Securing Software-Defined Networks using OpenFlow Protocol (SSNOP)

Ancy Julius, Hamil Stanly, Archa A.T.

***Abstract*—OpenFlow is currently the most commonly deployed Software Defined Networking (SDN) technology. SDN consists of decoupling the control and data planes of a network.Network management is a challenging procedure. To operate,maintain and to secure a network, the network administratorsmust cope with the vendor specific low-level configuration toimplement complexnetwork policies.The complexity ofunderlying infrastructure of the network provides only less possibility of innovation and improvement. Software defined networking separates data plane and control plane, leaving a centrally controlled software program to control the overall network behavior. SDN provides new possibilities for securing and controlling a communication network.**

***IndexTerms*— Network security, OpenFlow, Software defined networking.**

I. INTRODUCTION

With the introduction of software-defined networks (SDN) [3], procedures to automate and simplify network operations became popular.In SDN, the complexity of the network shifts towards the controller and brings simplicity and abstraction to the network operator. SDN separates data plane from control plane thereby migrating the control to a logically centralized software based controller. OpenFlow [1] is a standardized protocol that controls the communication of SDN with the devices in the network. 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.

II. SSNOP MODULES

SSNOP is an SDN framework [2] capable of forwarding flows to security processing units based on policies and to automatically react to events raised by these middleboxes.Using this framework, security devices such as intrusion detection middleboxes, firewalls or encryption units can be removed from the main data path between the LAN and the Internet. SSNOP leverages a smart control plane to allow end-users to direct only part of the traffic to these security units.This section describes the main modules in the SSNOP framework.

A. Network Monitoring Module

This module firstly gets the IP address of the server system. Varying the final bits of the IP address the system send ping messages on a loop. The active systems on reception of this ping message, acknowledges back to the server. The server collects the acknowledged IP addresses to an array and displays the same when requested.

The client details are displayed in order in which client tries to login by accessing the login details from the client table of the server.

B. Packet Monitoring

This module is concerned with monitoring the packets traversing through the whole network. Packets are classified into three categories (i.e. TCP, UDP, IP) based on policy rules.

Histograms are constructed using built in functions depending upon the packet count. This module also consists of a network activity segment that monitors the network card of the system to analyze its network activities. Port details and port activities are also monitored and analyzed using this module. The port status (i.e. active, listening, idle, etc.) are analyzed and displayed.

C. Policy Management

The creation and deletion of policies [3] is done by the policy monitoring module. SSNOP's policy specification language allows us to specify a matching pattern, a list of security units that should be traversed by such traffic and an automated reaction in case of receiving a notification from a unit.

	Value	Description
Flow	inPort, VLAN, etherSrc, etherDst, ipSrc, ipDst, TCP-SrcPrt, TCP-DstPrt	Uses OpenFlow match fields to describe a flow
Service	Encrypt, IDS, DPI, spam, DDoS or any other service registered	Identifies a security service that should be applied to the flow
React	alert, quarantine, block	Determines how to react if the service reports malicious content

Table 1: Syntax to create policies using SSNOP

Manuscript received Mar, 2017.

***Ancy Julius**, Computer Science and Engineering, Sarabhai Institute of Science and Technology. Trivandrum, India, +918086926329*

***Hamil Stanly,** Computer Science and Engineering, Sarabhai Institute of Science and Technology.*

***Archa A.T**Computer Science and Engineering, Sarabhai Institute of Science and Technology.*

The service corresponds to any service ID registered by the processing unit manager. Finally, the reaction can be to alert only (via email), to block or to re-route traffic to a quarantine device.

To illustrate how this language is used, an example is given here,

Flow: VLAN=192; **Service**: DPI; **React**: alert

The policy above specifies that all traffic tagged with VLAN 192 should be re-routed to the DPI unit. Also, if the DPI middlebox informs of a suspicious sender, OpenSec must only alert the operator via e-mail. Several match fields and several units can be listed when specifying the policy.

D. AnomalyDetection Module

This module is the major one which is further divided into two sub-modules namely,

i. Policy Monitoring

The major functionality associated with this module is, classifying the available packets on the basis of the rules. Thepackets related to a specific rule are listed and the anomalousbehavior of packets are traced and notified to the administrator. This segment also displays the rules on the basis of classification.

ii. System Monitoring

Whole system activities are monitored using the system monitoring module of the anomaly detection module. This module gives a complete detailing about the available memory space, installed programs, file modification rates, remote processes and report the file details with anomalous behavior. Such files are reported to the system administrator via email or alert and thus producing effective handling activities.

III. Operation of SSNOP: Policy Implementation

This section gives description of how the policy manager of SSNOP converts a policy created through the GUI into forwarding rules in the switches.

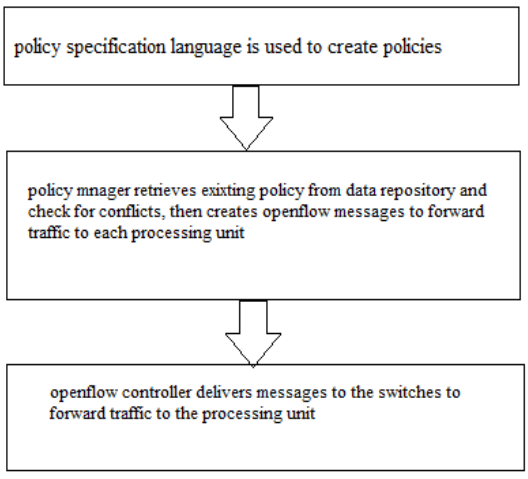

Fig 1: Steps needed to implement policy

First, the policy manager receives the policy definition filefrom the northbound interface (GUI) component. Next,policy is converted to policy object that can be processed by SSNOP to update flow rules in the switches. The policy implementer is the only component SSNOP that is currently tied to the OpenFlowprotocol. It converts the Flow objects created by the policyparser into the specific instance neededby the OpenFlowcontrollers to push newflow rules into theswitches.

Therefore, for each unit, SSNOP first finds the input port where traffic is expected. Once the input port has been found, the match is created based on the policy matching information. Finally, the action is computed as follows. First, an output port must be added so that traffic reaches the middlebox [4]. Second, a VLAN tag must be added to uniquely map this traffic to a policy. Indeed, when a processing unit informs the controller that malicious traffic has been detected, the VLAN id and the source IP address are provided in the notification. These two fields uniquely map a source to a policy, allowing SSNOP to react to traffic coming from the identified source as specified by the policy. This is available in the data repository where all existing forwarding rules are inserted.

IV. Reaction of SSNOP to Security Events.

One of the most enchanting features of SSNOP is its ability to automatically react to security alerts without involving the administrator. The overall process is depicted in the figure below.

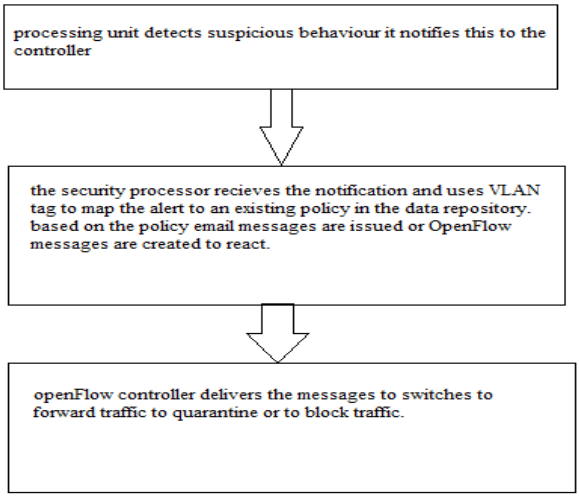

Fig 2: Steps to react to security event

The processing units of the SSNOP framework perform security scans on incoming traffic. As said earlier SSNOP tags traffic with a VLAN to uniquely map it to a given policy. So by sending the VLAN tag and IP address, the processing unit can identify the sender and also can indicate the SSNOP controller which policy caused this flow to be routed to the unit.

While the processing units scan traffic, the security event processor listens to notifications from processing units using application layer sockets. When a new message arrives, a new threat handles it. First, the process reads the VLAN tag and the IP source of the suspicious node. Next, the process

queries the data repository to get the policy mapped to the received VLAN tag. Finally, the process retrieves the type of reaction specified in the policy. As a result, SSNOP now knows if the source must be blocked or sent to the quarantine unit. If the specified reaction is 'alert,' forwarding rules are not modified and the network administrator is notified by e-mail.

The creation of new rules depends on the reaction type. To quarantine traffic, a processing unit logging all traffic isattached to one of the switches on a given port. By default, arule exists on all switches to forward to the quarantine unit alltraffic tagged with a specific VLAN tag. Therefore, to react tosuch an attack, OpenSec must simply insert a rule at the edgeswitch that will tag all incoming traffic from the suspicioussource with the VLAN tag associated to the quarantine unit.

Similarly, if the policy indicates that traffic should be dropped,then OpenSec inserts a rule at the edge switch to doso.

V. Comparison with Existing Systems

This section compares the performance of SSNOP with the existing systems such as Procera and Cloud-Watcher.

a) Procera

The major advantage of SSNOP over Procera is itssimplicity. A quantitative comparison is not provided because no comparable numerical results are provided in Voelli et al. [5]

b) Cloud-Watcher

The time needed for the translation of policies into OpenFlow messages is compared in this case. This comparison doesn't include the time required to send message from controller to switch but only the time required for translation. SSNOP attains faster time because the routing is not considered in this case

VI. Conclusion

In this paper SSNOP, an OpenFlow based framework that allows network operator to describe security policies in human readable language and help them to implement it across a network. This acts as a virtual layer between user and complex OpenFlow controller and automatically converts security policies into a set of rules that are pushed to the network devices. It also specifies how operators should react automatically on detecting malicious behavior. This is actually done based on predefined 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.

The evaluation shows several advantages of SSNOP. First, moving the analysis of traffic away from the controller and into the processing units makes our framework more scalable. Even when the load is high, the controller is not a bottleneck. Second, SSNOP is a first step towards moving the security controls away from the core of the network. Third, OpenSec fits well in scenarios that require mirroring of traffic to monitoring devices. Moreover, our scalability results show that SSNOP achieves a constant reaction time for different traffic rates.

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. Charalambides, P. Flegkas, G. Pavlou, A. Bandara, E. Lupu, A. Russo, N. Dulav, M. Sloman, and J. Rubio-Loyola, "Policy conflict analysis for quality of service management," in IEEE International Workshop on Policies for Distributed Systems and Networks, Stockholm,Sweden, June 2005.

[5] A. Voellmy, H. Kim, and N. Feamster, "Procera: a language for highlevelreactive network control," in Workshop on Hot Topics in SoftwareDefined Networks (HotSDN), Helsinki, Finland, August 2012