SDNIPS: Enabling Software-Defined Networking Based Intrusion Prevention System in Clouds

Tianyi Xing[1], Zhengyang Xiong[1], Dijiang Huang[1], Deep Medhi[2]
[1]Arizona State University, U.S.A,{tianyi.xing, zhengyang.xiong, dijiang}@asu.edu
[2]University of Missouri–Kansas City, U.S.A, dmedhi@umkc.edu

***Abstract*—Security has been considered as one of the top concerns in clouds. Intrusion Detection and Prevention Systems (IDPS) have been widely deployed to enhance the cloud security. Using Software-Defined Networking (SDN) approaches to enhance the system security in clouds has been recently presented in [1], [2]. However, none of existing works established a comprehensive IPS solution to reconfigure the cloud networking environment on-the-fly to counter malicious attacks. In this paper, we present an SDN-based IPS solution called SDNIPS that is a full lifecycle solution including detection and prevention in the cloud. We propose a new IDPS architecture based on Snort-based IDS and Open vSwitch (OVS). We also compare the SDN-based IPS solution with the traditional IPS approach from both mechanism analysis and evaluation. Network Reconfiguration (NR) features are designed and implemented based on the POX controller to enhance the prevention flexibility. Finally, evaluations of SDNIPS demonstrate its feasibility and efficiency over traditional approaches.**

I. Introduction

Cloud computing platforms have been widely proposed and implemented due to the flexibility, scalability, high availability, efficiency, and so on [3], [4], [5]. Security has been regarded as one of the most critical issues where cloud resource abuse and malicious insiders are among top threads considered in the current cloud computing systems [6]. Based on compromised cloud resources, attackers may spam, disseminate malicious codes, crack passwords and security keys, compromise vulnerable VMs and then deploy DDoS attacks, deploy botnet command and control, etc. Existing countermeasures usually provide add-on and customizable security models, and consider the cloud can afford the demanded resource.

Establishing the Intrusion Detection and Prevention System (IDPS) is a good way to protect the cloud system with both detection and prevention capability. Traditionally, the IDS can be configured and enabled to be an IPS. For instance, Snort [7] can be configured as the inline mode and work with a common firewall system, e.g., Iptables, to implement the IPS in the cloud networking environment [8]. However, there are several issues with such a traditional IPS: 1) Latency: in-bound IPS requires inspection and blocking action on each network packet, which consumes cloud system resources and increases the detection latency; 2) Resource Consumption: running the IDPS services usually consumes significant resources. For instance, configuring SPAN port mirroring technology will duplicate all traffic and forward to a port that an IDS is connected; 3) Inflexible Network Reconfigurations: traditional IPS does not have network programmability feature to reconfigure the virtual networking system and provide scrutinized traffic inspection and control.

SDN based security approaches in a cloud virtual networking environment has been considered as the trend for future virtual networking security solutions. Opensafe [9] 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 detection and prevention solution. In [10], 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, but does not work when the attackers know the internal address of victims. In a recent work [2], the authors presented an SDN-based IDS/IPS solution to deploy attack graph to dynamically generate appropriate countermeasures to enable the IDS/IPS in the cloud environment. SnortFlow [1] is another recent work focusing on the design and preliminary evaluation of OpenFlow [11] enabled IPS in the cloud environment. However, to our best knowledge, none of them address the issue below: 1) how to establish an efficient SDN-based IPS solution in the cloud virtual networking environment; 2) how does SDN-based IDS/IPS compare with traditional one; 3) how to design the SDN-based IDS/IPS networking architecture that provides a dynamic defensive mechanism for clouds.

Therefore, motivated by issues above, we present the SDNIPS, a SDN-based IPS security solution in clouds. This paper proposes a new design of IDS/IPS based on SDN network management, i.e., Open vSwitch (OVS); it also introduces a comprehensive comparative study based on the presented SDNIPS and Snort/Iptables based IPS [12] solutions to demonstrate the advantages of the SDN-based IDS/IPS solution.

II. SDNIPS: Design and Implementation

In this section, we present the designed architecture including components and the processing flow of the SDNIPS, which is then followed by the Network Reconfiguration (NR). The architecture and components are presented in Fig. 1.

A. Overall Architecture and Components

Cloud Cluster is the major component hosting cloud resources and the environment where the proposed SDNIPS is

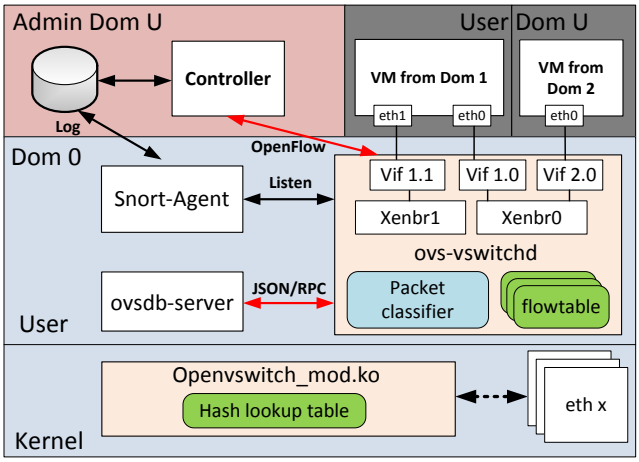

Fig. 1: The SDNIPS System Architecture.

applied. A cloud cluster can contains one or multiple cloud servers with major cloud-based OS installed. In this paper, our established system is based on XenServer that is an efficient parallel virtualization solution on top of the bare metal. **Open vSwitch (OVS)** is the pure software implementation of the OpenFlow switch [11]. OVS is usually implemented in the management domain or privilege domain of the cloud system. In user-space of OVS, there are two modules which are *ovsdb-server* and *ovs-switchd*. The module *ovsdb-server* is the log-based database that holds switch-level configuration; while the module *ovs-switchd* is the core OVS component that supports multiple independent data-paths (bridges). As shown in Fig. 1, *ovs-switchd* module is able to communicate with *ovsdb-server* through management protocol, with controller through OpenFlow protocol, and with kernel module through netlink. In the kernel space, kernel module handles packet switching, lookup and forwarding, tunnel encapsulation and decapsulation. Every Virtual Interface (VIF) on each VM has a corresponding virtual interface/port on OVS, and different virtual interface connecting to the same bridge can be regarded on the same switch. For example, VIF 1.0 (the virtual port of eth0 on VM from Dom 1) has the layer 2 connection with VIF 2.0 (the virtual port of eth0 on VM from Dom 2). **Snort** is a multi-mode packet analysis tool dominating the IDS/IPS market and has overall performance strength over other products [13]. It has sniffer, packet logger, and data analysis tools. In its detection engine, rules form signatures to judge if the detected behavior is a malicious behavior or not. It can be implemented in Dom 0 (privilege domain) or Dom U (unprivileged domain) based on Xen virtualization architecture. In this paper, we deploy the Snort in Dom 0, which makes it natively detect the bridge in OVS and has better performance [1]. **Controller** is the component providing a centralized view and control over the cloud virtual network. The controller contains three major components, SDNIPS daemon, alert interpreter, and rules generator. SDNIPS daemon is mainly for collecting alert data generated from Snort agent in Dom 0 of controlled SDN devices, i.e., OVS. The SDNIPS daemon is implemented in the format of JSON message. The alert information is stored in JSON message and the JSON server is running at the controller side. Alert interpreter takes care of parsing the alert and targets the suspect traffic. Several information is parsed out of the raw alert data, e.g., source IP address, destination IP address, TCP port, etc. 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.

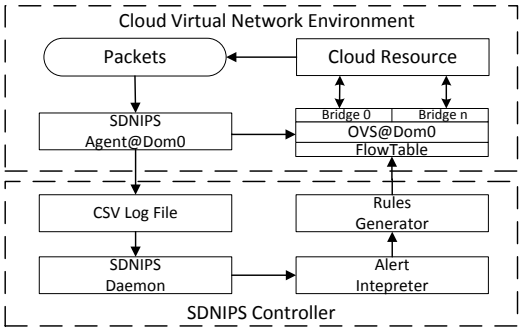

Fig. 2: The SDNIPS Processing Flow.

B. SDNIPS Processing Flow

The processing flow of the SDNIPS is illustrated in Fig. 2. The network traffic is generated from the cloud resources, i.e., VMs. Snort agent in Dom 0 has the advantage of directly detecting through the bridge, which is more efficient than sniffing the traffic by utilizing the SPAN technology. When any traffic matching the Snort rules is alerted into the log file, The SDNIPS daemon will store the alert information in JSON format and send over to the JSON server at controller side. After that, the alert interpreter will parse the alert information and extract all necessary information, e.g., attack type, source IP, destination IP, TCP port, etc. Finally, the rules generator will generate the OpenFlow rule entries and push them to the OVS to update the flowtable. Therefore, the following suspect traffic matching the newly updated flowtable entries will be swiftly handled with valid countermeasures in the data plane of the OVS with line rate. Currently, the system described in Fig. 1 is implemented.

C. Network Reconfiguration (NR)

NR is an approach to reconfigure the network characteristics including topology, packet header, QoS parameters, etc. With the SDN concept enabled in the cloud virtual networking environment, network reconfiguration can be applied to construct the countermeasure of the IPS system. Traffic redirection redirects the traffic by rewriting the destination address. QoS adjustment adjusts the QoS parameters. Traffic isolation isolates the traffic by using virtual networking technology such as VLAN. Filtering filters the packets by matching any field of the flowtable and take corresponding actions, e.g., drop. Block port blocks the TCP/UDP ports. Major network reconfiguration actions are summarized in Table I.

TABLE I: Network Reconfiguration Actions

No.	Countermeasure
1	Traffic Redirection
2	QoS Adjustment
3	Traffic Isolation
4	Filtering
5	Block Port

III. SDNIPS vs Traditional IPS (Snort/Iptables)

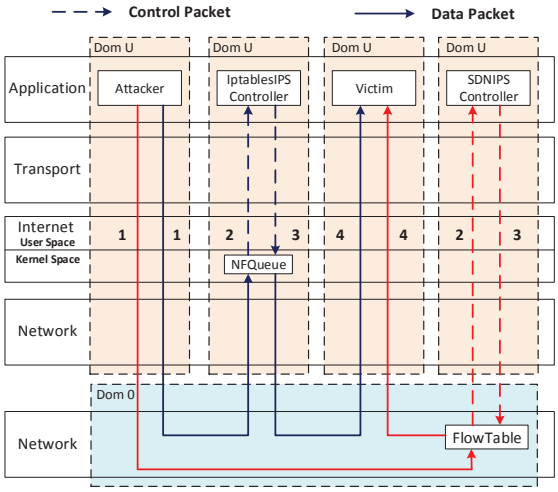

Fig. 3: SDNIPS and Snort/Iptables IPS Mechanisms.

Traditional IPS system is not specially designed for the cloud virtual networking environment, but for a general network environment. Two IPS solutions are different in terms of the essence, i.e., working mechanism and operation level. Fig. 3 indicates the scenario on how the Iptables IPS (blue lines) and SDNIPS (red lines) prevent the attacks. The number besides each line represents the sequence of the packet flow. Solids lines and dot lines represent the data traffic and control traffic respectively.

As shown in Fig. 3, when attacking packets generated from attacker's virtual interface, all the packets need to be passed through Dom 0 before being forwarded to the destination (blue line 1). When Snort detects any suspect traffic, it needs to inform the NFQUEUE, which is an Iptables and Ip6tables target delegating the decision on packets to a user-space software, to take the actions defined in the rules. The Iptables IPS needs to consult its brain (controller at application level), which then sends out control messages to issue command (blue line 2 & 3). Finally, the suspect packet is handled at Internet level kernel space at Dom U and will be either forwarded to victim or dropped (blue line 4). Unlike the Snort/Iptables IPS, SDNIPS stands out since both the detection engine and the packet processing are natively deployed in Dom 0, which is dramatically efficient especially handling large amount of traffic. When packets arrive at Dom 0 (red line 1), Snort detection engines is able to natively monitoring the bridges, even though the OVS controller is placed at Dom U, only few control messages between OVS at Dom 0 and controller at Dom U are generated (red line 2 & 3). After the controller update the flowtable, all traffic with the same pattern will be processed at OVS fast path in Dom 0 (red line 4). From the Fig. 3, it is also obvious that packets in Snort/Iptables IPS scenario need to be in and out the Dom 0 twice while the SDNIPS only needs once to fulfill the same task. Although control message in SDNIPS has further way to go than in Iptables IPS, the control message only updates the flowtable at the first time when traffic is suspected and all the traffic will be only handled by flowtable fast path at Dom 0 without interaction with controller at Dom U. Thus, due to the IPS working mechanism, SDNIPS should significantly outperformance any other Dom U IPS solution especially in cloud virtual networking environment.

IV. Evaluation

We establish the SDNIPS prototype by using one cloud server with OVS installed and properly configured in Dom 0 which has 4 virtual CPUs. The detection engine, i.e., Snort, in Dom 0 can directly access the virtual bridges in OVS to monitor all tenant networks while Snort/Iptables-based IPS agent in Dom U can only monitor the tenant network where it is. The OVS controller is implemented based on the POX controller [14]. Traditional Snort/Iptables IPS is implemented in a VM (Ubuntu 12.04 Server edition, 4 virtual CPUs and 2048 MB Memory) at Dom U. All VMs at Dom U are configured with VIF with 10GbE maximum capacity whose actual bandwidth is around 8 Gbits/s based on our testing in the real XenServer virtual network environment. In our evaluation, the traffic are generated by hacking tools and packet generators such as [15], [16], [17] to mimic the real attack scenario in the cloud virtual networking environment.

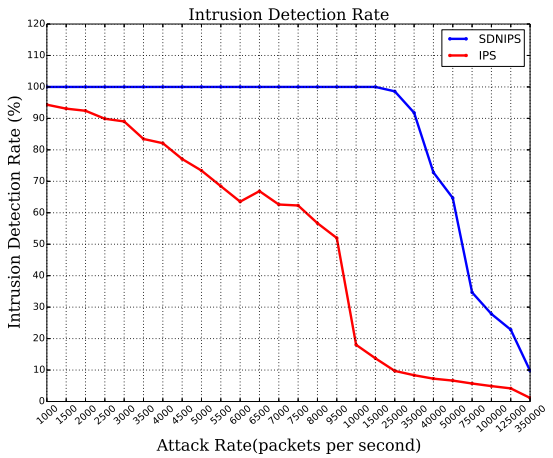

Fig. 4: Evaluation of Intrusion Detection Rate.

In Fig. 4, we evaluate the alert generation capacity of both IPS and SDNIPS under flood interference, which also states

how well the IPS can handle the attacking packets. To evaluate this performance, we generate two different types of attacks, which are DoS flooding attack acting as the stressful background traffic and ICMP flood attack acting as an potential threat to be tested. This evaluation mainly indicates whether IPS and SDNIPS can generate alert under high workload stress in terms of successful alert generation rate of ICMP attack under DoS attack interference. When the speed of the ICMP attack reaches to 15000 packets per second, IPS can only generate 13.72% alerts of the ICMP attack. On the other hand, SDNIPS is able to efficiently avoid interference from DoS flooding attack due to OVS capability, so it can successfully alert all the threats that are sent at the speed of 15000 packets per second. When the speed of the ICMP attack reaches to 30000 packets per second, the performance of SDNIPS start decreasing, and when the speed of ICMP attack increases to 300000 packets per second, Snort agent in SDNIPS is not able to capture packets and launch alerts because the snort detection engine itself almost reached its threshold.

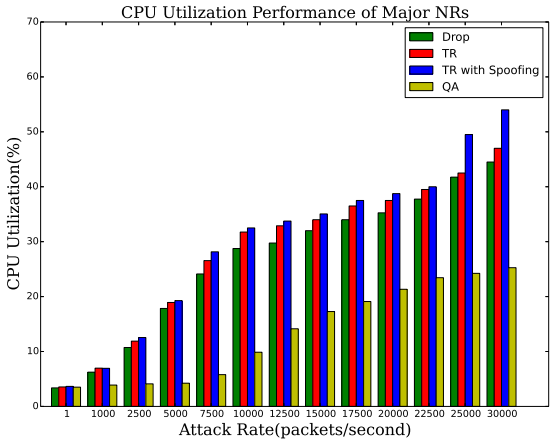

Fig. 5: CPU Utilization Performance of Major NRs.

After the SDNIPS and traditional IPS are comparatively evaluated, we evaluate the performance of SDNIPS NR alone since traditional IPS does not have the NR capability. We mainly evaluate two NR actions mentioned above, Traffic Redirection (TR) and QoS Adjustment (QA), as well as default drop action. Fig. 5 shows the performance of resources consumption in term of the CPU utilization of Dom 0 for NRs. We are using packet generator [15] to generate the packets captured by Snort and processed by flowtable in order to test the resources utilization change in the cloud system. In each NR approach, TR is implemented by using destination IP & MAC rewriting; while TR with spoofing reply is implemented by rewriting not only destination IP & MAC address but also source IP & MAC of victims. Thus, the attacking traffic can be redirected to a security appliance that is able to spoof the attacker by replying the packet with victims' IP & MAC address as source address. TR with spoofing feature consumes a little more resources than the pure TR since OVS modifies more packet fields to enable the spoofing feature. The default NR, i.e., drop packets, consumes less system resource because the OVS does not modify the matching flow and just simply drop them (output to a non-existing virtual port in POX controller implementation). In the QA scenario, it has the best performance among all NRs because the rate limiting action is performed based on OVS native mechanism, which means excess packets will be discarded and OVS does not have to inspect and match the packet with all kinds of fields.

V. Conclusion

In this paper, we propose an SDN-based Intrusion Prevention System called SDNIPS in the cloud virtual networking environment. It inherits the intrusion detection capability from Snort and flexible network reconfiguration from SDN. SDNIPS is firstly compared with traditional IPS from principle perspective and the real world evaluation. NR actions are also designed and developed based on OVS and POX controller. The evaluation proves its feasibility and efficiency.

Acknowledgment

The presented work is sponsored by ONR YIP award and NSF grant CNS-1029546.

References

[1] T. Xing, D. Huang, L. Xu, C.-J. Chung, and P. Khatkar, "Snortflow: A openflow-based system in cloud environment," in *GENI Research and Educational Experiment Workshop, GREE*, 2013.
[2] 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.
[3] T. Xing, D. Huang, D. Medhi, and S. Ata, "Mobicloud: a geo-distributed mobile cloud computing platform," in *Proceedings of 8th International Conference on Network and Service Management*, 2012.
[4] T. Xing, X. Liu, C.-J. Chung, A. Wada, S. Ata, D. Huang, and D. Medhi, "Constructing a virtual networking environment in a geo-distributed programmable layer-2 networking environment (g-plane)," in *Communications (ICC), 2012 IEEE International Conference on*. IEEE, 2012, pp. 5879–5884.
[5] D. Huang, T. Xing, and H. Wu, "Mobile cloud computing service models: a user-centric approach." *IEEE Network*, vol. 27, no. 5, 2013.
[6] C. C. S. Alliance, "Top threats to cloud computing v1.0," in *White Paper*, 2010.
[7] "SourceFire Inc." [Online]. Available: http://www.snort.org
[8] W. Morton, "Intrusion prevention straitegies for cloud computing," 2011.
[9] J. R. Ballard, I. Rae, and A. Akella, "Extensible and Scalable Network Monitoring Using OpenSAFE," in *INM/WREN*, 2010.
[10] J. H. Jafarian, E. AI-Shaer, and Q. Duan, "Openflow random host mutation: Transparent moving target defense using software defined networking," in *HotSDN*, 2012.
[11] 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.
[12] R. U. Rehman, "Intrusion detection systems with snort," in *BRUCE PERENS OPEN SOURCE SERIES*.
[13] 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.
[14] 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.
[15] R. Olsson, "pktgen the linux packet generator."
[16] hping3. [Online]. Available: http://linux.die.net/man/8/hping3
[17] NetPerf. [Online]. Available: http://www.netperf.org/