SnortFlow: A OpenFlow-based Intrusion Prevention System in Cloud Environment

Tianyi Xing, Dijiang Huang, Le Xu, Chun-Jen Chung, Pankaj Khatkar
Arizona State University, Tempe, USA
Email: tianyi.xing, dijiang, le.xu, chun-jen.chung, pkhatkar@asu.edu

***Abstract*—Security has been one of the top concerns in clouds. It is challenging to construct a secure networking environment in clouds because the cloud is usually a hybrid networking system containing both physical and virtually overlaid networks. Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) have been widely deployed to manipulate cloud security, with the latter providing additional prevention capabilities. This paper investigates into an OpenFlow and Snort based IPS called "SnortFlow", in which it enables the cloud system to detect intrusions and deploy countermeasures by reconfiguring the cloud networking system on-the-fly. The evaluation results demonstrate the feasibility of SnortFlow and provide the guidance for the future work.**

I. Introduction

Cloud computing technologies have been widely deployed and adopted today due to its demanding resource provisioning capabilities, such as scalability, high availability, efficiency, and so on. However, security becomes one of critical issues [1], where cloud resource abuse and malicious insiders are among top threads considered in the current cloud computing systems.

Intrusion Detection Systems (IDS) such as Snort [2] are regarded as the common tools to detect and prevent malicious attacks within a networking system. They monitor networking events and traffic to identify malicious activities, and then issue alerts of detected attacks. The 'detecting and alerting' nature of current IDS solutions demands the cloud security team to hire professional security experts. Moreover, the current cloud IDS lacks of proactive capability to prevent attacks at its initial stage. Thus, intrusion prevention (i.e., IPS) is preferred to IDS in order to automatically take action towards the suspect events. Basically, the IPS can be constructed based on IDS. For instance, Snort can be configured as inline mode and work with firewall systems to implement the IPS. However, there are several issues in the current IPS system: *Latency*: inbound IPS requires inspection and blocking on each network packet, which definitely degrades the performance of handling packets and results in a high latency. For some delay-sensitive services, it might not be appropriate to plant IPS in the system. *Accuracy*: the correctness of the detection is of the utmost importance and is still not that optimized. Reducing the false-positive is always one of the constant goals for the IPS. *Flexibility*: the most common way to realize the IPS prevention is to block the traffic or even block the certain range of the suspect network. This may affect the healthy traffic from innocent neighbors sometimes. How to dynamically reconfigure the network to quarantine the suspect traffic while keeping the traffic alive is the ultimate solution that most IPSs are looking for.

In NICE [3], we addressed the research challenge on how to minimize the intrusiveness when deploying countermeasures dynamically to enable the IPS in the cloud virtual networking environment. NICE proposed an attack-graph based IPS to detect and defense against attacks in the cloud environment. We focused on how to use attack graphs to select appropriate countermeasures and reconfigure the cloud virtual networking system to mitigate or prevent attacks. The alert correlation and countermeasure selection are performed by an attack analyzer in the cloud security control center. There is no module in NICE to function as a Snort management server to coordinate all alerts from multiple cloud servers. Hence, the coordinated multi-step attack across multiple servers is not easy to be detected in NICE. Therefore, motivated by the above challenges and enhancing NICE, this paper primarily focuses on the design and implementation of OpenFlow enabled network-based IPS in cloud environment. We propose a new architecture to support flexible and efficient IPS by integrating Snort and OpenFlow [4] components. By utilizing the power of OpenFlow switches, the cloud networking environment can be dynamically reconfigured based on the detected attacks in realtime. We build up a prototype based on OpenFlow switch and Xen-based [5] cloud environment. The evaluation results show that the proposed system is feasible in the cloud environment and provides valuable guidance for future work.

This paper is organized as follows. Section II introduces the background of OpenFlow and Snort, and discusses the state of the art in this area. The SnortFlow design and implementation are proposed in section III. This work is evaluated in section IV and concluded in section V.

II. Background and State of the Art

A. Background

OpenFlow is a novel technology that significantly improve the way of how current networking devices work. It introduces a centralized and separate controller to significantly enhance the flexibility and enable the network programmability. OpenFlow defines a standard interface to the controller for installing the packet-forwarding rules in flow table which can rapidly handle incoming packets. When a packet arrives at OpenFlow-enabled network device, the switch mainly processes the packet in the following three steps:

978-0-7695-5003-9/13 $26.00 © 2013 IEEE
DOI 10.1109/GREE.2013.25

1) It checks the header fields of the packets and tries to match any entry with the existing flow table. If there is no any matching entry in the flow table, the packet will be sent to the controller for further processing. What is more, there are multiple matching rules. It first picks the exact-match one if one exist, or a wildcard one with the highest priority.
2) It then updates the byte and packet counting information associated with the rules for statistic use.
3) If it matches with any rule, it just take action based on the flow table entry, e.g., forwarding to any specific port, or dropping.

With these features, dynamic network reconfiguration can be enabled by network programmability features in many kinds of systems, e.g., cloud. It is feasible to develop a tenant-based policy or protocol to control both internal Open vSwitch (OVS) [6] and external OpenFlow switch (OFS) in a cyber-physical environment. There are several OFS controllers available following the OpenFlow standard, such as NOX/POX [7]. Both OVS and OFS are OpenFlow-based switches. OpenFlow is also supported by a number of commercial Ethernet switch vendors, and has been deployed in several campus and backbone networks.

Snort is a multi-mode packet analysis tool dominating the IDS/IPS market and has overall performance strength over other products [8]. It basically has sniffer, packet logger, and data analysis tools. In its detection engine, rules form signature to judge if the detected behavior is a malicious behavior or not. It has both host and network-based detection engines and it has a wide range of detection capabilities including stealth scans, OS fingerprinting, buffer overflows, back doors, and so on. Network Intrusion Detection System (NIDS) mode has been widely used and focuses only on detection, thus the action to be taken when the rules are matched are usually log or alert, without disabling the ongoing attacks. The combination of Snort and Iptables is the most common way to implement the Snort IPS mode that is also known as inline mode. The IPS mode is different from IDS mode in that the IPS can prevent the attacks from happening beside just detection. As mentioned in section I, one of the main challenges of IPS is the performance since every packet needs to wait until Snort tells it is safe to pass or not. We will discuss the performance comparison of IDS and IPS in section IV based on the test result in our established cloud environment.

B. State of the Art

Improving the accuracy of NIDS is always the goal for researchers in this area. Selective packet discarding is proposed in [9]. They built up the prototype by using Snort to improve the accuracy of NIDS. In [10], authors show good throughput performance of IDPS by proposing a string matching architecture. Snort and Suricata are two major IDPS system on the market. The performance evaluation of Snort and Suricata in the cloud is discussed in [8], the overall performance of Snort is better than Suricata, which is also the reason why we choose Snort as the candidate for the detection engine.

OpenFlow switch, as a programming network device with high flexibility and scalability, has been largely deployed to enable new services or enhance performance especially for network scenarios [11], [12], [13]. Also, combining the OpenFlow with other opensource packages creates new opportunities. QuaggaFlow [14] integrates the Quagga opensource routing suite with OpenFlow to provide a centralized control over the physical OpenFlow switches and Quagga router in Virtual Machine (VM). However, very few researchers started using OpenFlow as a way for security purpose, especially in cloud environment. In [15], the authors proposed a mechanism called OpenFlow Random Host Mutation (OFRHM) in which the OpenFlow controller frequently assigns each host a random virtual IP that is translated to/from the real IP of the host. This mechanism can effectively defend against stealthy scanning, worm propagation, and other scanning-based attack. Opensafe [16] is similar with our works in that it is a system utilizing both OpenFlow and Snort technology but they focused on the area of how to route traffic to monitoring appliances, rather than attempting to provide a comprehensive solution. However, our aim is to go beyond single-option IPS system such as IP mutating and provide a comprehensive IPS system that can dynamically detect and prevent the attacks. To our best knowledge, we are the first to combine the representative IDS system Snort and the promising future Internet technology OpenFlow to build the flexible IPS system in cloud virtual networking environments.

III. SortFlow: Design and Implementation

In this section, we present the designed architecture and components of the SnortFlow, and then discuss the network reconfiguration mechanism. The architecture and components can be found in Fig. 1.

A. Components

Cloud Cluster is the major component hosting cloud resources and the proposed SnortFlow. It is based on XenServer that is an efficient parallel virtualization solution. There are two types of domains in Xen-based cloud: Dom 0 and Dom U. Dom 0 is the management domain that belongs to the cloud administrative domain. Dom U is the domain hosting the guest VM for users. Dom U resources are managed by Dom 0 and must go through Dom 0 to access the hardware. **OpenFlow Switch (OFS)** connects resources at different cloud servers. With layer-2 connections, any layer-2 and above technology such as VLAN, can be supported across different physical cloud servers. Physical OpenFlow switch can be usually implemented over NetFPGA [17] and selected commercial switches. **Open vSwitch (OVS)** is the software version of the OpenFlow switch. In our scenario, OVS is implemented in the Dom 0 of Xen hypervisor. Communications among different VMs in the same physical server only need to be managed and forwarded through OVS without exposing the traffic out of the physical box. **Controller** is the component providing a centralized control or view over the OpenFlow enabled infrastructure. We choose POX [7] among several

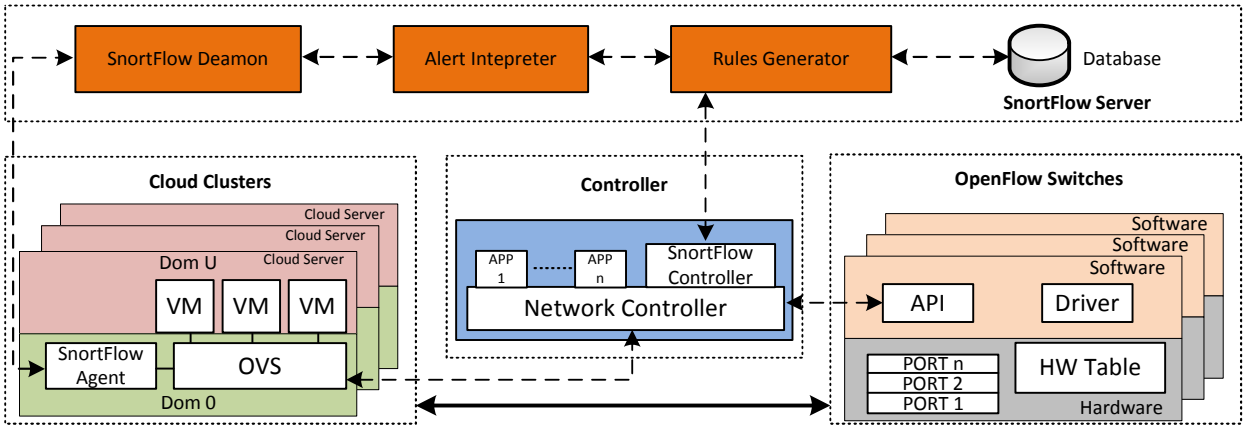

Fig. 1. SnortFlow System Architecture

controller candidates since it is easy to program while keeping excellent performance. One controller can control OFS and OVS simultaneously in order to make both virtual and physical networks synchronized. **SnortFlow Server** is the component that intelligently evaluates the network security status and generates actions to be pushed into the controller in order to fulfill the network reconfiguration tasks. There are basically three modules in the SnortFlow server, SnortFlow daemon, alert interpreter and rules generator. SnortFlow daemon is mainly for collecting alert data from Snort agent in Dom 0. Alert interpreter takes care of parsing the alert and targets the suspect traffic. Then, the parsed and filtered information is passed to rules generator that generates the rules to be injected to the OpenFlow device to reconfigure the network. Database is used to store the generated rules and original state for future operations like resuming functions.

B. Network Reconfiguration

Since the accuracy is one of major concerns for detection engines, taking direct actions on suspect traffic is not always an advisable choice since it may kill the healthy and innocent traffics. With the network programmability feature enabled in both OVS and OFS, the network for whole cloud environment can be reconfigured to provide security protections. Several network reconfiguration actions are summarized in Table I. Intrusiveness is the negative effect that a countermeasure brings to the Service Level Agreements (SLAs). Cost is the unit that describes the expenses required to apply the countermeasure in terms of resources and operational complexity. All the values are from historical experiment data.

TABLE I
NETWORK RECONFIGURATION ACTIONS

No.	Countermeasure	Intrusiveness	Cost
1	Traffic redirection	high	high
2	Traffic isolation	high	medium
3	Deep Packet Inspection	high	high
4	MAC address change	medium	low
5	IP address change	medium	low
6	Block port	high	low
7	Quarantine	high	medium

IV. EVALUATION

A. Evaluation Environment

We establish a simplified testing environment including one cloud server with OVS enabled in Dom 0. The SnortFlow agent is implemented by Snort and installed in both Dom 0 and Dom U. Detection engine in Dom 0 can directly access the bridge in OVS to monitor the cloud network while SnortFlow agent in Dom U can monitor the specific network too. All other components are placed in VMs. We mainly focus on the performance evaluation about SnortFlow agent deployed at Dom 0 and Dom U respectively in order to generate results as guidance for future work. All the traffic are generated by using packet generator to mimic the real traffic.

B. Performance Results

As shown in Fig. 2, the traffic load in terms of packet sending speed increases from 1 to 5000 packets per second. 5000 is the upper bound because we assume 5000 is the capacity threshold of each physical cloud server. Also, from Fig. 3, after 5000 (p/s), success analyzing rate is dropped to a very low value, which means it also exceeds the detection capacity of the SnortFlow. Both Dom 0 and Dom U are configured with 4 virtual CPUs allocated. The CPU utilization is measured in terms of the summation of CPU utilization of Dom 0 and Dom U because SnortFlow in Dom U also consumes computation resources in Dom 0 due to Xen architecture. Snort and Iptables based IPS will be compared with SnortFlow in terms intrusion prevention performance.

Fig. 3 represents the performance of SnotFlow agent in terms of percentage of successfully analyzed packets,i.e., the number of the analyzed packets divided by the total number of packets received. The higher this value is, the more packets the SnortFlow can handle. It can be observed from the result that the success analyzing rate drops when the traffic is above

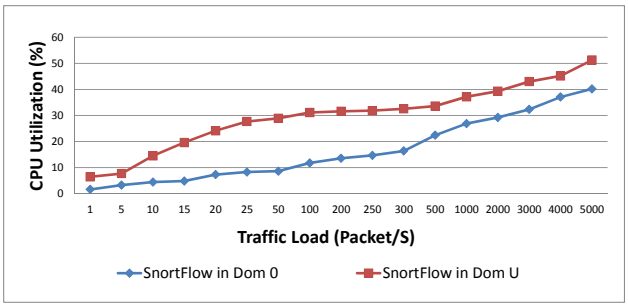

Fig. 2. CPU Utilization Performance Comparison

2500 packet per second. This is because detection agent does not store the captured packets but simply drops them when it reaches maximum capacity. However, they both experience packet drop when traffic load is huge while Dom 0 case is better for about 40%.

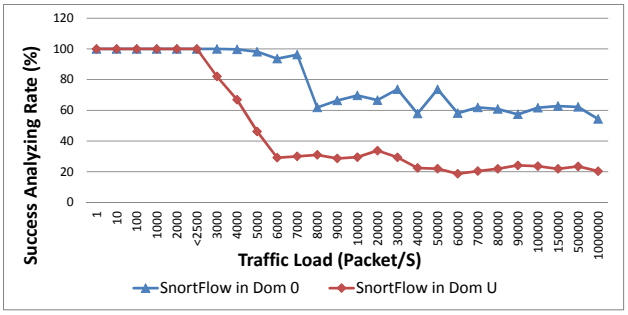

Fig. 3. Successful Analyzing Rate Performance Comparison

Thus, based on the evaluation in the real cloud environment, we can make the consensus that deploying the SnortFlow agent in Dom 0 has better performance than in Dom U. The main reason is that all the traffics of Dom U need to go through OVS in Dom 0, and virtual resources in each Dom U need to be routed to access the hardware resources through Dom 0. Thus directly deploying the SnortFlow in Dom 0 is more efficient in terms of both host and network resource utilization.

V. Conclusion and Future Work

In this paper, we propose an OpenFlow-based Intrusion Prevention System called SnortFlow in Xen-based cloud environment. It inherits the intrusion detection capability from Snort and flexible network reconfiguration from OpenFlow. The evaluation results generated in the real cloud environment show how the performance of SnortFlow agent varies in different deployment scenarios. This is a useful benchmark to conduct the future implementations. The future work of SnortFlow involves the following 3 steps: (1). Multiple control: The controller should be able to control multiple OFS and OVS simultaneously; (2). Synchronization: Currently there is only one Snort detection agent in the system. We are going to collect alerts from multiple SnortFlow agents and generate the network reconfiguration rules by correlating some alerts; (3). Network reconfiguration: Actions listed in Table. I need to be developed to enable the capability of controller to dynamically issue commands to OpenFlow device to reconfigure the network. Additionally, optimized algorithms are needed in both alert interpreter module and rules generator to correlate alerts and reconfigure the network without breaking down the detected vulnerable service. Finally, Snort/Iptables based IPS will also be compared with SnortFlow in terms security performance as future work.

Acknowledgment

Thanks to the organization committee and the tutorial lecturers in the first GENI summer camp, where the authors can get more inside to apply the OpenFlow technologies in the presented research. Particularly, thanks to the kind help and discussion with Dr. Bing Wang from the University of Connecticut, and Mark Berman and Niky Riga at BBN. The presented work is sponsored by ONR YIP award, NSF grant CNS-1029546, and NSF 1029546.

References

[1] C. C. S. Alliance, "Top threats to cloud computing v1.0," in *White Paper*, 2010.
[2] "SourceFire Inc." [Online]. Available: http://www.snort.org
[3] C.-J. Chung, P. Khatkar, T. Xing, J. Lee, and D. Huang, "Nice: Network intrusion detection and countermeasure selection in virtual network systems," in *IEEE Transactions on Dependable and Secure Computing (TDSC), Special Issue on Cloud Computing Assessment*, 2013.
[4] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," in *ACM SIGCOMM Computer Communication Review*, April 2008.
[5] P.Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfieldh, "Xen and the art of virtualization," in *Proceedings of the nineteenth ACM symposium on Operating systems principles (SOSP)*, 2003.
[6] Open vSwitch, http://openvswitch.org/.
[7] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenkes, "Nox: Towards an operating system for networks," in *ACM SIGCOMM Computer Communication Review*, July 2008.
[8] A. Alhomoud, R. Munir, J. P. Disso, I. Awan, and A. Al-Dhelaan, "Performance evaluation study of intrusion detection systems," in *The 2nd International Conference on Ambient System, Networks and Technologies*, 2011.
[9] A. Papadogiannakis, M. Polychronakis, and E. O. Markatos, "Improving the accuracy of network intrusion detection system under load using selective packet discarding," in *EUROSEC*, 2010.
[10] L. Tan and T. Sherwood, "A high throughput string matching architecture for intrusion detection and prevention," in *the 32nd Annual International Symposium on Computer Architecture (ISCA)*, 2005.
[11] B. Koldehofe, F. Durr, M. A. Tariq, and K. Rothermel, "The power of software-defined networking: Line-rate content-based routing using openflow," in *MW4NG*, 2011.
[12] R. Kappor, G. Porter, M. Tewari, G. M. Voelker, and A. Vahdat, "Chronos: Predictable low latency for data center applications," in *SOCC*, 2012.
[13] M. Suchara, D. Xu, R. Doverspike, D. Johnson, and J. Rexford, "Network architecture for joint failure recovery and traffic engineering," in *SIGMETRICS*, 2011.
[14] M. R. Nascimento, C. E. Rothenberg, M. R. Salvador, and M. F. Magalhaes, "Quagflow: Partnering quagga with openflow," in *SIGCOMM Poster*, 2010.
[15] J. H. Jafarian, E. AI-Shaer, and Q. Duan, "Openflow random host mutation: Transparent moving target defense using software defined networking," in *HotSDN*, 2012.
[16] J. R. Ballard, I. Rae, and A. Akella, "Extensible and Scalable Network Monitoring Using OpenSAFE," in *INM/WREN*, 2010.
[17] Stanford University. [Online]. Available: http://netfpga.org/