this procedure, on each trusted zone the IDS analyzes the traffic and sends an alert to the controller in case of a malicious flow detected. The controller makes a decision by sending a policy rule to the switch via the OpenFlow protocol prohibiting the flow request of a suspicious nodes. Each controller has its own security policies implemented based on the IDS response time.

To prevent the threats in other clusters, the controllers exchange information about security threats in their respective domains preventing the propagation to other clusters through its East-Westbound API.

4 Implementation

In this part, we explain the implementation of a network management by an OpenDaylight controller and a Snort IDS.

Installation of OpenDayLight. The OpenDayLight Controller is an open source network operating system developed in Java and supported by the Linux Foundation. It is based on a modular architecture and can be programmed via applications using the SDN northbound APIs. OpendayLight communicates with network devices using southbound APIs. The most common southbound protocol used in SDN environment is OpenFlow.

To make forwarding decision at the level 2/3 of the OSI model, the OpenDayLight controller knows the network topology as well as the identity of each devices connected with their IP and MAC addresses. The flow on the OVS switch are set up with OpenFlow 1.3 version in order to manage and update the entire network.

For test purpose we have installed a virtual machine on a VMware platform with 2 CPU, 16 GB of RAM and the OS Ubuntu 16.04. The SDN controller installed on this machine is the OpenDayLight Beryllium-SR4.

Architecture Implementation. In the literature many tested solution uses the mininet network simulator. We use virtualization in a production environment with VMvare platform in order to test real case of use.

To realize our virtual network architecture, a second virtual machine is set up with Ubuntu 16.04 OS, 2 virtual CPUs and 16 GB of RAM. On this machine, we installed an OpenFlow 1.3 compatible virtual switch (OVS version 2.6.0) and Qemu, an open source virtual machine emulator for x86 architecture.

The OVS is an open source software implementation of an Ethernet switch with a multilayered and distributed system. It is designed to support level 2/3 of the OSI model switch in virtual environments including different protocols and standards. In our work, it allows the communication between the end-point devices. Qemu is used to emulate the end-point devices with an Alpine Linux OS, a lightweight Linux distribution with 48 MB of RAM.

A bash script is developed to launch several virtual end-point devices with the ability to remotely manage each one of them. The same script is used to launch the OVS interconnecting Alpine Linux virtual machines with each other. This allows us to create the link between the OpenFlow switch and the OpenDaylight controller allowing the control of whole network with OpenFlow protocol. Also, a dynamic IPv4 address assignment with DHCP is perform by the same code to each device. On our set up OpenFlow network we have the ability to scale the number of nodes and the OVS dynamically.

Snort Setting Up. In order to detect the threats, we used a Snort IDS. It's an open source network intrusion detection software that allows to analyze IP network traffic in real time and to detect a wide variety of attacks (e.g. port scan) with the ability to analyze protocols and search the content of matching rules.

In this study, we used Snort in NIDS mode, suitable for monitoring multiple network interfaces. In this mode, Snort acts as a network intrusion detector by analyzing network traffic and comparing this traffic with rules set up by the network administrator.

To deploy Snort, we use a centralized architecture with an IDS which monitors the network traffic on a particular port. Then with a mirroring port technique all the ports traffic of the network is forward to a particular port that is constantly analyzed. This centralized architecture with Snort presents the advantage of a simple implementation, but the disadvantage is bottlenecks in the event to perform scalable networks.

The integration of snort 2.9.11 into the our network platform, a third virtual machine is set up with an Ubuntu 16.04 OS, 2 virtual CPU and 16 GB of RAM. Figure 2 shows the integration of snort in our testbed platform.

After setting up Snort, we defined some non-exhaustive rules for generating logs for any ICMP request queries such as echo request and echo reply, port scan and source or MAC IP address spoofing. The alert data generated by Snort are saved in a log file.

To simulate an attack and evaluate the Snort detection, we installed the Nmap tool on one of the client virtual machine device. Then, we launch many successive denial of service attacks, port scans and an IP address spoofing from the attacker machine. The simulated attack scenarios to observe the reaction of our solution are described as follow:

Service Denial. At this step the aim is to detect and block attempts to saturate a target machine with DDoS attacks with the ICMP protocol. We proceeded by sending ICMP requests to a target machine in our network and determine if Snort reacted by detecting malicious flows.

Port Scan. In this case, the IDS detects any port scan attempts on the TCP or UDP protocols and it can block these requests from the source machine. If an attacker launches a scan to identify open ports and available services on the network, the snort IDS can detected this attack attempt.

IP or MAC Address Spoofing. With this kind of attack an attacker attempts to spoof a legitimate MAC or IP address in order to send packets to the network. A replication of MAC or IP address the systems believe that the source address is trustworthy.

We notice that Snort detected all kind of attacks performed and saved the information into logs files. This procedure can be extended to other types of threats more complex and intelligent.

Linking Snort with OpenDayLight. After setting up the network managed by an OpenDaylight controller and then integrate snort to monitor the network, we developed an application that allows the extraction and the analysis of information on logs generated by Snort. Then, it sends via the REST API a security rule to the OpenDaylight controller in order to uninstall a the malicious flow. REST API is used by the most SDN controllers to exchange network information with applications. To support the REST API, we added the *odl-restconf* feature at the start up of the OpenDaylight controller.

Our developed application exchanges information with the OpenDaylight controller through the REST API to isolate the source machine by executing a shutdown on the port at the origin of the threat.

Figure 4 shows the integration of snort and the OpenDaylight controller on an SDN network.

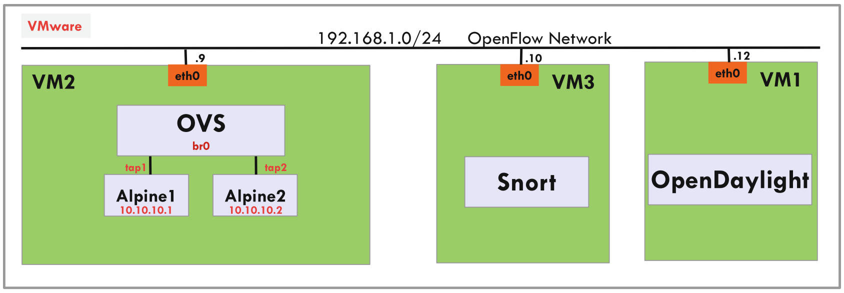

Fig. 4. Snort and OpenDaylight integration model

5 Conclusion

Traditional network security techniques based on independent network devices such as firewall, IPS/IDS and NAC are no longer enough to secure the needs of future networks architecture, especially the IoT. For this reason, we propose a new distributed security solution based on an automated threat analysis, detection and removal managed by the SDN concept. The SDN-based solution and the threat detection system provides the ability to manage security of a network based on events detection.

Nowadays, our solution is tested to secure traditional networks and future works it would be extended to verify the real behaviours performance of IoT devices. In perspective, the set up of a scalable network with several end-point devices will allow us to analysis the reaction time of our proposed solution. Finally, we focus experiments on a single cluster. We plain to monitor the communication between clusters and if it can be achieved by using the controller integrating SDN features.

References

1. Flauzac, O., Nolot, F., Rabat, C., Steffenel, L.A.: Grid of security: a new approach of the network security. In: 3rd International Conference on Network and System Security (NSS 2009), October 2009, Gold Coast, Australia, pp. 67–72 (2009)
2. Sezer, S., et al.: Are we ready for SDN? implementation challenges for software-defined networks. IEEE Commun. Mag. **51**(7), 36–43 (2013)
3. Lara, A., Kolasani, A., Ramamurthy, B.: Network innovation using OpenFlow: a survey. IEEE Commun. Surv. **16**, 493–512 (2014)
4. Wang, S., Li, D., Xia, S.: The problems and solutions of network update in SDN: a survey. In: IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 474–479 (2015)

5. Hu, F., Hao, Q., Bao, K.: A survey on software-defined network and OpenFlow: from concept to implementation. IEEE Commun. Surv. **16**, 2181–2206 (2014)
6. Javid, T., Riaz, T., Rasheed, A.: A layer2 firewall for software defined network. In: Conference on Information Assurance and Cyber Security (CIACS), pp. 1–4. IEEE (2014)
7. Othman, W.M., Chen, H., Al-Moalmi, A., Hadi, A.N.: Implementation and performance analysis of SDN firewall on POX controller. In: IEEE 9th International Conference on Communication Software and Networks (ICCSN), Guangzhou, pp. 1461–1466 (2017)
8. Pena, J.G.V., Yu, W.E.: Development of a distributed firewall using software defined networking technology. In: 4th IEEE International Conference on Information Science and Technology (ICIST), pp. 449–452 (2014)
9. Flauzac, O., Gonzalez, C., Nolot, F.: Original secure architecture for IoT based on SDN. In: International Conference on Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS), pp. 1–6 (2015)
10. Tootoonchian, A., Ganjali, Y.: HyperFlow: a distributed control plane for OpenFlow. In: Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking, vol. 103, pp. 3–3 (2010)
11. Jeong, C., Ha, T., Narantuya, J., Lim, H., Kim, J.: Scalable network intrusion detection on virtual SDN environment. In: IEEE 3rd International Conference on Cloud Networking (CloudNet), pp. 264–265 (2014)
12. Sayeed, M.A., Sayeed, M.A., Saxena, S.: Intrusion detection system based on Software Defined Network firewall. In: 1st International Conference on Next Generation Computing Technologies (NGCT), pp. 379–382 (2015)
13. Chen, P.J., Chen, Y.W.: Implementation of SDN based network intrusion detection and prevention system. In: International Carnahan Conference on Security Technology (ICCST), pp 141–146 (2015)
14. Abubakar, A., Pranggono, B.: Machine learning based intrusion detection system for software defined networks. In: Seventh International Conference on Emerging Security Technologies (EST), pp. 138–143 (2017)
15. Tang, T.A., Mhamdi, L., McLernon, D., Zaidi, S.A.R., Ghogho, M.: Deep learning approach for Network Intrusion Detection in Software Defined Networking. In: International Conference on Wireless Networks and Mobile Communications (WINCOM), Fez, pp. 258–263 (2016)
16. Vandana, C.P.: Security improvement in IoT based on Software defined networking. Int. J. Sci. Eng. Technol. Res. (IJSETR) **5**(1), 2327–4662 (2016)
17. Bull, P., Austin, R., Popov, E., Sharma, M., Watson, R.: Flow based security for IoT devices using an SDN gateway. In: IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud), Vienna, pp. 157–163 (2016)
18. Gonzalez, C., Flauzac, O., Nolot, F., Jara, A.: A novel distributed SDN-secured architecture for the IoT. In: International Conference on Distributed Computing in Sensor Systems (DCOSS), Washington, DC, pp. 244–249 (2016)
19. Gonzalez, C., Charfadine, S.M., Flauzac, O., Nolot, F.: SDN-based security framework for the IoT in distributed grid. In: International Multidisciplinary Conference on Computer and Energy Science (SpliTech), Split, pp. 1–5 (2016)