Intrusion Detection in Cloud Computing

Massimo Ficco, Luca Tasquier, and Rocco Aversa
Dipartimento di Ingegneria Industriale e dell’Informazione
Second University of Naples (SUN)
Via Roma 29, 81031 Aversa, Italy
Email: massimo.ficco, luca.tasquier, rocco.aversa@unina2.it

Abstract—Cloud Computing represents both a technology for using computing infrastructures in a more efficient way, and a business model for selling computing resources and services. On the other hand, such complex and distributed architectures become an attractive target for intruders. Cyber-attacks represent a serious danger, which can compromise the quality of service delivered to the customers. In this paper, we investigate the key research topics for supporting distributed intrusion detection in Cloud environments. Moreover, we present a distributed architecture for providing intrusion detection in Cloud Computing, which enables Cloud providers to offer security solutions as a service. It is a hierarchical and multi-layer architecture designed to collect information in the Cloud environment, using multiple distributed security components, which can be used to perform complex event correlation analysis.

Keywords—Cloud computing, intrusion detection, event correlation.

I. INTRODUCTION

In order to support the pay-by-use business model of Cloud Computing, the Cloud infrastructure has to continually adapt to changes of customer demands and operation conditions. Such a model involves service oriented paradigms, multi-domains, multi-tenancies, on-demand elasticity, and multi-user autonomous administrative infrastructures, which are prone to cyber attacks. Specifically, Cloud may suffer from several vulnerabilities at different architectural layers (infrastructure, platform and application), which are due to design, programming, or configuration errors of developers and service providers. Such vulnerabilities can be exploited by malicious users that can compromise the assessment of the contracted Quality of Service (QoS).

It is clear that the Service Level Agreement (SLA) availability and performance parameters alone are not enough to ensure the satisfaction of service delivery. By incorporating security parameters in the SLAs can improve the QoS of the service being delivered. This requirement has profound implications in the security solution to be implemented and delivered to protect the services provided in the Cloud.

Although several distributed Intrusion Detection System (IDS) solutions have been proposed to face security aspects in large scale networks [1], their utilization in Cloud Computing is still a challenging task [2]. Specifically, the security of the Cloud systems against cyber attacks, is hard to achieve for the complexity and heterogeneity of such systems. Moreover, the presence of different kinds of users lead to different security requirements. An important issue for the Cloud customers is the missing full control over the used infrastructure. They need to know if the service performance degradation is due to an attack against their applications, as well as if the acquired resources and services have been compromised to penetrate other hosts. Furthermore, each customer can require different types of security information, probes, configurations to monitor their virtualized resources and applications. Therefore, the Cloud providers have to offer to customers ad-hoc solutions for protecting the integrity and availability of their Cloud systems. On the other hand, the Cloud providers also want to protect both the virtualized components and the underlying infrastructure against cyber attacks (e.g., Denial of Service, Buffer Overflow, IP spoofing). These attacks can be performed by external attackers and insiders (e.g., malicious customers, deployed applications). Cloud providers need to know if their infrastructures are being used by attackers to penetrate other victims.

Moreover, in order to cope with the resource capacity limits of a single Cloud provider, as well as to address the vendor lock-in problems associated to the choice of a single proprietary Cloud solution, the concept of federating multiple heterogeneous organizations is receiving an increasing attention by the key players in the Cloud services market. The effects of attacks can span from the loss of some data to the potential isolation of parts of the federation [3]. Protecting the federated Cloud against cyber attacks is a key concern, since there are potentially significant economic consequences for any vulnerability to these attacks.

This paper discusses some key research challenges for supporting intrusion detection in Cloud Computing federated environments, and presents a short survey limited in scope to a subset of representative solutions proposed in the literature. Moreover, an extensible intrusion detection framework is presented. It can be used by Cloud providers in order to implement complex correlation process for detection of cyber attacks against their Clouds, as well as by the customers to monitor their distributed applications and virtualized systems.
The security framework enables to implement distributed intrusion detection architectures. It consists of a collection of components hierarchically organized. Specifically, security probes collect streams of information at different architectural levels (hypervisor, infrastructure, platform and application). The collected information are correlated and used to discriminate whether the monitored activities in the customer’s systems are due to malicious behaviors. Moreover, abstract security data can be derived from raw information and forwarded at higher level. They can be used by Cloud providers for further analysis (to detect distributed attacks in their federated Clouds).

The presented framework is an open-source solution, which provides APIs and interfaces to develop the distributed security components and build customized event correlation rules.

The rest of the paper is organized as follows: the main challenges that have to be faced to development of IDS for Cloud Computing are presented in Sect. II; Sect. III presents the related work in the field of intrusion detection in Cloud; the proposed IDS architecture is presented in Sect. III; Sect. IV-A presents the implementation details; and Sect. VI presents the conclusions and future work.

II. CHARACTERISTICS AND CHALLENGES OF CLOUD COMPUTING SYSTEMS

Although several distributed Intrusion Detection Systems (IDSs) have been proposed to monitor and protect large-scale networks, their utilization and deployment in Cloud Computing faces many difficulties and is still a challenging task [5]. The security of applications and services provided in Cloud, against cyber attacks, is hard to achieve for the complexity, heterogeneity, and dynamic of such systems. In particular, some challenges that have to be faced by developers, during development and deployment of IDS for Cloud Computing, are presented:

• In traditional distributed system, due to static essence of the monitored infrastructure, the security policies tend to be static, or at least rarely change over time. In a Cloud infrastructure, the monitored virtual networks and nodes are dynamically changed, added, and removed, and their security requirements tend to be different [6].

• Today the main limit to Cloud adoption is related to the perception of a security loss the customers have. Customers have security requirements, which have to be granted by Cloud providers. Therefore, the Cloud provider should offer service and tools to assess and monitor the implemented security requirements [2].

• The recent researches have provided evidence that most of the intruders come from insiders. Therefore, Cloud providers need to know if their systems and infrastructure is used by ‘legitimate’ users to penetrate other Cloud victims [7].

• In traditional systems, the security policies are usually established and managed by a security administrator responsible for the whole system. The Cloud federation has several system security administrators, which adopt different policies and mechanisms to protect their Cloud infrastructure.

• The shared infrastructure and virtualization technology put more vulnerability on Cloud Computing [10].

• Finally, lack of collaboration among different Cloud providers for detection of attacks is another drawback to current proposed approaches. Creating comprehensive distributed database to be used for detection issues is another major requirement in order for IDSs act as a comprehensive defense mechanism in the federated Clouds.

Both Cloud providers and customers will benefit significantly if there is a comprehensive IDS that evolves on the base of their requirements. The IDS components have to be easily integrated in different layers of the Cloud environment and scale without losing any functionality. Therefore, proposing a collaborative and distributed framework that considers the different Cloud security requirements is our motivation in this work.

III. RELATED WORK

In recent years, several research works that propose intrusion detection solutions for Cloud have been proposed [11].

For the operating intrusion detection in the Cloud infrastructure layer, Tupakula et al. [12] proposed a model based on a virtual machine monitor to protect from different types of attacks in the infrastructure layer. A virtual machine monitor solution embeds as a software layer to control the physical resources. The virtual machine monitors have complete control of the system resources and visibility of the internal state of the virtual machines. This model has not presented any solution to heal the system if part of the infrastructure collapsed due to a high severe attacks over the system.

Gustavo et al. [13] implemented several anomaly-based intrusion detection techniques, and presented an IDS for a reasonably complex Web application designated as SaaS. Vieira et al. [14] proposed a Grid and Cloud Computing Intrusion Detection System (GCCIDS), which uses integrated misuse and anomaly techniques to detect specific intrusions. The authors used Artificial Neural Network (ANN) to train the system. The drawback of these solutions is that they only can detect specific intrusions.

Xin et al. [16] developed a collaborative IDS with a central management approach. It is not a scalable solution, since the performance decreases with an increase of data load into the central manager node. In addition, the central manager is a single point of failure which is not appropriate in Cloud Computing.
Colajanni et al. [17] proposed a distributed architecture suitable for monitoring the IT infrastructure of large organizations consisting of multiple departments, each with multiple network segments. It combines hierarchical and peer-to-peer communication schemes in order to highlight the network security alerts by using a distributed alert ranking scheme. Even if this is a scalable solution, it is focused only on network attacks (the alert infrastructure uses NIDS for attack detection). Also the multi-tiered architecture takes into account only the clustering of sub-networks, that is a restrictive aspect with respect to a complex scenario such as the intrusion detection in Cloud Computing.

Dhage et al. [15] proposed an individual IDS structure for each user of Cloud services. In this structure, there is a single controller to manage the instances of IDSs, which employs the knowledge base and Artificial Neural Network techniques, using pattern matching of multiple false login attempts and access right violations. Their proposed structure suffers from the lack of scalability and sensitivity of central manager failure. A similar solution, proposed by Ficco et al. [18], proposes an Infrastructure as Service (IaaS) enriched with ad-hoc virtual machine for protecting the user resources against a set of attacks. The goal is to enable the customer to negotiate with the provider the level of security offered, so that the service acquired can be protected against a specific set of attacks, by additional security components offered by the service provider. The drawback is that a dedicated node is needed for the protection of each Cloud customer.

Chi-Chun et al. [19] propose a federation defense approach, in which the IDSs are deployed in each Cloud region. The IDSs cooperate with each other by exchanging alerts, and implement a peer-to-peer (P2P) judgment criterion to evaluate the trustworthiness of these alerts. Majority vote method is used for judging the trustworthiness of an alert. However, at the state of art, there is no real implementation of the proposed approach. A similar, solution is proposed by Kholidy et al. [21]. They support P2P network architecture and hybrid detection techniques, using network and host based detection, which provide a flexible and robust solution for Cloud Computing. However, although their systems are scalable, it is not sufficient for detecting large scale distributed attacks on Clouds. There is no central correlation handler to amalgamate all the alert information consistently to detect intrusions.
IV. The Distributed Intrusion Detection Architecture

In this section, we provide an overview of the proposed intrusion detection framework for Cloud Computing environments. The framework consists of three main components: Probe, Agent, and Security Engine.

- **Probes** work as detection elements. Each probe monitors a set of security parameters at a different Cloud architectural level (hypervisor, infrastructure, platform and application).
- **Agents** are software components that forward the security data collected by Probes to the Security Engines. Moreover, to decouple the Security Engine from the specific format of the Probe messages, Agents perform a normalization process that enables different kind of Probes to generate events using an unique language.
- **Security Engines** are enabled to correlate multiple streams of event data in real-time. They decide if the received information represent a potential attack pattern on the base of specific correlation rules.

As Fig. 1 shows, Security Engines are hierarchically organized according to three architectural layers. For each layer, they perform correlation at a different level of abstraction, and forward the aggregated data to higher layer. In particular, at lowest layer, the Security Engines correlate raw security data collected by security probes that can be offered by the Cloud provider as service, including IDSs (e.g., Snort, FSecure Linux Security, and Samhain), log analyzers (for syslog, flat files, etc.), and specific security mechanisms provided by the Cloud platform. Such probes can be configured to collect specific information on the base of the customer security requirements. At the higher level, the Cloud provider is responsible to enable additional IDSs, and attach them to independent VMs or physical machines. For example, Network-based IDS (for monitoring the infrastructure and platform layers), Host-based IDS (e.g., one for each physical machine), and specific Probes to collect complementary information (e.g., by accessing the hypervisor). By correlating such information with data collected by the Security Engines of lower layer, the provider can recognize compromised virtual components of their customers, as well as distributed attacks against his Cloud. Finally, Cloud provider is enabled to share their knowledge (security information about detecting malicious activities) with his peers. Therefore, at highest layer, Security Engines are used to monitor if the Cloud provider resources have been compromised to penetrate the resources of other Cloud providers, as well as to detect large-scale distributed attacks against the federated Cloud systems. However, we have to clarify that enable coordinated security services inside the Cloud federation requires that each provider has to share Cloud-related security information with the federated Cloud providers [20].

Finally, each layer can include several Security Engines. Specifically, in an environment subject to malicious faults due to intrusions, a single Security Engine can be attacked or corrupted. Thus, to provide intrusion tolerance, it is crucial to replicate the Security Engine. We suppose that the Security Engine is replicated in $n$ virtual nodes with $n \geq 2f + 1$, in which at most $f$ can be subject to Byzantine failures. Data collected at low level are forwarded to all redundant Engines. However, maintaining replica consistency in an environment subject to malicious faults is a complex problem. For this reason, we adopt a secure reliable group communication approach, which ensures that all the Engines process the same alerts, even if some of them behave maliciously. In this work, we assume the existence of an underlying group communication system as that proposed by Correia et al. [22], which provides the membership and the communication services. The former service allows to keep an updated list of the group members, process joins and leaves of group, and to detect the failure of the Engines. The latter, provides primitives for reliable and order multicast data transmission in the group. As a consequence of having redundant Engines, it is paramount that the non-faulty Engines agree on valid output data in the presence of the faulty Engines. Thus, the output must be based on cross-comparison of available replicas. Voting is used to resolve any difference in redundant output and to arrive at a consensus result based on the output data of perceived non-faulty Engines in the layer.

A. Security Engine

Each Security Engine acts as a Security Information Event Management (SIEM) system, which collects, normalizes, and correlates security-related events. Specifically, each probe reports events in centralized fashion using a secure connection to a local Security Engine. Each Security Engine processes the received events and delivers them to both a specified media (mysql database, postgresql database, XML file), and the higher level Security Engines (Fig. 1).

Security Engines are implemented by an open source solution, named Prelude, which acts as an event bus between the Agents and the Security Engines [23]. Security events are normalized thanks to a single format, called the ‘Intrusion Detection Message Exchange Format’ (IDMEF - RFC4765), which is an international standard created to enable interacting with the various security tools currently available on the market [24]. Prelude provides a library (called Libprelude), that guarantees Transport Layer Security (TLS) connections both between Agents and Security Engines, and between higher Security Managers. Moreover, Libprelude provides the necessary functionality for emitting IDMEF events, and for automating re-transmission of data in times of temporary interruption of one of the system components.

Moreover, a Prelude console has been extended to implement the Web Monitor used to view the detected attacks.
The Monitor is the central analysis server, which can be used by the security administrator to monitor the Cloud system status, and identify the best action to take in order to mitigate the intrusion effects on the target components.

B. Monitoring Features

According to the presented multi-tier system, the framework allows to monitor the Cloud at different levels. The monitor architecture is depicted in Fig. 2.

The Cloud resources and services can be monitored by specific security mechanisms offered by the proposed framework, which can be installed on the hosted VMs. By using the Libprelude, it is possible to implement Agents (available in C, C++, Python, Ruby, Lua and Perl) for retrieving security events from the most common IDSs (including Snort, FSecure Linux Security, and Samhain, CISCO IPS), as well as to support several types of log (system logs, syslog, flat files, etc.). The Agents exchange the collected data by using the IDMEF. The purpose of IDMEF is to define data formats and exchange procedures for sharing information of interest to intrusion detection and management systems. Using Libprelude it is possible to register an Agent by the following command:

```
prelude-admin register snort "idmef:w admin:r" localhost
```

Moreover, specific security features are offered by the proposed framework. In particular, on the basis of the security requirements of the customer, it is possible to monitor the user application features (such as application throughput, the number of failed query to the database, the access rights violations, the failed authentication requests, the number of HTTP requests to Web applications). This feature needs the installation of the mOSAIC platform on the Cloud infrastructure [25]. In mOSAIC, a Cloud application is modeled as a collection of components, which are able to communicate each other, and consumes Cloud resources. mOSAIC provides a set of APIs and a Software Platform, which is a distributed environment that offers a single homogeneous interface by which the components of the mOSAIC Application run. The Software Platform runs on the top of VMs, using a dedicated environment: the mOSAIC Operating System (mOS), which is a lightweight Linux distribution, customized in order to run the mOSAIC components. The Software Platform coordinates clusters of mOS machines distributing data components over the VMs in a completely transparent way. mOSAIC API offers a simple and flexible way to develop User Components, also
called Cloudlets (a name inspired by the Java ‘Servlet’). For monitoring purposes, a set of pre-configured Cloudlets is provided (e.g., threshold-based filters, proxy components, SLA based components), which act as security facilities for the user application [26]. However, about the monitoring of what is strictly related to the application, it has to be implemented by the developer through ad-hoc solutions. We mean that is up to the developer to instrument its application.

At the level of infrastructure, the Cloud Agency can be used. Cloud Agency is a multi-agent based system that is responsible for the Cloud resource brokering and monitoring. It has been designed and developed within the activities of the mOSAIC project. Cloud Agency is enabled to interact with the Cloud providers and the customers in order to negotiate and broker the needed resources. Moreover, it provides interfaces and API to expose management, monitoring and reconfiguration services. Cloud Agency addresses the monitoring of the Cloud infrastructure (e.g., the CPU and memory consumption of each virtual node), by automatically deploying and configuring a set of Probes on the Cloud resources. At this level, performance figures of each virtual host can be collected and used to detect anomalous conditions. The Probes are implemented by mobile agents, which are deployed on the computing resources, from which they send the collected measures to the local Decision Engine.

As for Cloud providers, additional security information can be inferred. In particular, the presented monitoring system offers a set of tools, whose goal is to evaluate the global state of both the distributed Cloud application and the Cloud resources. At lower level (physical), host- and network-based IDSs have to be installed. The IDS allows to monitor the low level infrastructure by using multiple inspection methods, such as misuse attack detection or anomaly based detection, and managing different techniques like file integrity checking, process monitoring, rootkit and malware detection, and network traffic analysis. Furthermore, visibility of activities performed by monitored virtual machines is enhanced to the system call level which encapsulates all the activities performed by a guest virtual machine.

V. EVENT CORRELATION APPROACH

As described previously, all the collected information are forwarded to the corresponding Security Engine, which locally stores and processes the received data. In particular, in order to recognize distributed attacks to the Cloud and enrich the semantic of diagnosis, a correlation approach based on Bayesian networks is adopted. It captures the causal relationships between the resulting alarms (which may represent symptoms of the same attack collected at different Cloud architectural levels), by correlating them on the base of temporal and logical constraints. It can be also used to detect complex attack scenarios that consist of specific sequence of malicious activities performed by the attacker in order to discover Cloud vulnerabilities.

The correlation approach uses two predictive parameters:
1) \( P(a|A) \): representing the probability of the occurrence of a symptom/attack \( a \) given that the set of symptoms/attacks \( A \) has been already detected;
2) \( T(a|A) \): representing the time associated to a symptom/attack \( a \) given that the set of symptoms/attacks \( A \) has been already raised. This means that in order to be related to an attack, the alert related to \( a \) must be triggered within \( T \).

The described parameters are associated to the nodes of an Attack Evaluation Tree (AET). An AET represents the attack against a system in a tree structure, delineating the goal of the attacker as the root node, and different ways of achieving that goal as leaf nodes.

![Figure 3. An example of AET](imageURL)

Specifically, AETs are founded on AND-OR trees. The offspring represent different attack sub-goals, and have to be collectively (AND-decomposition) or alternatively (OR-decomposition) performed by the attacker in order for the major intrusion to occur. For instance, Fig. 3 depicts a simple AET for attack goal \( A \), which has sub-goals \( B \) and \( C \), both of which must be achieved for attack goal \( A \) to be successful. Sub-goal \( B \) has \( D \) as leaf while sub-goal \( C \) is broken into sub-goals \( E \) and \( F \), that are in OR relation each other. Hence, attack goal \( A \) can be achieved in two ways: achieve sub-goals \( D, B, E, C \) or achieve sub-goals \( D, B, F, C \).

In order to detect malicious activities, the Security Engine processes a set of correlation rules that have to be defined by the security administrator. The rules should be specified on the base of the security requirements of the customer (e.g., the application to be protect and the potential threats to be detected).

In the following, an example of correlation rule definition is presented. In particular, as a case study, we consider an inter-Cloud distributed DoS attack based on the Coercive Parsing. It is a DDoS attack that exploits the XML verbosity and the complex parsing process (by using a large number of namespace declarations, oversized prefix
names or namespace URIs, and very deeply nested XML structures). In a previous work [27], we presented the effects of a Deeply-Nested XML attack against a simple Web application running in the Cloud. The experimental results show that during the DDoS attack the percentage of tags nested in each message is greater than the normal operation, as well as the CPU of the target Virtual host is fully committed to process the malicious messages.

In this work, we assume that some nodes of a federated Cloud A have been compromised and used to attack a Web application running on the nodes of another federated Cloud B. Therefore, in order to protect the application from an inter-Cloud DDoS Deeply-Nested XML attack, the Attack Evaluation Tree represented in Fig. 4 should be implemented.

![Attack Evaluation Tree](image)

Figure 4. AET of an inter-Cloud DDoS Deeply-Nested XML attack

Specifically, an approach to detect such kind of attack should detect and correlate the following symptoms:

- multiple sources in the Cloud A send flows $\phi$ of HTTP requests toward the same target hosted on the Cloud B;
- the target Web application is under an X-DoS attack.

An X-DoS attack is detected if a CPU overload is observed on the VMs that host the target application, and a large number of anomalous XML messages are received by the application.

Therefore, in order to implement the proposed approach, several Agents have to be enabled, as well as specific rules should be applied in the Security Engines. In particular, to detect anomalous incoming messages (which include an excessive number of nested XML tags), an application-based IDS can be adopted (e.g., based on an XML validation schema) [28]. If an anomalous behavior is observed an alert is forwarded to the local Security Engine.

Moreover, by using the Cloud Agency (CA), the local Security Engine can monitor the VMs on which the Web application is running. Each CA Agent running on the involved VMs implements the following rule:

```prolog
cpu_overuse(ID) :- event(ID,cpu_usage,average,greater,X), X > 90 .
aalert(cpu_overuse,ID) :- cpu_overuse(ID) .
```

The rule means that an overload is given by an average CPU value greater than 90%. If the verification of the rule succeeds, an alert is triggered.

The local Security Engine collects the alerts coming from the CA Agents and correlates them with the previous symptom, according to the following rule:

```prolog
alert = mag.getContent()
aalert_type = get_alert_type(alert)
int threshold = n_app_vms + 1
if (alert_type=="Coercive") or (alert_type=="CPU Overuse"):
    ctx = Context("SCAN_X-DOS",
        {"expire": 120, "threshold": threshold, "alert_on_expire": True},
        update = True)
    if ctx.getUpdateCount() == 0:
        ctx.Set("alert.correlation_alert.name", "X-DOS")
        ctx.Set("alert.classification.text", "X-DOS Attack")
        ctx.Set("alert.assessment.impact.severity", "high")
```

According to the above rule, the detection context is set to expire after 120 seconds; if 120 seconds pass without having another event about a possible attack, the context expires. The first time the event is verified, the context is filled with the information about the possible attack. A higher-level alarm is generated when the Security Engine receives an alert about a possible Coercive Parsing attack from the application IDS and the notifications of CPU Overuse coming from all the VMs that host the target application. These conditions are taken into account in the previous rule by setting the `threshold` value to the number of the VMs on which the application is running ($n_{app\_vms}$) in addition to the reception of the alert about the Coercive Parsing attack (`alert_type == 'Coercive'`).

The raised alert can be forwarded to a higher level Security Engine that correlates the X-DoS attack to other information (such as the sources of the attack, etc.), in order to discover an inter-Cloud DDoS attack.

VI. CONCLUSIONS

This paper proposes a framework to build distributed Intrusion Detection architecture in the Cloud Computing. It is an open-source solution that allows to develop and deploy security probes on the customer’s virtual resources and the Cloud infrastructure.

The implemented architecture is a first prototype of an IDS management system deployed in the Cloud. Future work will consist in designing an approach to automatically define the Security Engine correlation rules.
ACKNOWLEDGMENT
This work has been supported by PRIST 2009, Fruizione assistita e context aware di siti archeologici complessi mediante terminali mobili, founded by Second University of Naples; by the mOSAIC project (EU FP7-ICT and by CoSSMic (EU FP7-ICT-608806)

REFERENCES