Integrating a Network IDS into an Open Source Cloud Computing Environment

Claudio Mazzariello, Roberto Bifulco and Roberto Canonico
Dipartimento di Informatica e Sistemistica
Università degli Studi di Napoli Federico II
Via Claudio 21, 80125, Napoli, ITALY
Email: {claudio.mazzariello, roberto.bifulco, roberto.canonico}@unina.it

***Abstract*—The success of the Cloud Computing paradigm may be jeopardized by concerns about the risk of misuse of this model aimed at conducting illegal activities. In this paper we address the issue of detecting Denial of Service attacks performed by means of resources acquired on-demand on a Cloud Computing platform. To this purpose, we propose to investigate the consequences of the use of a distributed strategy to detect and block attacks, or other malicious activities, originated by misbehaving customers of a Cloud Computing provider. In order to check the viability of our approach, we also evaluate the impact on performance of our proposed solution. This paper presents the installation and deployment experience of a distributed defence strategy and illustrates the preliminary results of the performance evaluation.**

***Keywords*-Cloud Computing; Intrusion Detection; Performance Evaluation; Virtualization**

I. Introduction

Cloud Computing is undergoing an indisputable success, which could be indeed jeopardized by concerns about the risks related to potential misuse of this model aimed at conducting illegal activities. In fact, the cheap availability of significant amounts of computational resources can be regarded as a means for easily perpetrating distributed attacks, as it has recently been observed in several security incidents involving Amazon's EC2 cloud infrastructure [1]. The sheer power of attacks from EC2 is indeed raising serious concerns in the community of system administrators and security experts [2]. Furthermore, evidences in recent research works have shown how it is possible to exploit some properties and features of a common cloud computing infrastructure in order to perform attacks against competitors in an industrial scenario [3].

In this paper we address the issue of detecting Denial of Service attacks targeting SIP-based systems; more in detail, we will address the issue of detecting SIP (Session Initiation Protocol) flooding attack instances targeting services hosted within a cloud. To this purpose, we study the impact of security tools deployment in different locations of the cloud, trying to expose the peculiarities of their employment in a cloud infrastructure. In particular, we will evaluate the discrepancies in management cost overhead, due to the employment of one among the possible deployment strategies for the selected security tools. We investigate the usage of both centralized and distributed strategies to detect and block attacks, or other malicious activities, originated by misbehaving customers of a Cloud Computing provider or by external nodes attacking cloud machines and services.

Among the fundamental tools for defending computational and networking infrastructures from malicious behavior are Intrusion Detection Systems (IDS). In classical enterprise settings, an IDS is normally deployed on dedicated hardware at the edge of the defended networking infrastructure, in order to protect it from external attacks. In a cloud computing environment, where computing and communication resources are shared among several users on an on-demand, pay-per-use basis, such strategy is not effective: attacks may be originated within the infrastructure itself, since it can include several administrative domains from the content provisioning point of view, and also be directed against resources located within the cloud infrastructure itself. Hence, a proper defense strategy needs to be distributed. In this paper, we propose to deploy lightweight Network IDSs in each physical machine hosting customers' virtual machines.

This paper is structured as follows: in section II we describe the motivation of this work and general concepts regarding cloud computing, in section III we describe in further details Eucalyptus, a popular open source framework for cloud computing, in section IV we discuss the security tools used in this work and the proposed security system architecture, whereas in section V an experimental evaluation is sketched. Finally, section VI summarizes the paper's conclusions.

II. Cloud computing

Cloud Computing is an innovative computing model in which resources are provided as a service over the Internet, on an as-needed basis, relieving users from the responsibility of buying and managing a dedicated complex computing infrastructure. Cloud providers, on the other hand, can take advantage of scale economies in the organization and management of big datacenters, whose ICT resources can be efficiently exploited by partitioning and renting them to a high number of customers. Despite its success, the Cloud Computing paradigm poses new challenges in terms of security of the computing infrastructure: cloud providers have the responsibility to manage a large infrastructure that hosts a

number of highly dynamic virtual infrastructures operated by different users. Technologies like system virtualization have become for the first time widely adopted to offer computing resources as a service, allowing the dynamic spawn of virtual machines in the datacenter's networking infrastructure.

A. Security Issues

Resource rental on a per-usage basis shifts the responsibility of system management and administration towards specialized teams of experts, virtually reducing security risks typically due to system misconfiguration, lack of proper updates, or unwise user behavior. Despite that, the cloud computing paradigm introduces novel risks due to its inherent resource sharing requirement. Peculiar vulnerabilities, indeed, are introduced by the employment of virtualized host machines sharing common physical resources, by the availability of cheap large scale computation/communication/storage facilities and by the dynamicity of the cloud computing environment. Furthermore, misconfigured remote data storage can expose users' private data and information to unwanted access, or privacy infringement. Each of the security risks enumerated before needs to be dealt with by using a specific technique, trying to respond to the manifold known threats, as well as to the novel challenges which will emerge in the future. That is why an integrated approach, taking different aspects of security-related issues into account, is necessary in order to protect user data and preventing malicious actions both targeting cloud users and originating from within the cloud from being performed.

In the following, we will describe the Eucalyptus cloud computing architecture, trying to expose the security risks related to its employment. The deployment of an intrusion detection system in such a scenario will be described in details, and the resulting performance and computational overhead will be evaluated experimentally.

III. The Eucalyptus Cloud Computing System

Eucalyptus [4] is an open-source framework for cloud computing that implements the paradigm commonly referred to as *Infrastructure as a Service* (IaaS) [5]. Eucalyptus has been designed to be interface-compatible with one of the most popular commercial cloud service, namely Amazon EC2 [6]. The system is based on three components, each with a well defined Web-service interface. The software architecture has been organized according to a three-level hierarchy. The bottom layer consists of Node Controllers (NC), responsible of managing virtual machines running on top of a physical machine. The middle layer contains Cluster Controllers (CC). Each CC manages a set of NCs residing on the same physical subnet. The topmost layer is the Cloud Manager (CM), that manages all the CCs and takes care of high-level resource scheduling. The Cloud Manager is the entry-point to the whole Eucalyptus system for end users as well as administrators. To create instances (the name given to virtual machines in the Eucalyptus and Amazon EC2 terminology) Eucalyptus supports both KVM [7] and Xen [8] virtualization technologies. In this work we will just take Xen into account, since it is the reference technology used also in Amazon EC2. Eucalyptus allows four different networking configurations, but among others, that are mainly targeted for testing environments or small installations, the most interesting for our purposes is the "Managed Mode". In *managed mode* Eucalyptus provides all the functionality present in Amazon EC2, including istances' subnetworks isolation. Network isolation is obtained through the use of VLANs [9], which impose appropriate configurations in data center's switches. Following the Eucalyptus terminology, each instance's network is referred to as a *security group*. Each user is bound to at least one security group, but association to multiple groups can be defined as well if needed. When configuring Eucalyptus for *managed mode*, the administrator must define an IP subnet entirely dedicated to the cloud. Moreover, the administrator must define the number of IP addresses available for each security group, actually defining how subnetting is performed. To guarantee access to external networks, each security group includes the cluster controller among its hosts. Instances are configured to use the CC as default gateway (Figure 1). The CC

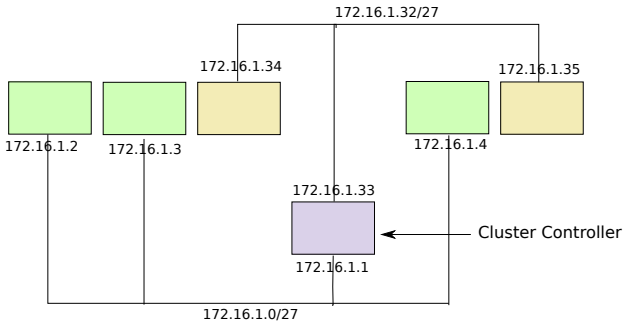

Figure 1. Eucalyptus managed mode networking: lv3 view

provides both DHCP and NAT services. NAT is realized using standard GNU/Linux's *netfilter* functionality. Features like *elastic IPs* are provided by means of rules configured on the CC's configured as a NAT. Eucalyptus exploits software bridges and Xen's virtual Network Interface Cards (NIC) to build virtual networks: when a security group is firstly created, i.e. when the first instance of a security group is allocated, Eucalyptus tags the physical NIC with the security group's VLAN tag and creates a software bridge for each physical machine; the bridge is actually created in the management virtual machine which starts at the boot of a physical machine. Such machine is usually named Dom0 in Xen context. The tagging process creates an abstract NIC to which tagged traffic is forwarded; such interface is then attached to the software bridge. Since Xen creates a new pair of "connected virtual ethernet interfaces", with one end

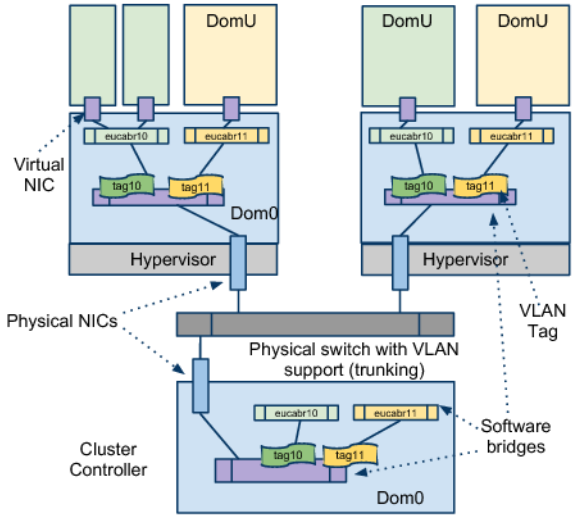

Figure 2. Eucalyptus managed mode networking: infrastructure view

of each pair in the virtual machine and the other end within Dom0, each newly created instance's virtual NIC that resides in Dom0 is attached to the corresponding security group's bridge (Figure 2). Access to security groups is controlled by the CC's firewall. By default, a security group is not accessible from external networks, and allowed traffic must be specified in terms of source network/address and port number through the Eucalyptus' API. E.g., in order to host a public web server in a security group, a rule to allow HTTP traffic from any network must be specified and added to the security group.

IV. Detecting Attacks in a Cloud Computing System

As stated earlier, cloud computing infrastructures have recently been the subject of technical news reports about severe attacks to several SIP-based communication infrastructures. What emerged by such reports about recent security incidents is the lack of a structured and well organized security system deployment in cloud computing infrastructures. The aim of this paper is to present the deployment experience of a production level, state of the art solution for intrusion detection (Intrusion Detection System - IDS). According to the analyzed data source, IDS can be classified in network and host based. Network based IDS analyze traffic flowing through a network segment, by capturing packets in real time, and analyzing and checking them against some "classification" criteria. IDS can be further characterized with respect to the type of detection mechanism implemented. Namely, IDS can explicitly model attacks, anomalies and unwanted behavior, thus implementing the misuse-based detection paradigm, or conversely model normal and expected events, consequently detecting as anomalous what doesn't conform to such "normality" model.

We have tried different deployment schemes, trying to exploit the advantages of distributed systems, and the inherent characteristics of a cloud computing architecture. We used a network based, signature based IDS. The employed IDS is network based since we want to deal with network-based attacks, and try to detect them by observing network, transport and application layer activity of cloud customers and external users. Furthermore, we use a signature based IDS in order to show how the careful deployment of well known, already available solutions could mitigate a severe problem in such a distributed computing framework as cloud computing. We will evaluate the tradeoffs between computational overhead and granularity of analysis, in terms of detection capabilities, percentage of total traffic analyzed, and cpu and memory consumption, as well as packet loss.

A. Building the proposed architecture

In this paper we will address the problem of deploying multiple instances of an IDS within a cloud computing system, allowing to rely on multiple security observation points. Eucalyptus, the cloud computing system used for the implementation of the experimental scenario, is characterized by the presence of a frontend, which also operates as a NAT, traversed by all traffic flowing to, and coming from, virtual hosts inside the cloud. Therefore, it is reasonable to think of an IDS installation which only relies on a single IDS installed near the frontend. Such an IDS would be able to see all the traffic related to virtual hosts hosted in the cloud, and provide a very good point of observation. The availability of such a large and significant amount of data, indeed, is obtained at the cost of high computational resources consumption. In fact, by forcing the analysis of all traffic to be performed at a single point, the machine physically hosting the IDS can be easily overloaded, losing packets, and producing inaccurate detection results. On the other hand, it is possible to deploy a network based IDS close to each of the physical machines. This configuration helps in reducing the load each IDS is subject to, thus helping to overcome the issue of packet loss. In fact, even though a virtual host can be subject to a Denial of Service attack, IDS's installed on other physical machines will not be affected, therefore preserving their detection power intact.

1) Snort: The IDS we chose to deploy in the proposed architecture is Snort [10]. Snort is a popular signature-based network intrusion detection system, mainly implementing the misuse based detection paradigm. Its modular architecture makes it easily extendable, and has fostered the integration of anomaly based detection plugins as well. In general, Snort's behavior is determined by rules, describing significant characteristics of events or specific attack signatures. Snort rules are organized in several groups, trying

to separate them both in terms of the targeted attack type or application scenario. For the sake of efficiency, rules are mainly structured in two parts: a header part, and a body part. The rule header contains information about the type of action to perform when the rule is matched. Such actions include, but are not limited to, the possibility of generating an alert. Furthermore, the header contains information about source and destination IP addresses and ports, thus allowing to apply each rule to restricted subsets of the analyzed traffic flows. The rule body, instead, contains information about the type of action to perform on packets in order to check whether they match the rule; furthermore, it can also contain byte sequences to check against the packet's payload. Typically, attack signatures are searched for in the payload of packets, and rules matching the content of such payloads are logged using one of the many available logging facilities, alongside with information allowing the identification of the traffic flow transporting attack-related traffic.

In order to identify which rules have to be evaluated for a packet, a fast multi-pattern search for the longest content string of each rule of a packets port group is performed on the packets payload. If this initial string matching algorithm finds a potentially matching rule, other mandatory fields of the rule (e.g., source and destination IP addresses) are checked and, upon success, the optional conditions of that rule are validated. This processing can include an expensive pattern matching operation which uses all the keywords of a rule and also validates their position. This two-phase approach has the advantage that not all rules need to be fully evaluated. Therefore, any deployment strategy allowing to reduce the load of each instance of the IDS, yet preserving the overall detection capabilities of the IDS ensemble, is worth investigating, since it can allow for a more effective detection of ongoing attacks.

V. Experimental evaluation

A. The reference testbed

In order to perform the experimental evaluation of the proposed attack detection scenario, we installed a testbed, depicted in figure 3 simulating the complete scenario for effective detection of SIP flooding attacks targeting hosts in a cloud computing environment. The testbed consists of six physical machines, two of these host a total of eight virtual machines, managed by Eucalyptus. The Eucalyptus controller plays the role of both a NAT traversed by traffic flowing to and from virtual hosts within the cloud, and as a cloud management station, controlling virtual hosts deployment and working as a concentration node for traffic on the configured VLAN's. In particular, in our experiments, we used two VLAN's, or in Eucalyptus' jargon two "security groups". As depicted in figure 3, we deployed two Asterisk [11] SIP servers, one for each security group, several RTP-using agents, as well as several attack instances

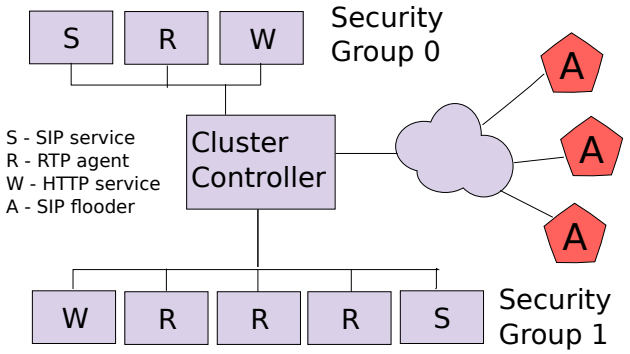

Figure 3. Experimental testbed - deployed services

generating SIP flooding traffic, implemented by using the "inviteflood" [12] tool. The aforementioned machines and services conveniently emulate the attack scenario, but are not realistic enough in themselves. For this reason we also deployed one Apache web server per security group, stressed by the hammerhead web stresser. This was useful for recreating a more realistic cloud scenario, where different services are likely to be hosted. In order to further differentiate traffic properties, we also installed D-ITG [13] as a background traffic generator. D-ITG inject traffic with specific properties in communication channels between a sender and multiple receivers; traffic can be of different types, and several properties such as inter packet time and packet size distribution can be configured. The implemented cloud consists of a machine implementing the frontend, and two physical machines implementing the physical nodes. Each of the physical nodes hosts four virtual machines, each hosting one testbed component, as represented in figure 4.

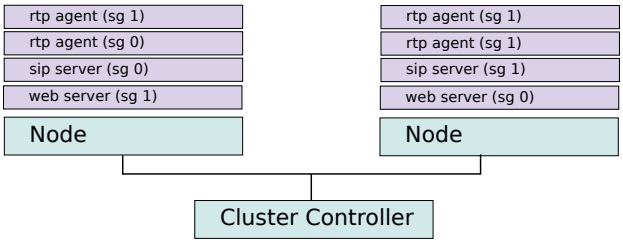

Figure 4. Distribution of services in nodes

B. Experimental evaluation

In order to evaluate the dependency of IDS performance on its position in the network, we imagined two different test scenario. In the former, we installed the IDS close to the Cloud Controller, thereby allowing it to sniff and analyze all the traffic flowing to and coming from virtual hosts. At the Cloud Controller's side, VLAN tags are removed by the virtual bridge, as discussed in previous sections. Therefore, all traffic is visible and the correspondent VLAN is indistinguishable. In the latter, instead, a Network IDS

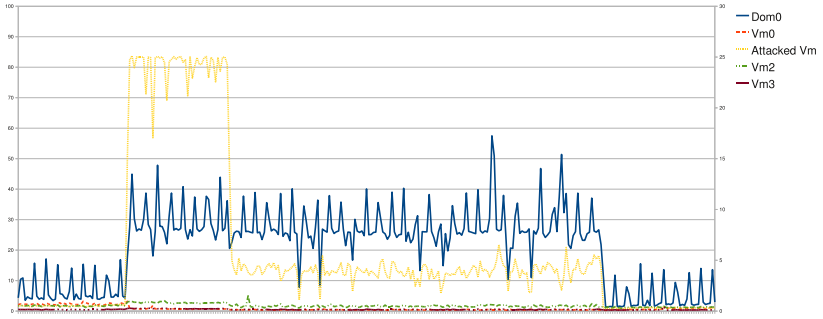

Figure 5. Attacked physical machine CPU load

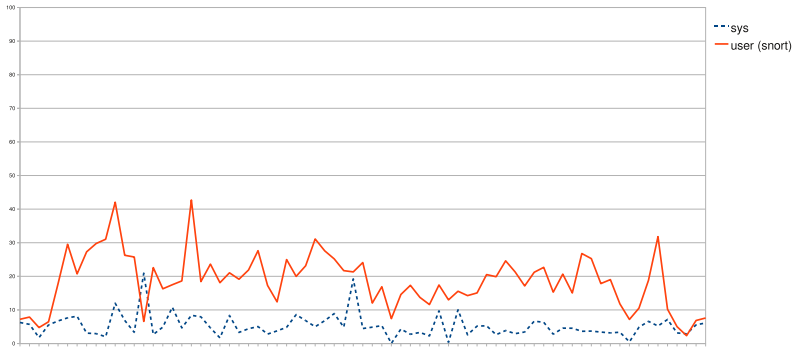

Figure 6. Cluster Controller CPU load

has been installed close to each of the two physical machines. Since each physical cloud node can potentially host virtual hosts belonging to different security groups, the IDS has to perform VLAN tag stripping, before being able to correctly analyze each packet. For each configuration, we evaluated the detection capabilities of the IDS with respect to the selected INVITE flooding attack. In both cases, the IDS's were able to correctly detect attack instances, issuing alerts communicating the result of packet analysis. Yet, it is interesting to point out how "expensive" such detection process is, showing some interesting properties and giving some insight. In particular, we observed CPU usage, in order to show whether the system hosting the IDS still has some resources to dedicate to detection during a flooding attack. Such an evaluation is useful since modern attacks consist very often in coordinated actions aimed at hitting big targets and totally disrupting networks and services. Furthermore, in the case of cloud computing, where the typical customer can be a company outsourcing service hosting, unfair competition between enterprises can become a good motivation for perpetrating dramatically effective attack campaigns. The first thing to point out in the tests' result discussion is that in both scenarios we were able to detect that a SIP flooding attack was in act. We are confident that such a result is caused by the relatively small impact of the attack itself, which was indeed able to saturate the resources of the SIP server, but not our cloud's physical resources. In figure 5 we show the CPU load of the physical machine hosting the virtual machine containing the SIP server under attack. The graph clearly shows a significant increment in CPU usage due to both the presence of the virtual machine and of the administrative domain (Dom0). This remains true until the Dom0 is able to perform both packet forwarding actions (i.e. forwarding packets from the physical NIC to the virtual machine's virtual NIC) and packet analysis through the IDS. When Dom0 reaches its physical performance limit, it is no more able to forward packets to the attacked virtual machine, and that's why such machine's CPU load decreases significantly. Clearly, the shown Dom0 performance pattern is caused by the absence of countermeasures subsequent to attack detection. It is worth pointing out that during the attack, other virtual machines running concurrently with the attacked one underwent a performance degradation, because of the Dom0's overload. On the other hand, the second physical machine was totally unaffected by the attack. Looking at the second scenario, where the IDS is deployed close to the Cluster Controller, we must take into account that each performance degradation is reflected on the entire cloud. Figure 6 shows the impact of running Snort co-located with the Cluster Controller. The CPU "system-level" load is caused by the packet forwarding activity, while the "user-level" load is mainly caused by the IDS's activity. During

the attack the IDS uses a double amount of CPU time with respect to the system's CPU time, even though our attack instance is not very powerful. Since an overloaded Cluster Controller is a bottle-neck for the cluster, we should avoid to add such big load on it. Even installing the IDS on a separate machine next to the Cluster Controller would result in an overloaded machine, since it should analyze all the traffic, therefore being prevented from operating properly.

VI. Related works and Conclusion

The application of IDS in cloud systems is a new research field that is gaining interest due the spread use of cloud computing services and the increasing number of both attacks targeting cloud services and originating from inside a cloud computing infrastructure, exploiting it as an infrastructure for deploying attacks. Due to the young age of this research field there a few papers on the topic. Current researches are mainly targeted at defining a new IDS model that can take advantage from additional information provided by the Cloud Infrastructure itself. In [14] an example of a distributed IDS for cloud environments is presented. The proposed IDS is designed to work in a Cloud system providing services according to the Platform as a Service paradigm, and is structured as an added service of the cloud system's infrastructure. Further works can be found as applied at computational GRIDs. In these works the reference system architecture is somewhat different from Cloud, but there are also some similarities, and hence there are solutions that could be applied in the cloud context as well. E.g., in [15] a solution based on the analysis of data gathered from traditional IDS and monitoring systems (e.g. Snort) deployed in the GRID's network is described.

In this paper we have shown the practical implementation experience of different deployment strategies of a well known IDS in a cloud computing system, in order to provide a fast and cheap solution to the intrusion detection problem in cloud environments. Depending on the deployment choice, several benefits and shortcomings have been pointed out and discussed. A single IDS could be placed close to the Cluster Controller, being able to monitor all traffic flowing to and from the cloud computing infrastructure. In this scenario, the single IDS is heavily overloaded, thus allowing for coordinated attack actions to disrupt the IDS's functionality by means of specifically crafted traffic before starting the real attack. On the other hand, by deploying IDS's next to the physical machines, each IDS would be able to control a smaller portion of traffic, thus being hardly overloaded. This deployment scenario needs a properly designed correlation phase in order to gather meaningful information from different security tools spread across the monitored network, which will be the subject of future work. The choice of the best deployment strategy, obviously, depends on the characteristics of the application scenario, and on the administrator's and users' requirements.

VII. Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 224263 (OneLab2).

References

[1] "Amazon ec2 sip brute force attacks on rise," http://www.voiptechchat.com/voip/457/amazon-ec2-sip-brute-force-attacks-on-rise /.

[2] "Sip attacks from amazon ec2 cloud continue," http://www.voiptechchat.com/voip/538/sip-attacks-from-amazon-ec2-cloud-continue /.

[3] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," in *CCS '09: Proceedings of the 16th ACM conference on Computer and communications security*. New York, NY, USA: ACM, 2009, pp. 199–212.

[4] "Eucalyptus web site," http://www.eucalyptus.com/ .

[5] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus open-source cloud-computing system," in *Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID '09)*, May 2009, pp. 124–131.

[6] "EC2 web site," http://aws.amazon.com/ec2/ .

[7] "KVM web site," http://www.linux-kvm.org .

[8] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," vol. 37, no. 5. ACM New York, NY, USA, 2003, pp. 164–177.

[9] "Vlan tagging," IEEE 802.1Q .

[10] M. Roesch, "Snort: Lightweight intrusion detection for networks," in *Proceedings of the 13th USENIX Conference on Systems Administration (LISA-99), Seattle, WA, USA*. USENIX, Nov. 1999, pp. 229–238.

[11] "Asterisk open source telephony project," http://www.asterisk.org/.

[12] "Sip invite flooder," http://www.hackingvoip.com/tools/inviteflood.tar.gz.

[13] "Distributed internet traffic generator," http://www.grid.unina.it/software/ITG/.

[14] K. Vieira, A. Schulter, C. Westphall, and C. Westphall, "Intrusion detection techniques in grid and cloud computing environment," *IT Professional*, vol. 99, no. PrePrints, 2009.

[15] B. C. Stuart Kenny, "Towards a grid-wide intrusion detection system," in *Proceedings of the European Grid Conference (EGC2005), pp. 275-284, Amsterdam, The Netherlands, February 2005.*