Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 130 (2018) 1025–1030

Procedia Computer Science

www.elsevier.com/locate/procedia

The 8th International Symposium on Frontiers in Ambient and Mobile Systems (FAMS 2018)

Security as a Service for Public Cloud Tenants(SaaS)

Mohamed Hawedi[a,*], Chamseddine Talhi[a], Hanifa Boucheneb[b]

[a]*Department of Software and IT Engineering, École de technologie supérieure,Université du Québec, Montreal, H3C 1K3, Canada*
[b]*Department of Computer Engineering,École Polytechnique de Montréal, Université de Montréal, Montreal, QC H3T 1J4, Canada*

Abstract

Cloud computing is a novel paradigm that is known for its elasticity and diversity in terms of the services provided to the end users. Although these services offer many benefits such as availability, cost-reduction, flexible payment plans, security, and efficiency, there are still several security concerns that are considered major obstacles preventing tenants from transferring their resources to the cloud. The security provided by the cloud service providers is still inadequate for the tenants security requirements because providers use standard security mechanisms regardless of the tenants requirements. Thus, cloud tenants cannot rely on the security provided by the cloud providers. Although, traditional intrusion detection systems have been proposed and developed by different entities including researchers, academicians, and practitioners as a solution that cloud providers can offer to their tenants to cope with security issues in the cloud, the requirements of the tenants in terms of security are not considered. It is very important, however, to consider the differences in the tenants security requirements to provide better security services. In this research, we concentrate on providing a security service that allows cloud tenants to monitor and protect their virtual environments by deploying a security mechanism that is capable of meeting their needs. The main contribution of this work is providing an efficient, flexible, and low-cost security as a service model - based on Intrusion Detection Systems (IDS) - that takes the tenants security requirements into consideration.

Peer-review under responsibility of the Conference Program Chairs.

Keywords: Cloud Computing, Intrusion Detection System, Security as a Service, Tenants Security Requirements;

1. Introduction

Cloud computing (CC) is a new paradigm that enables tenants to access a large pool of computing resources such as networks, storage, applications, servers, and services remotely over the Internet with minimal management and a minor investment. These computing resources are dynamically allocated to the tenants by utilizing data center servers and software networks. Moreover, CC enables tenants to obtain services on-demand and pay per use. CC can be categorized into different business models namely: Software as a service (SaaS), Platform as a service (PaaS), and Infrastructure as a Service (IaaS). SaaS is designed to provide application software to tenants while PaaS provides a platform that enables tenants to perform their operations. The physical or virtual devices are provided to tenants in the form of IaaS [1].

Despite these services offering many benefits such as cost-reduction and efficiency, there still are several security concerns that are discouraging tenants from transferring their resources to the cloud. In fact, security has become an

* Corresponding author. Tel.: +15145571433
E-mail address: mohamed-omran-ali.hawedi.1@ens.etsmtl.ca

1877-0509
Peer-review under responsibility of the Conference Program Chairs.
10.1016/j.procs.2018.04.143

essential requirement for cloud computing[2]. As the systems run over the Internet, services provided by cloud providers are prone to different attacks (e.g. Denial of Service (DoS), Distributed Denial of Service (DDoS), and SQL injection). The DoS and DDoS attacks have a direct impact on the service cost as the attacks aim at decreasing the availability and performance of the service by exhausting service resources such as memory, processing resources, and network bandwidth[3]. Despite this, the cloud service providers are still unable to provide a robust security mechanism as they do not have any information regarding the tenant networks topology and the application is run on the virtual machines (VMs). Basically, they are still not able to provide a comprehensive security mechanism to their tenants; and they are unable to provide robust security that meets tenants security requirements. Furthermore, some cloud service providers assign the responsibility of protecting cloud infrastructure resources to the tenants as they are free to deploy the VMs as well as choose the operating system (OS) and run any application[4]. Their duties are limited to protecting the physical security of the data-center, network infrastructure, virtualization platform and infrastructure. Thus, tenants should have their own security arrangements to overcome the lack of security provided by cloud providers.

In this paper, we propose a security architecture that offers elastic and efficient security as a service paradigm for cloud tenants. The proposed security can be offered by third parties to their tenants or it can be used by the tenants to monitor their VMs. The SaaS intends to meet the tenants security requirements, reduces cost and enhances efficiency. It is designed to be deployed on the top of the hybrid cloud. Moreover, it is a hybrid IDS that is based on signature and anomaly detection techniques. It enables tenants to overcome privacy concerns as it allows the tenants to partially have control over the required security deployed on their sides.

The remainder of this paper is organized as follows. In Section 2, we present the related work on the use of IDS in cloud computing. In Section, 3, we describe the cloud tenant requirements. The architecture of the proposed security is described in Section 4. Section 5, we explain our experimental setup and discuss the results of our experiments. We conclude the paper and present the future work in Section 6.

2. Related Work

In this section, we present some of the existing work that have been conducted to propose different security mechanisms based on adopting traditional intrusion detection in cloud environments. Due to their capability of monitoring and examining traffic, an intrusion detection system has been adopted to promote attack detection on the cloud environment. An IDS is a security tool that is designed to monitor network traffic and system logs looking for any suspicious activity.

A security architecture that provides a service model that cloud providers can offer to tenants was proposed by Varadarajan and Tupakula[4]. Their approach enables cloud providers to protect their infrastructure and tenants to have additional security functionalities. The security architecture consists of two main components: Service Provider Attack Detection (SPAD) and Tenant Specific Attack Detection (TSAD). The SPAD is responsible for receiving the tenant virtual machine traffic and enforces the security baseline policies that are required by the cloud service provider to ensure that traffic does not include malicious content. If the traffic violates the SPAD security policy, the VM would be isolated and an alert would be generated. The cloud service provider would enforce an additional security service (TSAD) based on the tenants requirements. Tupakula et al.,[5] proposed an architecture aiming to protect the tenant VM deployed on the IaaS cloud against attack. Hybrid intrusion detection techniques (Signature, Anomaly) are used for detecting suspicious activity on the tenant VMs. The IDS is deployed on top of the virtual machine monitor (VMM) to monitor all traffic that is inbound or outbound the VMM. This ensures that the traffic does not belong to an intrusion and does not contain any malicious content.

Some approaches have suggested that the tenant should have full or partial control of the security mechanism. Alharkan et al.,[6] proposed a scalable, elastic, adjustable, on demand IDSaaS framework for public cloud tenants that enables them to monitor their VMs created on their Virtual Private Cloud (VPC). IDSaaS enables tenants to control the IDS deployed. The framework is NIDS targeted IaaS and implemented on top of AWS. IDSaaS monitors and examines virtual machine traffic against different rules. It looks for certain patterns, such as matches on packet headers or data payload information by which it can decide if a given pattern is that of an intruder.

Overall, all reviewed studies have some limitations. Most of the proposed security approaches target the cloud service providers, where they can offer security to their tenants. In a nutshell, the cloud providers have full control of the security mechanism, which means that the tenants requirements, in terms of security, cannot be fulfilled. Another

significant issue is that most of the proposed work suggests that the security mechanisms should be deployed on the privilege domain (Dom0) of the VMM by which they can monitor all traffic on the VMS. But the VVM can be compromised by the cloud administrator who has privileged access to the privilege domain (Dom0) and thus has access to the tenants virtual machines. Ergo, the VMM can be compromised and can fail to provide the appropriate isolation between VMs. This leads to disclosing the content of all virtual machines. Some approaches solely rely on signature-detection-based techniques, which are limited in detecting a known attack. Thus, anomaly detection techniques are needed to overcome the drawbacks of the signature techniques. Additionally, deploying sensors in each tenant VMs to capture and monitor the traffic increases the resource consumption of the tenant VMs. This increases the tenants billing charges. All reviewed approaches have not been able to fill the gap as they do not consider the security requirements of tenants.

3. Tenant Security Requirements

Since cloud computing offers different cloud services to different tenants, the security requirements of tenants tend to be different. Thus, tenants should have security mechanisms that meet their requirements as they have to pay per use for the services offered by the cloud provider. In fact, there are different factors that influence the security requirements. The following list demonstrates these factors and the security requirements:

- The size of the organization and the services it offers: for instance, small-size tenants tend to deploy a small set of services on top of the IaaS to provide web services that require less security services. This is in contrast to what large-size tenants with a large IaaS will do. Such large tenants often use larger IaaS to offer services for a financial business.
- The number of VMs deployed and the type of the applications that run on them are considered as important factors that the security service provider should keep in mind in order to provide a better security mechanism. Lets assume that **Tenant A** deployed a tier of web servers to provide a web service for users, while **Tenant B** deploys a tier of database servers on top of the VMs. **Tenant A** may request Dos, DDOS service protection while **Tenant B** might only require a security service to monitor attacks against his application like SQL injection attack. Accordingly, designing a security as service model that meet the tenants requirements provides several advantages.

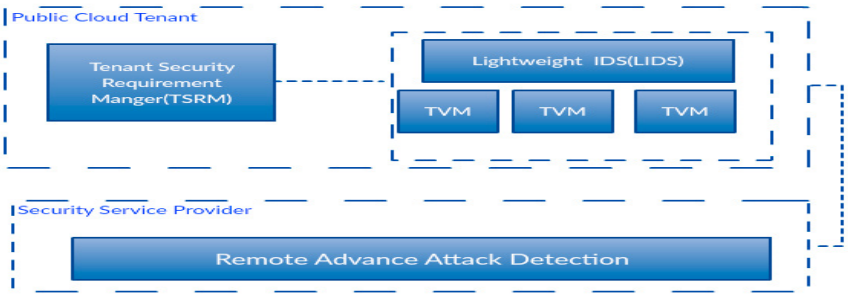

Fig. 1: SaaS Architecture

4. Overview of the SaaS Architecture

In this section, we present the architecture of the proposed Security as a Service. The idea is to build a security architecture that takes into consideration the tenants security requirements and offer a flexible security mechanism with reduced cost. As shown in figure1, the proposed SaaS architecture is composed of three main components: Tenant Security Requirement Manger (TSRM), Lightweight IDS (LIDS), and Remote Advanced Attack Detection (RAAD) that are deployed on top of the public and private cloud platform respectively.

Algorithm 1 Algorithm for allocating the security service based on the tenant's requirement

```
1: Type Element SRname,status;   ▷ SRname refers to the name of the rules-set while status refers to the rules sets status (activate or deactivate).
2: Input:
3: configList:Element[n];
4: SetrulesIndex Map < Element.SRname, Element.status >;   ▷ The rule set is the key of the index map.
5: for i = 1 to n do
6:     elt= configList[i];   ▷ elt get the actual rule that will be configured.
7:     Sr = SetrulesIndex.get(elt.SRname);   ▷ Sr the correspondent set rule in the ruleset index.
8:     if elt.status = 1 then
9:         if Sr.status=0 then
10:            activateSetRule(elt.SRname);   ▷ Activate this ruleset.
11:            Sr.setStatus(1);   ▷ update the index.
12:            write(Ruleset is Aactivated);
13:        end if
14:    else
15:        if Sr.status=1 then
16:            deactivateSetRule(elt.SRname);   ▷ Deactivate this ruleset.
17:            Sr.setStatus(0);   ▷ update the index.
18:            write(Ruleset is Deactivated);
19:        end if
20:    end if
21: end for
```

- **Tenant Security Requirement Manager (TSRM):** is a component that is responsible for getting a tenants security requirements and the topology of the tenants application by which the (TSRM) can assign the optimal LIDS that meets the tenants requirements. We have developed a new algorithm for allocating the security service (LIDSs) to the cloud tenants based on their security requirements. The proposed algorithm, that is, Algorithm 1 , takes the benefits of the classified sets of rules residing on the IDS such as snort for automatically activating and deactivating these sets (security service) to meet the tenants needs. In our algorithm, indexing tables, which employ the hash table technique[7] are used for speeding up the execution of the rule sets management.
  Algorithm Description: first, the Algorithm 1 defines the name of the security service that meets the tenants needs. In snort IDS for example, all security services are classified into different sets of rules based on the attack detection (e.g. DOS set, DDOS, and SQL set of rules, etc). Therefore, as it is shown in Line 1, SRname refers to the name of the rules set while status refers to the rules sets status (activate or deactivate). In line 3, the configList refers to the list of rules sets or security service required by the tenant to be activated/deactivated. In line 4, the indexed rule sets are illustrated as a map that contains the key (rules sets names) and the value (status of the rules sets in the IDS engine). Lines 5 to 21 execute the required actions and then update the index table. In line 6, etl refers to the actual rules sets. For instance, dos 1 means activate dos rules set while 0 is used for deactivating the rule set dos 0 as an example. Sr in line 7, represents the correspondent set of rules in the index table. Finally,depending on the the Sr status, the action is performed either by being activated (line10) or by being deactivated (line16). Then, the index is updated (lines 11 and 17).
- **Lightweight IDS(LIDS)**: built on the top of each tenants public cloud environments. The tenant can have control over the LIDS. The LIDS is a rule-based IDS that that applies all known attack signatures to provide strong security based tenants requirements. The LIDS is in charge of targeting specific threats (e.g. DoS attack) to meet the needs of tenants. As mentioned above, the tenants requirements determine the type of LIDS (e.g. DOS or SQL, etc.) that should be deployed on their environment.
- **Remote Advanced Attack Detection(RAAD)**: is an additional layer of security that is deployed on the private cloud. It is responsible for receiving traffic forwarded by a tenant for an advance inspection. Tenant is free to either forward the traffic or capture events. Basically, the traffic that is not related to the activated security services required by the tenant is forwarded to the RAAD for some inspection. Different signatures are applied by RAAD for analyzing the forwarded traffic or the captured events. RAAD first compares the captured traffic or

events against different IDSs signatures. If the traffic matches any of the signatures, an alarm will be raised and then the rule set residing in the LIDS will be updated based on this information. Machine learning techniques will then be applied to determine what activity is considered as malicious in case the traffic does not match any of the signatures. New rules will be generated based on this outcome. All the generated alerts will be saved in a database. And then, the existing set of rules resided on the tenant IDS will be updated based on the finding.

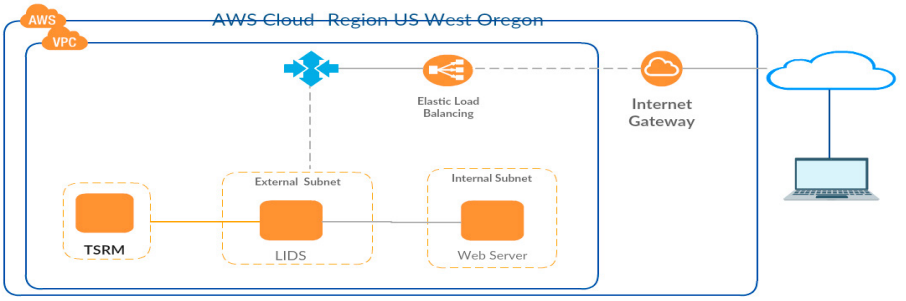

Fig. 2: Test Architecture

5. Implementation and Evaluation of SaaS

Here, we describe and discuss the prototype implementation and the evaluation of the proposed SaaS. To evaluate the proposed architecture, we develop a prototype for only TSRM and the LIDS components that will be hosted on top of the Amazon Web Services. Although our security architecture can be deployed on cloud services provided by different providers, AWS was chosen as a case study. The following assumption, which is inspired from the mentioned security requirements explained in Section 3, is considered to demonstrate the value of our proposed approach:

Assumption: let us assume that we have different tenants who have different security requirements. The current topology of the tenant involves a tier of web servers. So, **Tenant A** needs to protect his VMs against DoS while Tenant Bs topology involves a tier of databases. **Tenant B** is only looking for a SQL injection but the tenants administrator proposed an IDS that involves all signatures. TSRM will activate or deactivate the security service based on these requirements. Basically, it will activate the DoS set of rules for **Tenant A** while all signatures (sets of rules includes DoS, DDOS, Sql injection, Ftp, ICMP, and so on) are activated for **Tenant B**. Based on this assumption, several compression benchmarks are considered to evaluate the efficiency and advantage of the proposed SaaS.

Now, we describe the experiments. Our first measurement is on the capability of the LIDS in terms of performance, including the CPU performance. The main objective of this test scenario is to determine the percentage of CPU consumption of the LIDS compared to the FLIDS where all security services or signatures are loaded on the IDS, which can be unnecessary and costly.

To perform the experiments, we first built a network architecture on top of the Amazon Elastic Compute Cloud (EC2). As shown in Figure 2, the TSRM and LIDS are deployed on top of the external subnet while the tenant application is deployed in a private subnet. We created a Virtual Private Cloud **(VPC)**; later, we added two subnet internal and external subnets on top of the VPC. Then, the web server (Tomcat[8]) was placed on the internal subnet while the IDS was placed on the external subnet. **Snort**[9] , which is an open source intrusion detection system, was used to deploy the LIDS. We deployed Snort Version 2.9.6.0. The LIDS is built as NIDS where all of the VMs internal or external traffic should pass through the IDS before entering or going out of the designated VM. **Route Table:** Route Table is used to enforce the inbound/outbound traffic to pass through the IDS located in the public subnet to be examined before hitting the tenants service residing on the internal subnet. Moreover, the Security Group acts as a firewall for the deployed VMs; and it is used for controlling both that incoming and the outgoing traffics at the VMs level. We initiated the Internet Gateway which is used to enable the communication between instance residing on the VPC and Internet. Additionally, it is used to perform network address translation **(NAT)** for VMs that have been assigned public IP addresses[10]. Subsequently, the Internet Gateway is attached to the VPC.

We developed a workload network generator to measure the efficiency of the IDS in terms of CPU utilization and the packet loss results. The iperf3, which uses the concept of the server and client, was used to generate the traffic.

We generated traffic from 100mbps to 1Gbps and at each time, we computed the CPU utilization and packet loss. The generated traffic at each time was compared against a LIDS and FLIDS to determine the difference in terms of the IDS efficiency and the value of a lightweight IDS that can meet the tenants needs. **In our experiment,** the web server VM was used as the iperf server that receives traffic from a client who resides outside the cloud. The scalability features offered by AWS should be used to meet the availability. In our scalability policy, the VM should be scaled up when the CPU utilization is above 65% and scaled down if it reaches 40% or below. We wrote a Python program to compute the performance. Figure 3 illustrates the result of our experiment with the CPU utilization of the LIDS and the Full IDS deployed on top of the AWS. The X-axis shows the experimental traffic volume while the Y-axis gives the percentage of CPU consumption. It can be observed that the percentage of CPU consumption of the LIDS is roughly from 10% and it is around 30% for FLID when traffic of 100 Mbps is generated. The FIDS is overloaded when the traffic exceeds 700 Mbps while it is approximately 19% in LIDS. Figure 4 shows the percentage of the packet loss resulting from comparing the LIDS and FLIDS. The percentage of packet loss uses the Y-axis, while the traffic volume is shown on the X-axis. Overall, the percentage of packet loss of the LIDS is less than the packet loss of FLIDS.

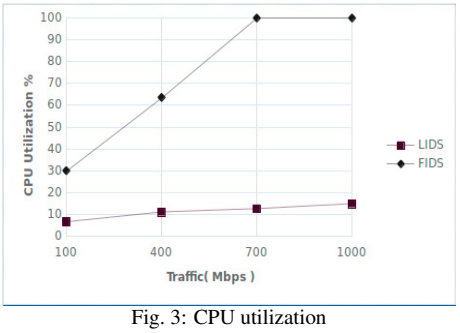

Fig. 3: CPU utilization

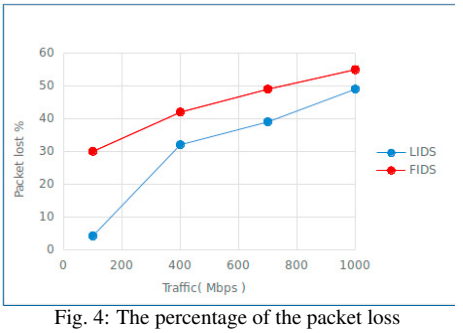

Fig. 4: The percentage of the packet loss

6. Conclusion and Future Work

SaaS is an efficient, flexible, and low-cost security as a service model based on an Intrusion Detection System (IDS) that takes into consideration the tenants requirements and designed to overcome the lack of security offered by the cloud providers. It adds a new layer of security and works in line with the security services offered by the cloud providers. SaaS enables tenants to protect their virtual private network deployed on public cloud with cost reduction and pay as you go. Currently, we are working on implementing and developing an optimized IDS that adapts to changes occurs in the tenant environment. It will be added as another security layer of SaaS. In future, we are going to apply and test our proposed algorithm with other open source IDSs such as Bro and Suricata.

References

1. A. Juels, A. Oprea, K. D. Bowers, Security issues for cloud computing [3], Meta 4 (5).
2. C. Saadi, H. Chaoui, Cloud computing security using ids-am-clust, honeyd, honeywall and honeycomb, Procedia Computer Science 85 (2016) 433–442.
3. M. Ficco, S. Venticinque, B. Di Martino, Mosaic-based intrusion detection framework for cloud computing, in: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", Springer, 2012, pp. 628–644.
4. V. Varadharajan, U. Tupakula, Security as a service model for cloud environment, IEEE Transactions on network and Service management 11 (1) (2014) 60–75.
5. U. Tupakula, V. Varadharajan, N. Akku, Intrusion detection techniques for infrastructure as a service cloud, in: Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on, IEEE, 2011, pp. 744–751.
6. T. Alharkan, P. Martin, Idsaas: Intrusion detection system as a service in public clouds, in: Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), IEEE Computer Society, 2012, pp. 686–687.
7. R. G. Bal, Ip packet filtering using hash table for dedicated real time ip filter, International Journal of Wireless and Microwave Technologies.
8. `https://tomcat.apache.org/index.html`, online; accessed:8-April-2017 (2017).
9. `https://www.snort.org/` (2017).
10. Amazon Virtual Private Cloud User Guide, `https://aws.amazon.com`, online; accessed:8-October-2017 (2017).