Network Intrusion Detection System as a Service on OpenStack Cloud

Chen Xu, Ruipeng Zhang, Mengjun Xie, Li Yang
The University of Tennessee at Chattanooga
{kjx384,smj793}@mocs.utc.edu,{Mengjun-Xie,Li-Yang}@utc.edu

***Abstract*—Cloud computing has become a major computing paradigm and data processing approach in almost all sectors. To ensure normal business operation and data security, performing traffic monitoring and detecting suspicious network packets and possible network intrusions have become a daily job for tenant administrators. Using existing tools, a tenant administrator can set up a Network Intrusion Detection System (NIDS) on a virtual machine (VM) instance in the tenant and mirror the traffic from other instances in the tenant to the NIDS instance via Tap as a Service (TaaS) or a Switched Port Analyzer (SPAN) port. However, this type of mechanisms can consume significant resources (e.g., CPU and bandwidth) in the cloud environment. In this work, we propose a new lightweight approach, namely Network Intrusion Detection System as a Service (NIDSaaS), for OpenStack cloud. Our preliminary experimental results show that our NIDSaaS approach consumes much less CPU compared to the existing TaaS approach.**

***Index Terms*—Cloud Security, Neutron Plugin, NIDS, OpenStack**

I. Introduction

As cloud computing becomes increasingly popular, open source cloud platforms such as OpenStack, Eucalyptus, and OpenNebula are receiving more attention and support and growing rapidly. OpenStack, in particular, has become one of major cloud computing platforms and been used to build both public and private computing clouds in many sectors. With more data being moved into the cloud, network security and data privacy in their tenant networks have become vital to the organizations that rely on cloud computing for their businesses. Network intrusion detection systems (NIDS) are often used to guard against cyber attacks. However, existing solutions for implementing network intrusion detection in a cloud environment can incur significant resource consumption and may even affect normal businesses. To cope with daily traffic detection in tenant networks and offer an optimized network security solution, we propose a new approach, Network Intrusion Detection System as a Service (NIDSaaS), for cloud computing. We start with our NIDSaaS design for OpenStack. It provides network intrusion detection capability for tenant virtual networks. By using NIDSaaS, tenant and cloud administrators do not need to set up special NIDS instances. They can use easy-to-use NIDSaaS commands to launch NIDS services to monitor traffic from/to one or multiple instances in real time with low resource consumption. Our preliminary experimental results show that our NIDSaaS approach consumes much less CPU compared with the existing TaaS approach.

II. Related Work

A few studies have been conducted on NIDS deployment in a computing cloud. Santoso *et al.* [1] configured the switch SPAN port that can collect the cloud traffic to forward the traffic to a Snort system. Mahajan *et al.* [2] tried to deploy a Snort instance in different locations to examine external and internal traffic by configuring the Open vSwitch (OVS) SPAN port. Improving NIDS effectiveness is another important topic in cloud security. In vNIDS [3], the authors focused on the challenges regarding effective intrusion detection and non-monolithic NIDS provisioning by using detection state sharing and microservice techniques. The authors of Kitsune [4] utilized neural network to efficiently track patterns of a network channel. However, these proposed methods do not address the issues of tenant instances distributed on different compute nodes or complex configuration of NIDS service.

III. System design and implement

In our design, NIDSaaS has three major components: client extension, plugin, and plugin agent. The NIDSaaS client extension extends the Neutron client command line, which is used to send command messages to the Neutron server. The NIDSaaS plugin residing in the Neutron server handles the commands from clients and plays a role of RPC (Remote Procedure Call) client by sending RPC messages to the network and compute nodes that have deployed NIDSaaS plugin agent. The NIDSaaS plugin agent registers as an RPC server and handles the RPC messages from its plugin.

Figure 1 illustrates how NIDSaaS works. There are three types of commands available for tenant administrators. Commands of type ① are used to create/destroy an NIDS service. Commands of type ② are for updating detection rules of a given NIDS service. Finally, by calling a command of type ③, users can mirror the network traffic from/to a specific Neutron port to the NIDS service, which eventually connects the NIDS service to the target tenant network.

User requests triggered by client commands are sent to the NIDSaaS plugin via HTTP. The NIDSaaS plugin listens on the NIDSaaS REST API endpoints for NIDSaaS commands. Moreover, the NIDSaaS plugin is responsible for storing necessary information of the network, NIDS port and rule set, etc. If the plugin receives command request ①, it will send an RPC message containing the request of command ① to the plugin agent on the network node. After the NIDSaaS plugin agent receives the message, it will perform two important

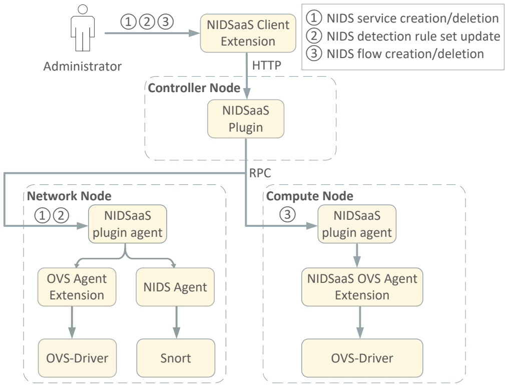

Fig. 1. Workflow of NIDSaaS

tasks. First, the plugin agent creates a Neutron port for the NIDS service to listen on and also sets up OpenFlow rules for the OVS bridge so that it can deliver mirrored packets to the Neutron port. Second, the agent launches the NIDS service with the default configuration and connects the service to the Neutron port. If a command of type ② arrives at the plugin, the NIDSaaS plugin will notify the plugin agent on the network node. The agent will update the rule sets stored locally and restart the corresponding NIDS process to apply those changes. If the plugin receives the command request of type ③, the RPC message will be directed to the NIDSaaS plugin agent on the applied compute node(s) and then forwarded to the NIDSaaS OVS agent extension. The NIDSaaS OVS agent extension on the compute node will invoke the OVS driver to set up the bridges and OpenFlow rules to mirror packets and send them to the NIDS port on the network node.

IV. Evaluation

In our preliminary evaluation, we employed four bare-metal nodes in the Chameleon cloud [5] each with two 2.60 GHz Intel Xeon CPUs (24 cores) and 192 GB RAM to build the OpenStack test environment, which includes one controller node, one network node and two compute nodes. All these bare-metal nodes are connected to a 10 Gbps VLAN. We chose Snort 2.9.11 configured with community rules (3881 rules) as the NIDS service in our experiments.

To simulate network attacks, we applied the network traffic trace collected from Mid-Atlantic Collegiate Cyber Defense Competition (MACCDC) to generate attack traffic, which is sent from a sender VM to a receiver VM using *tcpreplay*. The trace was modified for replay. Each VM used in the experiments has 4 vCPUs and 4 GB RAM. We used *pidstat* to collect resource usage. We collected the CPU usage of Snort in the following three scenarios.

- **Baseline**: Install and launch the NIDS service in one OpenStack compute node and sniff network traffic from the target network directly.
- **TaaS**: Boot an NIDS-enabled VM in the OpenStack. The target network traffic is mirrored to this VM using TaaS so that the NIDS in the VM can sniff the network traffic indirectly.

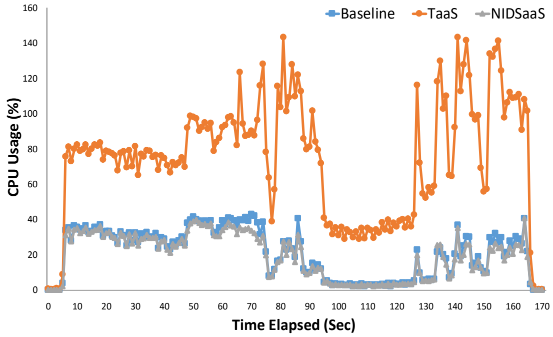

Fig. 2. CPU usage

- **NIDSaaS**: Deploy and launch the NIDS service on the OpenStack network node directly. The NIDS service will sniff on a dedicated Neutron port to which the target network traffic is mirrored. Compared with the baseline, NIDSaaS can monitor tenant network traffic from different compute nodes.

We replayed the MACCDC trace at the rate of 50 Mbps ten times in each scenario. The average CPU usage of Snort in the three scenarios is shown in Figure 2. From the figure, Snort in the Baseline consumes the least CPU among all the three scenarios, which is expected. Snort in the TaaS scenario consumes much more CPU due to the virtualization overhead. The CPU usage of the Snort VM sometimes exceeds 100%, which means two or more CPU cores are being occupied by the VM. Compared to TaaS, NIDSaaS consumes far less CPU no matter whether the sender and receiver VMs are on the same physical node or not. In fact, the experiments in the NIDSaaS scenario show quite similar CPU usage patterns as those in the Baseline scenario.

V. Conclusion

We have presented our preliminary study on implementing NIDS as a Service on OpenStack. Compared with an existing approach through TaaS, our method consumes less CPU. Our future work includes more comprehensive measurement and optimization of NIDSaaS performance and scalability.

References

[1] B. I. Santoso, M. R. S. Idrus and I. P. Gunawan, “Designing Network Intrusion and Detection System using signature-based method for protecting OpenStack private cloud.” In *Proc*. InAES, pages 61-66, 2016.

[2] Varun Mahajan, Sateesh K Peddoju, “Deployment of Intrusion Detection System in Cloud: A Performance-based Study.“ In *Proc*. IEEE Trustcom/BigDataSE/ICESS, pages 1103-1108, 2017.

[3] Hongda Li, Hongxin Hu, Guofei Gu, Gail-Joon Ahn, and Fuqiang Zhang. “vNIDS: Towards Elastic Security with Safe and Efficient Virtualization of Network Intrusion Detection Systems.“ In *Proc*. CCS, pages 17-34, 2018.

[4] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. “Kitsune:An Ensemble of Autoencoders for Online Network Intrusion Detection.“ In *Proc*. NDSS, 2018.

[5] K. Keahey, P. Riteau, D. Stanzione, T. Cockerill, J. Mambretti, P. Rad, P. Ruth (2019) Chameleon: a Scalable Production Testbed for Computer Science Research. In: Jeffrey Vetter (eds) Contemporary High Performance Computing: From Petascale toward Exascale, Vol 3. CRC Press.