An OpenFlow-based Architecture for IaaS Security

Antonio Marotta[1], Gabriella Carrozza[2], Stefano Avallone[1], Vittorio Manetti[2]

[1]University of Naples Federico II (DIETI) Via Claudio 80125 Naples, Italy
{antonio.marotta, stavallo}@unina.it

[2]SESM s.c.a.r.l.Via Circumvallazione Esterna, Giugliano in Campania 80014 Naples, Italy
{gcarrozza, vmanetti}@sesm.it

General Terms
Security, Design, Experimentation, Measurement, Performance

ABSTRACT
Cloud Computing technology and its service model, Infrastructure as a Service, are emerging as the leading approaches to encourage the scalable and efficient utilization of resources and the convenient consumption of elastic services. Despite all the advantages that derive from the application of Cloud Computing IaaS model, when dealing with mission and safety critical infrastructures "built in the cloud", it is needed to be also aware of the security gaps and concerns. In this work we present our proposed architecture to tackle cloud security issues and we describe the first results of our experimental campaign.

INTRODUCTION
Cloud Computing technology and its service model, Infrastructure as a Service, are emerging as the leading approaches to encourage the scalable and efficient utilization of resources and the convenient consumption of elastic services. The IaaS paradigm allows to deploy, configure and run heterogeneous applications without the need to be conscious about the underlying physical infrastructure. The use of a Cloud Computing platform to create a virtualized testbed in order to reproduce a real operational environment allows to obtain a lot of benefits in terms of:

- chance to reproduce real world scenarios in house to perform testing campaigns;
- availability of automatic procedures to implement backup and disaster recovery of entire testbeds;
- possibility to configure and manage the testbed components and the testbed versioning through automatic mechanisms.

Despite all the advantages that derive from the application of Cloud Computing IaaS model, when dealing with mission and safety critical infrastructures "built in the cloud", it is needed to be also aware of the security gaps and concerns. Based on the analysis of the literature, Cloud Computing security issues can be related to different scopes.

The way authentication, authorization and accounting are handled assumes a very high influence: security threats are often originated from internal users, so there is the need to be sure that only an authenticated user can access his granted resources, according to clear-cut global policies. The actions performed by users in relation to the platform's resources should be also registered and accounted for further analysis in case of policy violations. Another important task is the management of the security principles, which are availability, integrity and confidentiality of the cloud data storage. In this case, advanced encryption schemes can be used to guarantee that the proper users are able to access, modify and delete given information.

Virtualization technology, which is the heart behind IaaS model, has rapidly changed the needs and the requirements for network security. Traditional security means, like internal security devices and access control lists are not sustainable when dealing with virtualized servers and resources, due to the strains for making them up-to-date with the rapid changes in the topology. Only authorized hosts and devices should be able to communicate in the virtualized networks, while malicious ones have to be identified and somehow confined. The virtualization layer also poses new security challenges, because virtual guests can be easily compromised in different ways and they can also damage other virtual machines. So, one of the possible remedies is to check virtual machines' behavior by intercepting attempts in the modification of sensible code. At the same time, virtual machines' images can be checked, in order to verify their integrity.

In order to deal with security issues in the cloud and with the dynamism, typical of IaaS approach, we propose an OpenFlow-based [1] architecture which uses classical intrusion detection mechanisms to identify patterns of attacks and that realizes mitigation and recovery strategies in reaction to them. The architecture has been designed and implemented in a virtualized testbed, deployed on an IaaS platform, namely OpenNebula [2], which represents a real world Air Control Center (ACC). The nature of the

application fulfilled by the components of the testbed, really stresses out the lack of robust security solutions and the need for automatic procedures of disaster and attacks recovery. Here we present the first experimental activities that were conducted for the design of the architecture and that cope with:

- the performance comparison among different Open Source OpenFlow Controllers;
- the characterization of three different Open Source IaaS platforms on the basis of the Provisioning Time metric;
- the implementation in the selected Controller of a new functionality in order to provide L2 VLAN encapsulation/de-encapsulation.

OPENFLOW AND THE SOFTWARE DEFINED NETWORKING PARADIGM

The way networking is handled and configured in the virtualized testbed is based on the Software Defined Networking [3] (SDN) paradigm, which can be considered as a new way of thinking about the network. It is basically founded on a sharp distinction between data plane, which is still related to the network devices, and the control plane, which is external and logically centralized. The main benefits that derive from its adoption are in terms of a complete isolation for the application layer and the global view of the network. In the first case, researchers can build their own applications on top of the control layer, so that they are completely isolated from the network devices. Therefore you can write new protocols or applications without affecting the internals of the devices. The second advantage deals with the availability of a global view of the network itself, so it is easy to react to events and changes in the topology. OpenFlow is one implementation of this approach and embodies the interface between the control and data layers. It defines all the messages that are exchanged through a secure channel established between the network switches and an external Controller, that determines the logic according to which traffic flows are forwarded. Nowadays, SDN paradigm is extremely appealing to Cloud Computing Networking as a Service, since it represents a flexible way to create virtual network on the fly and to guarantee multi-tenancy L2 isolation or other network services. Furthermore, results obtained from previous conducted analysis and experiments, lead us to confirm that OpenFlow can allow to reach great flexibility in the network, by assuring dynamism and security policy enforcement, without the need to change the internal architecture of the network components. That is why OpenFlow can be considered as an effective mean to face vulnerabilities, even in a dynamic context like the one of Cloud Computing IaaS, and to implement automatic mitigation/recovery strategies, in the case of security attacks.

THE PROPOSED ARCHITECTURE

The architecture we propose has to be analyzed considering three different layers. The Cloud Layer shows two data-centers, which are geographically connected through a private enterprise backbone network. With the aim to further increase the security level in the connection between the data-centers, we use a splitting mechanism based on MPLS [4], MultiProtocol Label Switching Protocol, that splits packet into parts and redirects them to disjoint paths, so that malicious users that intercept traffic are not able to reconstruct the messages. Each data-center has its own IaaS cluster and there is one main node which is in charge of managing the overall infrastructure. In the Virtualization Layer, the view is independent from a particular platform deployed in one of the data-centers. Regarding the organization, every physical machine, namely a "compute" node, hosts a virtual switch to which all the network interfaces of the guests are plugged. In the virtual switching layer we use the OpenvSwitch [5] technology, which offers a set of functionalities, among which the OpenFlow protocol (v1.0) is also implemented. The flow tables of the switches are programmed by an OpenFlow Controller: when a packet generated by a virtual guest arrives to the switch and there is no match with the available rules, it is sent to the controller, which can decide to install a new flow rule in the switch to forward or discard it. All the traffic produced by the virtual machines is controlled and checked against some well-known patterns of malicious traffic to identify possible attacks. When an anomalous network activity is detected, the alarm generated by Snort [6] is sent through a TLS (Transport Layer Security) socket to an Alarm Correlator that performs the following actions:

- event storage;
- notification process after the extraction of information needed to determine the severity level of the attack;
- identification of the mitigation strategy to implement on the basis of the aforementioned severity level. Such a strategy will be triggered by also interacting with the IaaS manager and the OpenFlow Controller.

The mitigation strategy we intend to implement when an attack against a node of the virtualized testbed is detected, consists in migrating the attacked VM to a different data-center which belongs to the same infrastructure. After the migration process is accomplished, the Correlator can instruct the Controller to change the flows in the virtual switch of the physical node where the guest was previously hosted, in order to assure the transparency of its location.

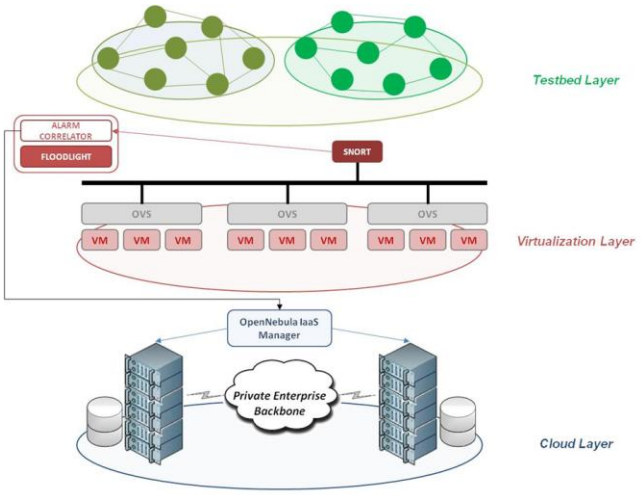

Figure 1 The overall infrastructure

EXPERIMENTAL CAMPAIGN

We carried out the first experimental work with the aim of selecting an Open Source solution among several OpenFlow Controllers. The comparison of the controller's performaces was accomplished through OFlops [7], namely OpenFlow Operations Per Second, which is composed by two software packages:

- OFlops, a particular controller that allows to benchmark lots of features of the switches;
- Cbench (Controller benchmarker), that generates packet-in events for the controller by emulating switches' connection. It is able to calculate the maximum packet-in message generation rate, the delay between packet arrival and packet-in event and the processing delay.

Flow-mod/s	Nox Hub	Nox Switch	Nox Learning switch	Trema	Floodlight
Min	7427	19225	7127	52207	56368
Max	7440	20109	7147	56204	57535
Avg	7435	19916	7137	54333	57142
Stdev	3,55	299,31	6,84	1285	407,81

Table 1 Controllers comparison

Table 1 shows the results in terms of Flow-mod messages per second: through this message, the controller is able to install, modify or delete a flow rule in the switch table. In the comparison we took into account only Open Source Controllers and we also considered other parameters such as the chance to extend and easily modify the modules of the controller, the availability of RESTful APIs and the support behind the development of the project. Our choice fell on Floodlight [8], a Java event-based Controller released under the Apache license, and developed by an open community.

Since we use VLAN technology for communication among virtual machines with the aim to provide L2 isolation, we modified the "Forwarding" module of Floodlight in order to implement VLAN tag encapsulation/de-encapsulation though OpenFlow actions. The VLAN tag that has to be used is directly retrieved from the Cloud platform itself, which is aware of the virtual machines belonging to the virtual network with a specific VLAN tag. The other modification copes with the securing of the channel between the Controller and OpenvSwitch. The latter natively supports SSL handshaking and so we handled the creation of a secure communication with public/private key pairs (generated with the Java keytool [9]) in the implementation of Floodlight connection module.

As last step of our experimental campaign, we evaluated the Provisioning Time of three different IaasS platforms: such a metric refers to the period of time starting from the request for the creation of new VM (through APIs) and ending with the achievement of the "ready" status in the platform. We considered 16 different combinations of four parameters which are:

- **service offering**: the flavor required for the new virtual machine, namely the number of virtual CPUs and the size of the RAM;
- **data storage** (binary): secondary disk storage for the VM;
- **physical node stress**: the number of VMs (0-5) already hosted on the node;
- **automatic scheduling** (binary): such a facility is in charge of picking a physical host where the new VM will be allocated.

We computed the arithmetic mean on 10 different requests of VM creation with the same configuration. For the sake of brevity we just report measures related to a specific combination (the most relevant to our opinion) of the above parameters: (i) a medium service offering (1 vCPU, 2GB RAM), (ii) data storage required, (iii) 5VMs already hosted on the physical node, (iv) scheduling module activated.

	Provisioning Time (s)
CloudStack	17,5714
OpenNebula	22,4789
OpenStack	27,6996

Table 2 Provisioning time

CONCLUSION AND FUTURE WORK

In this work we firstly discussed the context which deals with the challenge of the main security issues related to the Cloud Computing environment. Then we proposed a SDN-based approach to guarantee network security and to undertake selected reactions in case of attacks, by describing all the components needed by our architecture. As future work we aim at using more sophisticated intrusion detection mechanisms in order to be able to detect unknown and unusual traffic patterns. Furthermore, we intend to extend the experimental campaign by performing a more accurate comparison among Cloud Computing IaaS platforms, that is based on other metrics such as: Elasticity, Agility, network stressing and CPU/memory usage.

REFERENCES

1. Nick McKeown, Tom Anderson, Hari Balakrishnan et other "OpenFlow: Enabling Innovation in Campus Networks" Whitepaper
2. www.opennebula
3. Open Networking Foundation "Software-Defined Networking: The New Norm for Networks" Whitepaper
4. Avallone S., Manetti V., Mariano M., Romano S.P. A Splitting Infrastructure For Load Balancing and Security in an MPLS Network. 3rd International Conference on Testbeds and Research Infrastructure for the Development of Networks and Communities, 21-23 May 2007. Pages 1-6
5. Ben Pfaff, Justin Pettit, Teemu Koponen, "Extending Networking into the Virtualization Layer". In 8th ACM Workshop on Hot Topics in Networks (HotNets-VIII) New York City October 2009
6. M. Roesch M. Snort Lightweight Intrusion Detection For Networks Systems Administration Conference (LISA 99), Seattle, WA, Nov. 1999
7. http://www.openflow.org/wk/index.php/Oflops
8. http://floodlight.openflowhub.org/
9. http://docs.oracle.com/javase/tutorial/security/index.html