A Survey on OpenSec: Policy-based Security Using Software-defined Networking

Geeta Zaware[1], Prof.R.N.Phursule[2]
[1, 2] Department of Computer Engineering
[1, 2] Imperial College Of Engineering and Research, Pune

Abstract*-Nowadays policy driven networks has received more attention due to the popularity of software defined networks by a software-based, network-aware controller have replaced manual configuration of multiple an automated approach where a software-based, network-aware controller handles the configuration of all network devices. Software applications running on top of the network controller provide an abstraction of the topology and facilitate the task of operating the network. We propose OpenSec, an OpenFlow-based security framework which allows a network security operator to create and implement security policies written in human-readable language. Using OpenSec, the user can describe a flow in terms of OpenFlow matching fields, define which security services are to be applied to that flow such as deep packet inspection, intrusion detection, spam detection, etc and specify security levels that defines how OpenSec will react if malicious traffic is detected in the system. In this paper, we have provided details about how OpenSec converts security policies into a series of OpenFlow messages which are needed or required to implement such a policy. Then, we describe how the framework will automatically reacts to the security alerts that are specified by the policies. After doing this,, we will perform experiments on the GENI testbed to evaluate the scalability of the proposed framework using existing datasets of campus networks. Our results will show that up to 95% of attacks in an existing data set can be detected and 99% of malicious source nodes can be blocked automatically. Further, we will show that our policy specification language is simpler while offering fast translation times compared to existing solutions.*

Keywords-Software Defined Networking, OpenFlow, Network Security, Policy-based Network Management, Policy Specification

I. INTRODUCTION

In SDN, the complexity of the network shifts towards the controller and brings simplicity and abstraction to the network operator. As we move away from manual configuration at each device, we get closer to automated implementation of network policies and rules. SDN decouples the control plane from the data plane and migrates the former to a logically centralized software-based network controller. More complex network-control applications can thus be implemented at the controller and exploit the fact that they are network-aware due to the centralized nature of the control plane. OpenFlow [6] is a protocol that standardizes how an SDN controller communicates with the network devices. An OpenFlow-compliant switch exposes to the controller an abstraction of its flow table and allows the controller to manipulate it by inserting, modifying or deleting rules in the table. Using OpenFlow, an application running on the network controller can thus control how one or more layer 2 switches forward incoming packets. We propose OpenSec, an OpenFlow-based network security framework that allows campus operators to implement security policies across the network. To motivate this work, suppose a campus operator needs to mirror incoming web traffic to an intrusion detection system (IDS) and e-mail traffic to a spyware detection device. Our goal is to leverage SDN to allow the operator to write a high-level policy to achieve this, instead of having to manually configure each device. Furthermore, suppose the IDS detects malicious traffic and the sender needs to be blocked from accessing the network. Instead of having the operator configure the edge router to manually disable access to the source, we are interested in blocking the sender automatically. Because OpenSec provides an abstraction of the network, the operators can focus on specifying simple and humanreadable security policies, instead of on configuring all the devices to achieve the desired security. OpenSec consists of a software layer running on top of the network controller, as well as multiple external devices that perform security services such as firewall, intrusion detection system (IDS), encryption, spam detection, deep packet inspection (DPI) and others and report the results to the controller. The main goal of OpenSec is to allow network operators to describe security policies for specific flows. The policies include a description of the flow, a list of security services that apply to the flow and how to react in case malicious content is found. The reaction can be to alert only, or to quarantine traffic or even block all packets.

II.REVIEW OF LITERATURE

Our work is related to security policies that uses human readable language and uses it to implement them throughout the underlying networks. Next we evaluate the performance of OpenSec in implementing policies and reacting to security alerts. We also demonstrate the benefits of automated blocking, as well as the advantages of moving middleboxes away from the main datapath and intelligently mirroring traffic using OpenSec. Next, we also measured the time needed to check for conflicts when a new policy is implemented. Although the check time increases linearly, it remains in the order of a few milliseconds. Furthermore, it is unlikely that such a large number of policies will be implemented on a campus network .However, this shows that OpenSec scales well.

A. Procera

The main advantage of OpenSec with respect to Procera is simplicity. We showed in Table VI how OpenSec's syntax is simpler than Procera's to deploy a Science DMZ. Also, the fact that OpenSec does not expose switch events to the enduser simplifies the network administration. We do not provide a quantitative comparison because no comparable numerical results are provided in Voelli et al. [7].

B. CloudWatcher

Next, we compared the time needed by OpenSec to translate policies into OpenFlow messages with the results achieved by CloudWatcher. The experiment consists of translating 50 different random policies. Because CloudWatcher evaluates multiple algorithms, a range of time is given. In all cases, OpenSec achieves a faster time because we do not consider routing in our proposed solution. However, we do note that times for both solutions are similar.

C. Fresco

It compares the time needed to implement the network rules using only one controller, Fresco or OpenSec.The results show that OpenSec needs less time to parse the policy and push rules into the switches. The experimentconsists of implementing 50 different policies with random

III.PROBLEM STATEMENT

Software applications running on top of the network controller provide an abstraction of the topology and facilitate the task of operating the network. We propose OpenSec, an OpenFlow-based security framework that allows a network security operator to create and implement security policies written in human-readable language. Using OpenSec, the user can describe a flow in terms of OpenFlow matching fields, define which security services must be applied to that flow that includes deep packet inspection, intrusion detection, spam detection,etc. and specify security levels that define how OpenSec reacts if malicious traffic is detected. In this paper, we first provide a more detailed explanation of how OpenSec converts security policiesinto a series of OpenFlow messages needed to implement such a policy. Second, we describe how the framework automatically reacts to security alerts as specified by the policies. Third, we will perform additional experiments on the GENI testbed to evaluate the scalability of the proposed framework using existing datasets of campus networks. Our results will show that up to 95% of attacks in an existing data set can be detected and 99% of malicious source nodes can be blocked automatically. Further, we will show that our policy specification language is simpler while offering fast translation times compared to existing solutions.

IV.CONCLUSION

In this paper we propose OpenSec, an OpenFlow-based framework that allows network operators to describe security policies using human-readable language and to implement them across the network. OpenSec acts as a virtual layer between the user and the complexity of the OpenFlow controller and automatically converts security policies into a set of rules that are pushed into network devices. OpenSec also allows network operators to specify how to automatically react when malicious traffic is detected. OpenSec allows for automated reaction to security alerts based on pre-defined network policies. By doing so, it contributes to hiding the complexity of the network to security operators, who only need to focus on defining the policies. This shows several advantages of OpenSec. First, moving the analysis of traffic away from the controller andinto the processing units makes our framework more scalable. Even when the load is high, the controller is not a bottleneck. Second, OpenSec is a first step towards moving the securitycontrols away from the core of the network. This is a key requirement in a network that leverages Cloud security, for example. Instead of controlling every device, the local network just sends data to the cloud and reacts based on the alerts received by the cloud service provider. After this OpenSec will fit in scenarios that require mirroring of traffic in order to monitor devices.

V. ACKNOWLEDGEMENT

I wish to express my profound thanks to all who helped us directly or indirectly in making this paper.Finally, I wish to thank to all our well wishers who supported us in completing this paper successfully. I am especially grateful to our guide Prof. R.N.Phursule Sir for this valuable guidance.Widout the support and cheerful encouragement of my guide,the paper wouldn't have been completed on time.

REFERENCES

[1] A. Lara and B. Ramamurthy, “OpenSec: a framework for implementing security policies using OpenFlow,” in IEEE Globecom Conference,Austin, Texas, USA, December 2014.

[2] A. Lara, A. Kolasani, and B. Ramamurthy, “Network innovation using OpenFlow: A survey,” IEEE Communications Surveys Tutorials, vol. 16, no. 1, pp. 493–512, First Quarter 2014.

[3] “Simplifying network management using Software Defined Networking and OpenFlow,” in IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS), Bangalore, India, December 2012.

[4] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, “Ethane: taking control of the enterprise,” SIGCOMM Comput. Commun. Rev., vol. 37, no. 4, pp. 1–12, October 2007.

[5] H. Kim and N. Feamster, “Improving network management with software defined networking,” IEEE Communications Magazine, vol. 51, no. 2, pp. 114–119, February 2013.

[6] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “OpenFlow: enabling innovation in campus networks,” SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69–74, March 2008.

[7] A. Voellmy, H. Kim, and N. Feamster, “Procera: a language for highlevel reactive network control,” in Workshop on Hot Topics in Software Defined Networks (HotSDN), Helsinki, Finland, August 2012.

[8] 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 2012 20th IEEE International Conference on Network Protocols (ICNP), Austin, Texas, October 2012.

[9] R. Hofstede, L. Hendriks, A. Sperotto, and A. Pras, “SSH Compromise Detection using NetFlow/IPFIX,” ACM SIGCOMM Comput. Commun. Rev., vol. 44, no. 5, pp. 20–26, 2014.

[10] S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, and M. Tyson, “FRESCO: Modular Composable Security Services for Software-Dened Networks,” in Network and Distributed System Security Symposium (NDSS), San Diego, California, U.S.A., February 2013.