Network-based Intrusion Detection in Eucalyptus Private Cloud

Bhavesh Borisaniya, Avi Patel, Reema Patel
Computer Engineering Department
Sardar Vallabhbhai National Institute of Technology
Surat, India
Email: {borisaniyabhavesh, avi2687, reema.mtech}@gmail.com

Abstract—Cloud computing technology is emerging as a prominent platform for providing scalable, virtualized and on-demand services over the Internet. It is a matter of fact that as the technology around Internet evolves; threats to that technology also increases due to its open access. Network based Intrusion Detection Techniques have been used to overcome threats in network systems over the years.

This paper proposes a scheme to enable effective Intrusion Detection in Cloud Computing environment. We have analyzed the working of an NIDS system and Snort, in a Eucalyptus private cloud environment.

Keywords—Cloud Computing; Intrusion Detection; Private Cloud; Eucalyptus; Snort;

I. INTRODUCTION

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [1]. It is a general term used for anything that delivers the hosted services over the Internet. These services are broadly classified into three categories: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS).

On the basis of deployment, cloud can be classified as private, public or hybrid [1]. Cloud can be considered to be Private if the infrastructure is solely operated for an organization, or Public if the infrastructure is made available to the general public or a large industrial group and is owned by an organization, selling cloud services. Hybrid cloud is one in which cloud infrastructure is a composition of multiple clouds (private or public) that remain unique entities but are bound together by standardized or proprietary technology, enabling data and application portability.

Cloud is an integration of many known technologies [2]. Each technology or software may have some bugs or vulnerabilities, which make them an easy target for attackers to get into the system. Without a suitable intrusion detection mechanism, cloud users may not be able to assure that the service is thoroughly secure which may, in turn, affect the user’s trust on the provider. Cloud user should not only be able to know which attacks it suffers from but also if it is being used as a victim to attack other hosts [2]. This can prove as an alarm for users/providers to take an appropriate and timely action against the threat.

Network-based Intrusion Detection System (NIDS) is a process of inspecting the network traffic for any illegitimate activity which can cause violations of computer security policy. Generally, NIDS uses signature-based detection technique to filter the intrusive traffic. With this technique, the system tries to match each incoming network packet with a set of predefined patterns of known attacks, also called signatures. If a match is found, the system triggers an alert. Snort [3] is an open source example of NIDS which uses signature based technique for detecting network attacks. Snort sniffs network data packets and examines them for content that matches predefined known attack patterns. There are many IDS products available like ISS RealSecure [4] and Cisco IDS [5]. Also there are some free open source IDS projects like Prelude IDS [6]. We have chosen Snort because it is configurable, free, widely used, can run on multiple platforms (i.e. Linux, Windows) and constantly updated.

This paper discusses a technique of using NIDS in a typical cloud computing environment providing IaaS. In our approach, we have considered a setup consisting of Eucalyptus [7], as a framework implementing a private cloud, and Snort as the NIDS. Here, we provide a mechanism by which a user can not only verify the security of the service but also get to know the type of attack and its source.

The rest of the paper is organized as follows: section 2 describes literature review and related work. Section 3 explains Eucalyptus Cloud architecture. Section 4 discusses packet flow in Eucalyptus private cloud. Section 5 discusses our approach to deploy NIDS in private cloud. We present our conclusion in Section 6 and references at the end.

II. RELATED WORK

There has been relevant work done in the field for IDS for cloud computing. The major approaches are listed as follows:
Sebastian Roschke, Feng Cheng and Christoph Meinel [2] points to the need for deploying IDS in the cloud by proposing extensible IDS architecture which can be used in a distributed cloud infrastructure.
Noah Guilbault and Ratan Guha [8] shows a way for designing and implementing distributed grid based IDS using virtual servers deployed on Amazon.com’s Elastic Compute Cloud service.

Kleber Vieira, Alexandre Schulter, Carlos Westphall and Carla Westphall [9] have described an intrusion detection system based on Grid and cloud computing system which can identify unknown as well as known attacks.

Realizing the need of IDS as mentioned in these approaches we have attempted to design an approach for placing a NIDS in cloud computing environment using Eucalyptus as an example cloud computing framework.

III. EUCLYPTUS-BASED PRIVATE CLOUD

Eucalyptus is an open source software framework for cloud computing which implements Infrastructure as a Service, giving users an ability to run and control virtual machine instances, deployed across variety of physical resources [10]. It is an acronym for “Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems”.

The architecture of Eucalyptus system is flexible and modular with hierarchical design (Figure 1). It gives its users control over an entire virtual machine by using an emulation of Amazon EC2’s [11] SOAP and “Query” Interfaces i.e. users of Eucalyptus interact with the system using similar tools and interfaces that they use to interact with Amazon EC2.

Fig. 1. Eucalyptus Cloud Architecture

Eucalyptus system is divided into five components: node controller, cluster controller, storage controller, walrus and cloud controller [10][12].

Node Controller (NC): It is the component which resides on the physical resources. It hosts the VM instances and controls its execution, inspection and termination.

Cluster Controller (CC): It monitors available resources and schedules VM execution on specific node controllers. It also manages virtual instance network.

Walrus: It is a data storage service to utilize standard web service technologies (Axis 2 and Mule) and implements Amazon’s Simple Storage Service (S3) interface [13].

Storage Controller (SC): It is also a data storage service, but it implements Amazon’s Elastic Block Storage (EBS) interface [14]. It provides highly available, highly reliable storage volumes that can be attached to a running instance and exposed as a device within the instance.

Cloud Controller (CLC): For users and administrators it is the entry-point into the cloud. It gathers information about available resources from node managers, take high-level scheduling decisions and put them into actions with the help of cluster controllers.

Eucalyptus provides a functionality called security groups which acts as a firewall for running machine instances. It is a named collection of network access rules, defining which incoming traffic is delivered to instances. A user can add or remove a security group to meet his security needs. By using this, a user can open or close ports to control the inbound or outbound network traffic over it. By restricting the number of ports of an instance made open; a user can only decrease the probability of an attack to some extent. If the services running on these ports are vulnerable, then it’s easy for an intruder to exploit it. To demonstrate this we made a vulnerability scan on a running Windows Server 2003 instance and discovered the vulnerability of remote desktop service on port 3389. The results are shown in the Table I.

IV. PACKET FLOW IN EUCLYPTUS PRIVATE CLOUD

Figure 2 shows the typical setup and packet flow mechanism in Eucalyptus private cloud with its components. This experimental setup consists of one cloud controller (CLC), two cluster controllers (CC) with two storage controllers (SC), two node controllers (NC) each registered with different cluster controller and two clients (C1, C2). I1, I2, I3 and I4 are four machine instances; I1 and I2 hosted on one node and the remaining on other.

As shown in the Figure 2, it is required that CC and SC should be on a single machine, which constitutes a cluster with multiple NC’s connected with it. The cluster controller assigns two IPs i.e. private and public to each machine instance. Public IP is for external network and should be accessible from outside, while the private IP is for internal network management. NC and instance itself are not aware of public IP; they can only understand the private IP. CC is responsible for routing between these two networks. CC does this job using Linux iptables Network Address Translation (NAT) facilities by defining Destination NAT (DNAT) and Source NAT (SNAT).
rules for public IP to private IP and private IP to public IP address translation.

Figure 2 also shows the packet flow in the Eucalyptus cloud environment. As client C1 sends the packet to instance I1, packet contains the IP of C1 (i.e. 172.16.2.182) as source address and public IP of I1 (i.e. 172.16.2.210) as a destination address. The packet is then forwarded to the CC #1 through a network switch because the NC (172.16.2.205) is registered with CC #1 which hosts the VM instance having 172.16.2.210 as IP. The CC #1 then routes this packet to the NC by changing the destination address of packet from public IP (i.e. 172.16.2.210) to private IP (i.e. 172.19.1.1) of instance I1. At last NC will forward this packet to destined VM instance (i.e. 172.19.1.1). Here, CC works as a router and use the DNAT rules of iptables to convert the destination address of packet from public to private. Similarly if an instance sends a packet to external network, this packet will also go through CC. This is shown in the figure 2 as I4 sending the packet to the client C2. Here the packet contains the private IP of instance I4 (i.e. 172.19.1.4) as source address and IP of C2 (i.e. 172.16.2.183) as destination address. This packet is then forwarded to CC #2 through NC (172.16.2.207). CC #2 uses the SNAT rules and change the source address of the packet from private IP (i.e. 172.19.1.4) to public IP (i.e. 172.16.2.220) of instance I4.

V. PROPOSED APPROACH

To provide intrusion detection functionality in cloud, we need to place the Snort at a territory where it can sniff all the network packets intended to machine instances for further analysis regarding intrusion behaviour.

As shown in Figure 2, all the packets pass through CC. CC works as a router and route the packets to or from the NC registered to it according to scheduling policy. Hence, for intrusion detection in cloud, CC can be considered as an appropriate location for placing the Snort as it can monitor the network activity of each NC, registered with it. We have installed the Snort in both CC machines for our setup as shown in figure 3. Snort is configured to log the alerts into MySQL database. MySQL server can be installed in any of the machine within the network; we have installed it in CLC.

Snort will match the packets flowing through CC related to machine instances hosted by NC(s) with signature database of known attacks, generates alerts and log it to database stored in MySQL server.

We have developed a web application which retrieves alerts’ details from the database and accessible through an instance. When anyone accesses it from inside an instance, web page will display the alerts logged for that particular instance.

VI. EXPERIMENTS AND ANALYSIS

We have done two experiments to check whether the Snort is able or not to capture the attacks targeted to running machine instances. In both experiments, we ran the Snort in intrusion detection mode at both CC. In first experiment, we scan the ports of a running machine instance using Nmap [15]. This is considered as an intrusion activity by the Snort and hence it logs the alerts to the database. The owner of a running instance can see these alerts by accessing the web application. Figure 4 shows the page, showing alerts, generated for that particular instance.

In another experiment, we build a custom packet using Scapy [16] and send it to a running instance. To generate and send the packet, we used following command in Scapy:
On the other side, in order to capture this packet we have written the Snort rule in its configuration file.

```
sr1(IP(dst="172.16.2.210")/TCP(dport=80, flags="S")/"badpacket")
```

Using these configurations, we verified that Snort successfully captures the packet and generates the alert with the message defined in the rule itself (i.e. “Bad Packet from Scapy”).

To get the Snort’s performance statistics, we configured the Snort with perfprofile preprocessor and enabled the following in Snort configuration file.

```
preprocessor perfmonitor: time 60 file /var/snort/snort.stats pktcnt 500
```

With this configuration, we tuned Snort such that it profiles performance stats in a file every minute. We have launched ten machine instances of Windows Server 2003 on a single NC and run the Snort at CC. From a machine in the same network, we scanned the ports of each instance using Nmap at an interval of two minutes. From our experiments, we were able to obtain several statistics. As per the relevant information, we plot a graph which shows the number of alerts generated per unit time as well as throughput as shown in Figure 5.

![Graph](image)

**Fig. 5.** (a) Alerts per second and (b) Throughput - Mbits per second

From the Figure 5, it is clear that the Snort analyzes packets and generates alerts as and when the instance is attacked. Almost after each alternate minute, it shows high peak in the chart which refers to Nmap’s port scan attack in this case. Ideally, the peak value of bar should be same at each interval, as we scan the port at a regular interval of two minutes but it’s not as the performance stats given by the snort are not at exact interval of a minute.

**VII. CONCLUSION**

In the context of cloud computing, an IDS is absolute necessity to provide an Intrusion safe computing environment. However, this concept is still in a primitive stage and there is a scope for appropriate research and development. Proper integration of IDS in cloud will make it easier for critical application to utilize services in the cloud in a secure manner. Though, the proposed scheme is applied to only Eucalyptus system, it may be applicable for other cloud solutions with different architectures. This can be considered as a base to build the general model for applying NIDS system in cloud environment. Also, it makes room for researchers to develop a system to take preventive action against intruders in cloud environment.

**REFERENCES**