CIDS: A framework for Intrusion Detection in Cloud Systems

Hisham A. Kholidy
Dipartimento di Informatica
Università di Pisa, Pisa, Italy
hkholidy@di.unipi.it, hisham_dev@yahoo.com

Fabrizio Baiardi
Dipartimento di Informatica
Università di Pisa, Pisa, Italy
baiardi@di.unipi.it

Abstract

By impersonating legitimate users, intruders can use the abundant resources of cloud computing environments. This paper develops a framework for "CIDS" a cloud based intrusion detection system, to solve the deficiencies of current IDSs. CIDS also provides a component to summarize the alerts and inform the cloud administrator. CIDS architecture is scalable and elastic with no central coordinator. This paper describes the components, architecture, detection models, and advantages of CIDS.

Key Words: cloud computing, security, intrusion detection, attacks, masquerade.

1. Introduction

Cloud computing has a broad appeal because it enables IT managers to provision services to users faster and in a cost-effective way. However, it does raise some concerns and chief among them is securing data in the cloud because of their operational models, the enabling technologies, and their distributed nature, clouds are easy targets for intruders. While intrusions can be handled by an Intrusion Detection System (IDS) [1], current IDSs have many deficiencies which hinder their adoption in a cloud environment. This paper describes CIDS, a framework for a Cloud based Intrusion Detection System to deal with attacks like: (1) Masquerade attacks: where threats impersonate legitimate users, (2) Host-based attacks: these can be a consequence of masquerade attacks and generally result in an observable user behavior anomaly and (3) Network-based attacks. CIDS also summarizes the intensive network based IDS alerts according to the attack signature and target. Section 2 briefly introduces a cloud security and the seven known top threats to cloud computing systems. Then, it classifies vulnerabilities of the cloud computing paradigm. The next section surveys the related works. Section 4 describes the components, architecture, detection models, and advantages of our CIDS framework. Section 5 outlines future work.

2. Cloud computing security

Threats of cloud computing systems differ from those of traditional IT solutions. CSA (Cloud Security Alliance)[2] ranks seven threats that apply across cloud computing SPI models [3]: (1) Abuse and nefarious use of cloud computing, (2) Insecure interfaces and APIs, (3) Malicious insiders, (4) Shared technology issues, (5) Data loss or leakage, (6) Account or service hijacking, (7) Unknown risk profile. [4] defines seven risks a user should raise before committing: (1) Sensitive data should be processed outside the enterprise only with the assurance that they are only accessible and propagated to privileged users, (2) One customer data should be fully segregated from those of another customer, (3) A customer needs to verify if the infrastructure complies with some regulatory security requirements, (4) The cloud provider should commit to store and process data in specific jurisdictions and obey local privacy requirements on behalf of the customer who do not know where data is stored, (5) The cloud provider should offer replication and disaster recovery mechanisms, (6) Investigative support needs to be ensured, (7) Data should be accessible even when the provider is acquired by another company or if the user moves to another provider.

3. Related work

IDSs may be classified according to the source of data into: (1) Host-based IDS (HIDS), where sensors that detect an intrusion are focused on a single host. (2) Network-based IDS (NIDS), where sensors are focused on a network segment. (3) Distributed IDS (DIDS) which integrates both types of sensors, DIDS can be categorized as Mobile Agent IDS (MAIDS), Grid based IDS (GIDS), and recently Cloud based IDS. Traditional NIDS and HIDS cannot identify suspicious activities in a cloud environment. As an example, a NIDS can not detect an attack anytime node communication is encrypted. Attacks can also be invisible to HIDS, because they may not leave traces in the node operating system where the IDS resides. Since in clouds, distinct users share computing and communication resources, attacks may be originated from and be directed against several resources within the cloud infrastructure. Hence, only a distributed strategy may be appropriate. The adoption of DIDS solutions [5] is still challenging in cloud computing because the complex architecture of the infrastructure and the distinct kinds of users lead to different requirements and possibilities for being secured. Some of these IDSs are scalable but they have the problem of single point of failure as most

978-0-7695-4654-4/12 $26.00 © 2012 IEEE
DOI 10.1109/ITNG.2012.94

distributed hierarchical IDS. Also, some distributed IDSs are strongly centralized and lack the flexibility to be applied to different cloud architectures. This category of IDSs can not respond to attacks against the IDS itself, another deficiency is that some IDSs use only one technique for detecting the attacks, whether, the knowledge based technique or the behavior-based one. A good IDS should integrate them to detect known and unknown attacks with a reasonable false alarm rates. Mobile Agent-based IDSs [6] are not a suitable solution for clouds, because their hierarchical structure poses both reliability and scalability problems. Furthermore, they are not flexible and can not respond to attacks against the intrusion detection system itself. Other problems are recalled in [6]. GIDS solutions in [7, 8 9, 10, 11, 12, 13,14] offer a partial or complete methodology for dealing with attacks against the processes running either inside or outside the kernel space, and Grid protocol stack and network devices. However GIDS solutions can not be adopted because of: (1) Distinct cloud service models, SaaS, PaaS, and IaaS, with different types of threats and distinct users with distinct requirements, (2) The high scalability of cloud systems, (3) GIDS solutions do not correlate the alerts coming from different nodes, and (4) The performance and the load balancing inside cloud network are higher than in grid systems. Some cloud based intrusion detection systems have recently been proposed. [15] proposes an IDS based on the Mobile Agents (MAs) technology. The most important deficiencies are the performance and the security issues related to MAs [16, 17]. [18] proposes a theoretical framework for dealing with attacks targeting any service model but it does not correlate the alerts from components in the cloud infrastructure. The analysis of previous work confirms that, a proper defense strategy for cloud systems needs to: (1) Be distributed and scalable to adapt the cloud characteristics, (2) Avoid any single point of failure, (3) Protect the IDS by isolating it from vulnerabilities in the host machine, (4) Have a flexible architecture to be applied to several cloud architectures, (5) Integrate both behavior and knowledge based techniques, and (6) consider different service models and user requirements.

4. The proposed Framework (CIDS)

CIDS has a scalable and elastic architecture with a P2P solution and no central coordinator. Hence, there is no single point of failure. CIDS architecture distributes the processing load at several cloud locations and isolates the user tasks from the cloud by executing them in a monitored virtual machine. This helps in protecting CIDS components from threats that can control a task in the VM and that can modify CIDS components. To increase attack coverage, CIDS integrates knowledge techniques and behavior based ones. Furthermore, it collects events and audits from VMs so that the detector and correlator components can analyze them. Each node also includes an audit system that monitors messages among nodes and the middleware logging system, and collects events and logs from the VMs. By sharing both the knowledge and behavior databases in each node among the audit components, CIDS can detect the masqueraders that access from several nodes and both host-based and network-based attacks. Furthermore, to take into account the large volume of data in a cloud that prevents administrators from observing any action, a further CIDS component parses and summarizes a highly intensive number of alerts from a NIDS component in a physical or virtual switch inside the cloud virtual network. A report for the administrators collects alert messages from all IDS detectors installed in the cloud system. CIDS resides inside the cloud middleware which provides a homogeneous environment for accessing all nodes. The middleware sets the access control policies and supports a service-oriented environment. Since the middleware can be install inside different grid and cloud systems, CIDS can be applied to several Grid and cloud systems.

4.1 CIDS Architecture

In the proposed architecture, each node has two IDSs detectors, CIDS and HIDS. In this way, the node can cooperatively participate in intrusion detection by identifying the local events that could represent security violations and by exchanging its audit data with other nodes. Figure 1 shows the sharing of information among the following CIDS components:

Cloud nodes: contains the resources homogeneously accessed through the cloud middleware.

Guest task: it is a sequence of actions and commands submitted by a user to an instance of VM.

Logs & audit collector: it acts as a sensor for both CIDS and HIDS detectors and collects logs, audit data, and sequence of user actions and commands.

VM: it encapsulates the system to be monitored using VMM. The detection mechanisms are implemented outside the VM, i.e. out of reach of intruders. A single instance of a VM monitors can observe several VMs.

Type II Virtual Machine Monitor (VMM): CIDS uses type II VMM [19] implemented as a process of an underlying operating system of the host machine. Some VMMs are useful in system security, among them: Isolation, Inspection, and Interposition [19]. VMM stores in the audit system the data collected by the logs & audit collector component and forwards them to both CIDS and HIDS correlator components.

The audit system: this component implements three main functions. First of all, it monitors message exchanges among nodes and extracts from them the behavior of the cloud user. Then, it monitors the middleware logging system in the node itself. CIDS can

collect all audit data and middleware events such as user's login or logout from the cloud system or tasks submissions. The third function collects and stores events and logs from the VM system. A log entry is created for each node action with the action's type, (e.g. error, alert, or warning), the event that generated it, and the message.

CIDS correlator and detector: it correlates user behaviors, e.g. sequence of commands or actions collected from several sources, and analyzes them through our new heuristic semi-global alignment approach (HSGAA). We will briefly explain later the HSGAA approach in CIDS.

HIDS correlator and detector: it correlates user's logs and signatures collected from several sources. Then it analyses them to detect known trails left by attacks or predefined sequences of user actions that might represent an attack. It is implemented by the OSSEC IDS tool [20] that receives user's logs and signatures and determines whether a rule in the knowledge based database is being broken. After that, it computes the probability that a user action represents an attack, and it communicates this to the alert system that alerts the other nodes if the probability is sufficiently high.

Behavior-based database: it is a profile history database for the behavior of cloud users. It is important that all nodes share the same behaviour database for the same user. This helps in correlating the normal behaviors of a specific user to detect a suspected behavior distributed among several nodes. Since behavior deviation in one node can be normal in another one, correlation reduces the false alarms rate and it is more suitable for adapting the deployment and utilization of the cloud system, as a user task can be executed in more than one machine. Access to all databases, including events collected by the VMM from the VMs, can be easily implemented by the middleware that transparently creates a virtual homogeneous environment and synchronizes the nodes. As an example, consider that the audit system can create a log entry such as: "User Roy only logs in for 2 to 3 hours and uses a specific sequence of UNIX commands", only if the nodes know the behavior of the user in all the nodes.

Knowledge-based database: it stores a set of rules and signatures for known attacks. It describes a malicious behavior with a rule to be compared against those in the database. Like the behavior-based database, all nodes should share or exchange the same knowledge base, through the services provided by the middleware.

Alert System: it uses the middleware's communication mechanisms to alert other nodes if the CIDS or HIDS correlator and detector components signal an attack. It also communicates its alerts to the report producer component in the scheduler machine.

Parser and summarizer: It parses and summarizes the alerts fired by a component in the cloud virtual network. We will briefly explain later, the adopted algorithm.

Report producer: it collects alerts from any cloud IDS and sends a report about attacks to the cloud scheduler. It helps also service providers to know if their infrastructure is exploited to penetrate other victims.

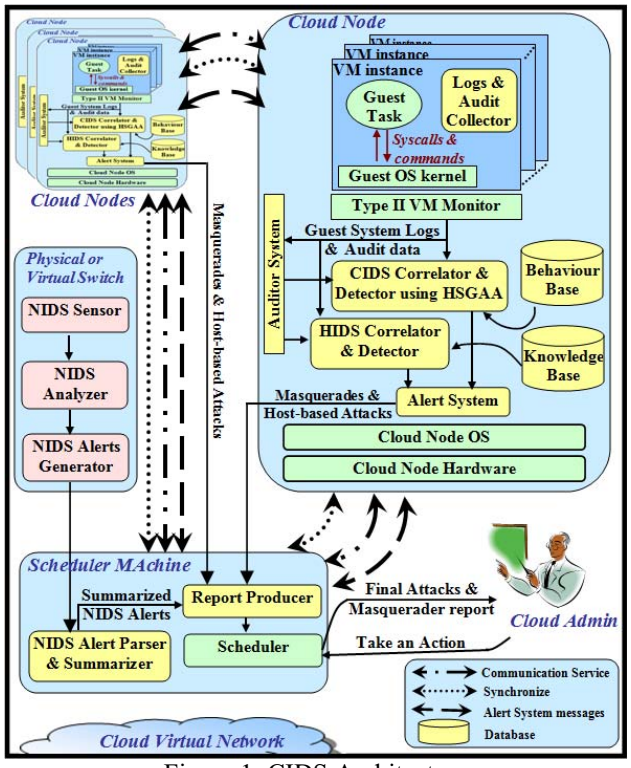

Figure.1: CIDS Architecture

Yellow components are CIDS components, Green ones are cloud system components, and Pink ones are NIDS components

4.2 CIDS deployment models

We recall the P2P architecture of CIDS helps in balancing the load among all nodes. In CIDS each node has its own analyzer and detector components that are connected to the behavior and knowledge based databases. The individual analysis reduces the complexity and the volume of exchanged data, but at the expense of the node processing overhead. As discussed in the following, our approach can reduce this overhead. The lack of a single point of failure also improves the framework attack resistibility. Some components of the scheduler machine (i.e., Report Producer and NIDS Alert Parser and Summarizer) do not represent a single point of failure. As a matter of fact, a cloud runs several copies of the scheduler node with a fault tolerance technique provided by the middleware to backup the processing data. According to the architecture of the cloud system, CIDS components can be deployed into one of two models namely, hybrid and pure P2P models. In the hybrid model, each node communicates to nodes outside its domain through its domain controller i.e., nodes in different domain are not directly connected. Whereas, in the pure P2P model, each node communicates directly to

other nodes i.e., the domain controllers acts like a peer but with a scheduler to perform the scheduling tasks.

4.2 CIDS deployment models

We recall again, CIDS framework has P2P architecture with no central coordinator, where the network load is symmetrically distributed to all nodes. In CIDS each node has its own analyzer and detector components that are connected to the behavior and knowledge based databases. This differs from distributed centralized IDSs, where a centralized management system collects all the data. The individual analysis reduces the complexity and the volume of data exchanged among nodes, but at the same time it increases the processing overhead inside a single node. We will explain later, how our HSGAA approach can reduce this overhead. Since CIDS has no single point of failure, the framework represents a moderate solution for attack resistibility. The cloud scheduler machine has some components (i.e., Report Producer and NIDS Alert Parser and Summarizer) that do not represent a single point of failure, because there are several copies of the scheduler node in the cloud with a fault tolerance technique provided by the middleware to backup the processing data. According to the architecture of the cloud system, CIDS components can be deployed into one of two models namely, hybrid and pure P2P models. In the hybrid model, each node communicates to any other node outside its domain through its domain controller i.e., no direct connection between two nodes in different domain. Whereas, in the pure P2P model, each node communicates directly to other nodes without using the domain controller i.e., the domain controllers work like the other peers but with a scheduler to perform the scheduling tasks.

4.3 Attacks and services covered by CIDS

CIDS is a defense strategy that satisfies the previous IDS requirements and deals with some attacks against SaaS, PaaS and IaaS clouds.

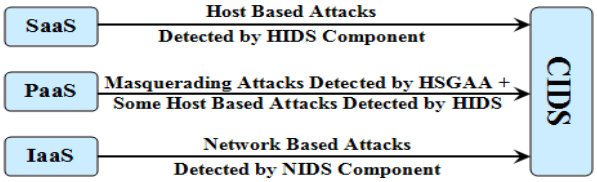

Figure.2: Attacks and services covered by CIDS.

Figure 2 shows the attacks and services discussed in the following:

1) Masquerading attacks:
This is a PaaS attack that includes any attack that impersonates a legitimate user to use service resources maliciously. This is by far the most critical attack, as its exploitation is rather easy. An intruder masquerades as a legal user by obtaining the user's password and this leaves some trails left at the service location. CIDS detects this attack through HSGAA.

Heuristic Semi-Global Alignment Approach (HSGAA): it detects masquerade attacks based on the Semi-Global Smith Waterman alignment algorithm [21]. The main idea underlying HSGAA is to compute the best alignment score, by aligning the active user's session sequence (e.g., mouse movements, system calls, opened windows titles, written Commands, opened file names) to the previous stored sequences for this user. By properly defining the misalignment areas, we can label them to be anomalous. The presence of several anomalous areas is a strong indicator of masquerade attacks. The value of HSGAA is its ability to align sequences not only using lexical matching, such as string matching or longest common substring searches. Furthermore, it allows for small mutations in the sequences with small changes in the low-level representation of the commands functionality (e.g., using *vi* instead of *cat* in UNIX command line interface).

From the computational point, HSGAA accelerates the detection and update operations, by implementing these operations in distinct threads. As a further improvement, the heuristic approach divides the signature sequence into a smaller set of overlapped subsequences to reduce the computational load of the alignment process. This also results in shorter masquerader live time inside the system.

From the security point, unlike traditional semi-global alignment algorithm that uses the same parameters for all users, HSGAA can reduce both false positive and false negative rates by pairing each user with distinct scoring parameters. This increases the hit ratio of the detection system. Furthermore, it supports two update modules that support the slight changes in the user behavior due to some project requirements or other individual considerations. HSGAA provides two scoring systems that enable changes in the low-level representation of the commands functionality, by categorizing user's commands to a set of groups and enabling the alignment of commands in the same group without reducing the alignment score.

2) Host-based attacks:
Host based attacks may be a consequence of a masquerading attack. CIDS detects several host based attacks using the current HIDS tools, which integrate both the log analysis and data mining techniques into a log mining technique. As previously mentioned, we use OSSEC [20] as an example of HIDS tools.

3) Network-based attacks and the summarizer and log analyzer component:

CIDS detects network attacks by analyzing network packets using NIDS tools. We use Snort [25] as NIDS. The summarizer and log analyzer component analyzes and summarizes NIDS alerts and logs, and sends a report

to the cloud administrator. This component is an IDS service supported in the IaaS model and works as outlined below.

Parser and Summarizer Approach: from the point of view of the cloud administrator, a clear, summarized, and readable alarm report is fundamental. A NIDS component produces an intensive number of alerts, because of the high scalability of the cloud network. Our parser and summarizer component reduces the number of alerts for the cloud administrator. Among the approaches to summarize and integrate NIDS alerts, we recall, [22, 23]. A more suitable and clear approach to store NIDS alerts is [24] that is based upon the alert parameters shown in Table 1. Our approach is based upon the idea that, if one or more hosts, are attacking the same machine using the same attack signature, we should reduce the intensive redundant alerts fired by NIDS components. This can be achieved by merging all the alerts with the same combination (destination IP, attack signature) into one alert only that also merges their attributes. Our implementation uses SNORT with MySQL. Table 1 shows an example.

Table.1: An example for the alert description table.

Alert ID	Destination IP	Signature ID	Alert Description						
			Signature Name	Signature Class ID	Signature Priority	Source IP	IP Protocol	Source Port	Destination Port
A1	1.2.3.4	11	DOS	1	3	1.1.2.3	TCP/IP	50	70
A2	1.2.3.8	13	Rootkit	3	5	1.2.3.3	TCP/IP	20	60
A3	1.2.3.8	13	Rootkit	3	5	1.2.3.7	TCP/IP	30	72
A4	1.2.3.4	11	DOS	1	3	1.1.2.3	TCP/IP	50	70
A5	1.2.3.12	13	Rootkit	3	5	1.2.3.6	TCP/IP	122	17
A6	1.2.3.4	11	DOS	1	3	1.1.2.3	TCP/IP	50	70
A7	1.2.3.12	14	Buffer Overflow	2	4	1.1.2.4	TCP/IP	152	23
A8	1.2.3.8	13	Rootkit	3	5	1.2.3.3	TCP/IP	30	82

Our approach neglects the source IP address because it can be spoofed. Spoofing can be detected by the HIDS component. However, the final summarized table would contain all information that describes the attack including the source IP address that can be used later by the cloud administrator.

Table.2: The final alerts summarization table.

Alert ID	Destination IP	Signature ID	Alert Description						
			Signature Name	Signature Class ID	Signature Priority	Source IP	IP Protocol	Source Port	Destination Port
S1	1.2.3.4	11	DOS	1	3	1.1.2.3	TCP/IP	50	70
S2	1.2.3.8	13	Rootkit	3	5	1.2.3.3, 1.2.3.7	TCP/IP	20 30	60, 72, 82
S3	1.2.3.12	13	Rootkit	3	5	1.2.3.6	TCP/IP	122	17
S4	1.2.3.12	14	Buffer Overflow	2	4	1.1.2.4	TCP/IP	152	23

Table 2 shows the final alerts produced by our approach, we note that alerts A1, A4, and A6 refer to the same signature, their attacks targets the same machine and the attacker uses the same method three times. The alerts are summarized by alert S1. The alerts A2, A3, and A8 have the same signature but with different signature details. The attackers fired these attacks from two different host machines. These alerts are summarized to alert S2 in Table 2. Finally, the attacks related to the alerts A5 and A7 target the same machine but with different signatures, hence these alerts have not been summarized. Algorithm1 shows the parsing and summarization.

Algorithm1: parsing and summarization

01: Begin
02: Build Table *T with rows= n //This table is similar to table 1.*
03: Define*:*
dest-ip=1, sig-id=2,
i=1, // Index for rows of table T
alert-dscrp-strct = T(1)(signature-name, signature-class-id, signature-priority, score-ip, ip-protocol, source-port, destination-port) // Is a structure contains one record of table T with 7 columns of alert description (from 4 to 10 of Table 1),
summarized-T: // This table is similar to table 2.
04: While *(Length(T) >1 and i < Length(T))*
05: For *j=i+1 to Length(T)* ***do***
06: If *((T(i, dest-ip) = T(j, dest-ip))* ***And*** *(T(i, sig-id) =T(j, sig-id))* ***And*** *(T(i, alert-descrp-strct) = T(j, alert-descrp-strct)))* ***Then***
07: Add *the i^{th} record to table summarized-T*
08: Delete *the i^{th} and the j^{th} records from table T, set i=1*
09: Else
10: If *((T(i, dest-ip)=T(j, dest-ip))* ***And*** *(T(i, sig-id)=T(j, sig-id))* ***And*** *(T(i,alert-descrp-strct)* ***!=*** *T(j, alert-descrp-strct)))* ***Then***
11: ***Merge*** *the i^{th} and the j^{th} records of table T and add the resultant merged record to table summarized-T*
12: Delete *the i^{th} and the j^{th} records from table T, set i=1*
13: End If
14: End If
15: End For
16: *i=i+1*
17: End While
18: If *(T is not Empty)*
19: Add *table T to table summarized-T*
20: End IF
21: Return *(summarized-T)*
22: End

4.4 CIDS detection models

In the following, we describe the three alternative models to detect masquerade and host-based attacks namely: (A) Audit exchange model, (B) Audit exchange model with a neural network, and (C) Independent model.

A) Audit exchange model

In this model, nodes exchange their audit data among each others so that each one has a complete audit data for its current users. The detection phase depends on two parameters: (1) The alignment score computed in the CIDS detector component, (2) Alerts fired by the HIDS component. In this way, the detection overhead is balanced among nodes with no single point of failure. The detection efficiency is high because the user audit is concentrated in one place and the masquerader surviving is very short. As a counterpart, this model needs a fast periodic update of user audit data in all the nodes related to the user and this introduces some overhead in the cloud network. The processing steps are:

If user HIDS **or** CIDS instance fired **Then** //This denotes that an attack has been detected (Host-based or masquerade attack).

1. Alert all nodes that have VM instance(s)for that user to stop exchanging his audit data.
2. Send alerts to the scheduler node to do the following tasks:
   a. Stop the current tasks related to this user from all his VMs. if the alert is coming from HIDS detector then, stop only this malicious VM.
   b. Prepare a summarized report to the cloud administrator contains some information about the masqueraded user, the malicious VMs, and the detected attack.
   c. Apply the administrator action against this user by re-initializing his malicious VM(s) or by Blocking or suspending his account.

End if

B) Audit exchange model with a neural network.

This model integrates the audit exchange model (model *A*) with the neural network one. The resulting steps are: (1) For each user, prepare the training patterns that describe the user normal behaviors. (2) During the training phase, a history profile is built for each user with the following parameters: average number of failure login hits, average time between the typed commands, average time from login to logout or to the end of the session, source IP address(es) for each user login, VM name(s) used by the user, and time interval of logging to the cloud system (e.g., [7:8], [10:12], 15:18]). (3) At each login, the user profile is updated in any cloud node related to the user. In step 2, parameters are updated, and HSGAA computes the Sequence Alignment Scores (*SAS*) for the current login session of the user. (4) The neural network collects the previous input parameters and adapts the weights according to the back propagation algorithm [26]. (5) Perform a periodic update for each profile of the cloud nodes. (6) Go through the processing steps in the audit exchange model, for each user IDSs instance (i.e., CIDS or HIDS) firing. Besides the advantages of model *A,* this model offers these advantages. (1) The masquerader surviving is shorter than both models *A* and *C*, because of the neural network classification. (2) A higher hit rate with a fewer false positive and false negative alarms than both models *A* and *C*. This model has the same disadvantages of model *A*. Furthermore, its performance is worst than both *A* and *C*, because of the network and processing overheads to update the neural network parameters.

C) The independent model

Each cloud node evaluates its own user audits without exchanging data with other nodes. The detection phase depends upon the same two parameters of model *A* (i.e., *SAS* computed by HSGAA and alerts fired by HIDS). Login usage patterns for a user are evaluated using both CIDS and HIDS detectors inside a cloud node *CN* and by using the behavior-based and signature-based of *CN* only. If the HIDS detector of *CN* fires an alert, the algorithm will behave according to step 2 of model *A* for each user HIDS instance firing. If the CIDS detector of *CN* fires an alert, the algorithm checks the current login usage patterns against the audit data of the current user in the other nodes related to the user until a node accepts the current pattern, otherwise this user will be marked as a masquerader. Then, this model will behave according to step 2 of model *A* for each user CIDS instance firing. Algorithm 2 shows the steps of model *C*. The model advantages are: (1) It does not require a periodic update of user audit data in each node related to that user, (2) A very low overhead for the cloud network, as there is no data exchange, except if the score SAS_i is less than the previous define threshold θ_{SAS} then, the exchanged data is the test audit data (*test_d*) produced by the user during the login session, (3) A lower processing overhead for each cloud node than models *A* and *B*, because each node executes the HSGAA alignment of the test audit data (*test_d*) of the node which has the login session, only if the score SAS_i is less than the previous defined threshold θ_{SAS} . The disadvantages are: (1) A longer masquerader surviving than both models *A* and *B* because the analysis requires a long time to check the audit data (test_d) in all nodes. (2) A lower hit rate than model *B*, because the neural network classification model can not be applied due to the lack of the exchanged audit data. The hit and the false alarm rates of the model are the same of model *A*. Model *B* has a better false alarm rate.

Algorithm2: The analysis algorithm for model C

01: Begin
02: Inputs*: test audit data (test_d) produced by user (i) during the current login session behavior-base(behavr_d) stored for user (i) during the training phase inside the current login cloud node,* SAS_i *is the alignment score for user i computed by HSGAA,* θ_{SAS} *is the alignment threshold defined for user i, Not-Masq-flag = False.*
03: *Use HSGAA to compute* SAS_i *by aligning (test_d) against (behavr_d) in the same machine.*
04: If $SAS_i < \theta_{SAS}$ ***Then***
05: ***For*** *each cloud node(C_node)contains (behavr_d) of user i,* ***do****:*
06: *Use HSGAA to compute* SAS_i *for the* i^{th} *user in (C_node)*
07: ***If*** $SAS_i \geq \theta_{SAS}$ ***Then***
08: *Not-Masq-flag =* ***True***
09: *Exit the loop;*
10: ***End if***
11: ***End for***
12: End If
13: If *Not-Masq-flag = false* ***or*** *HIDS instance is fired* ***Then***
14: *Run step 2 of model A for each user i IDS instance firing.*
15: End If
16: End

5. Conclusion and future work:

Cloud computing risks and threats differ from traditional ones and current IDS technology is not suitable for cloud computing. This paper has proposed CIDS framework to define a proper defense strategy for cloud systems. CIDS is a scalable and elastic solution with P2P

architecture with no central coordinator that avoids a single point of failure. CIDS has two P2P deployment models hybrid P2P and pure P2P. To increase flexibility and portability, the middleware where the framework resides can be installed in distinct cloud and grid systems. To increase attacks coverage, CIDS integrates knowledge-based and behavior-based approaches and monitors each node to identify local events that could represent security violations. When an attack occurs, it alerts other nodes. CIDS exploits the distinct execution spaces of a VMM to separate the intrusion detection system from the system under monitoring so that the intrusion detector components become invisible and inaccessible to intruders. CIDS includes an audit system to discover those attacks that network-based and host-based systems cannot detect. It also parses and summarizes a high intensive number of alerts fired by NIDS component to prepare a readable report for the cloud administrator. CIDS provides three alternative detection models. The CIDS webpage [27] describes further details about CIDS.

For future work, we will apply our framework to a cloud system. We need to adapt a suitable dataset for cloud systems. For cloud systems, one user can have more than one instance of audit data in several machines, which is not currently available for the existing datasets. We also need to practically evaluate the parameters of HSGAA approach for each user separately. Another enhancement concerns the summarizer and parser algorithm. To reduce the corresponding overhead, the algorithm can be parallelized by dividing the collected alarms into a set of groups to be parsed and summarized in distinct nodes. The summarized alerts from each machine are collected and sent to the scheduler machine. Finally, we need also to apply the three proposed detection models, in order to choose the most suitable one according to both users and providers requirements.

6. References

[1] Karen Scarfone and Peter Mell, “Guide to Intrusion Detection and Prevention Systems (IDPS)”, National Institute of Standards and Technology(NIST), Special Publication 800-94, Feb. 2007.

[2] “Top Threats to Cloud Computing”, Cloud Security Alliance, http://www.cloudsecurityalliance.org/csaguide.pdf, V. 1.0 (2010)

[3] Foster, I.; Yong Zhao; Raicu, I.; Lu, S., "Cloud Computing and Grid Computing 360-Degree Compared", Grid Computing Environments Workshop, GCE '08, vol., no., pp.1-10, Nov. 2008

[4] J. Brodkin. “Gartner: Seven cloud-computing security risks”, http://www.networkworld.com/news/2008/070208-cloud.html.

[5] Jansen W., Karygiannis, T. 1999, “Mobile agents and security”. Special Publication 800-19, NIST.

[6] W Jansen, P Mell, T Karygiannis, Marks, "Applying Mobile Agents to Intrusion Detection and Response (1999)", National Institute of Standards and Technology Interim Report - 6416

[7] O. Choon and A. Samsudin, “Grid-based intrusion detection system,” in Proc. 9th Asia-Pacific Conference on Communications, vol. 3, pp. 1028-1032, September 21-24,2003.

[8] S. Kenny and B. Coghlan, “Towards a grid-wide intrusion detection system,” in Proc. European Grid. Conference (EGC2005), pp.275-284, Amsterdam, Netherland, February 2005

[9] F-Y. Leu, Fang-Yie Leu, Jia-Chun Lin, Ming-Chang Li, Chao-Tung Yang, and Po-Chi Shih, “Integrating Grid with Intrusion Detection”, Proc. Int’l Conf. Advanced Information Networking and Applications (AINA 05), vol. 1, March, 2005, pp. 304–309

[10] M. Tolba, M. Abdel-Wahab, and I. Taha, and A. Al-Shishtawy, “GIDA: Toward enabling grid intrusion detection systems”, in Proc. 5th IEEE/ACM Int. Symp. on Cluster Computing and the Grid (CCGrid2005), Cardiff, UK, May 9-12, 2005.

[11] Fang-Yie L., Jia-Chun L., Ming-Chang L., and Chao-Tung Y., “A Performance-Based Grid Intrusion Detection System,” in Proc. 29th Annual IEEE International Computer Software and Applications Conference (COMPSAC), pp.525-530, July, 2005

[12] Guofu Feng, Xiaoshe Dong, Weizhe Liu, Ying Chu, and Junyang Li, “GHIDS: Defending Computational Grids against Misusing of Shared Resource”, Proc. Asia-Pacific Conf. Services Computing (APSCC 06), December.2006, pp. 526–533.

[13] Schulter, A.; Navarro, F.; Koch, F.; Westphall, C.B., “Intrusion Detection for Computational Grids”, Proc. 2nd Int’l Conf. New Technologies, Mobility, and Security, IEEE Press, November, 2008, pp. 1–5.

[14] Hisham A. Kholidy, "A Study for Access Control flow Analysis with a proposed Job analyzer component based on Stack inspection methodology", 10th International Conference on Intelligent System Design and Applications (ISDA 2010), 29 Nov -2 Dec 2010.

[15] Amir Vahid Dastjerdi, Kamalrulnizam Abu Bakar, Sayed Gholam Hassan Tabatabaei, “Distributed Instrusion Detection in Clouds Using Mobile Agents”, Third International Conference on Advanced Engineering Computing and Application in Sciences, October 11-16, 2009 - Sliema, Malta

[16] Scerri, Paul and Vincent, Régis and Mailler, Roger, booktitle "Coordination of Large-Scale Multiagent Systems", Springer US, isbn: 978-0-387-27972-5, p. 231-254, 2006

[17] W. A. Jansen, “Intrusion detection with mobile agents”, Computer communication (15): page: 1392-1401, 2002.

[18] Roschke, S., Cheng, F., Meinel, “Intrusion Detection in the Cloud”, The 8th International Conference on Dependable, Autonomic and Secure Computing (DASC-09) China, Dec. 2009

[19] N. Kelem, R. Feiertag. “A Separation Model for Virtual Machine Monitors”, Research in Security and Privacy. Proceedings of the IEEE Computer Society Symposium, pages 78-86, 1991.

[20] http://www.ossec.net/main/

[21] Scott E. Coull, Joel W. Branch, Boleslaw K. Szymanski, Eric A. Breimer. 2008. “Sequence alignment for masquerade detection”. Journal of Computational Statistics & Data Analysis. 52, 8 (April 2008), 4116-4131. http://dx.doi.org/10.1016/j.csda.2008.01.022.

[22] D. Andersson, M. Fong, and A. Valdes, “Heterogeneous Sensor Correlation: A Case Study of Live Traffic Analysis,” Third Ann. IEEE Information Assurance Workshop, Jun. 2002.

[23] A. Valdes and K. Skinner, “An Approach to Sensor Correlation,”Proc. Recent Advances in Intrusion Detection, Oct. 2000.

[24] Fredrik Valeur, Giovanni Vigna and Richard A. Kemmerer, “A Comprehensive Approach to Intrusion”, IEEE Transactions on Dependable and Secure Computing, Jul. 2004

[25] http://www.snort.org/

[26] Stuart Russell and Peter Norvig. Artificial Intelligence A Modern Approach. p. 578.

[27] http://www.di.unipi.it/~hkholidy/projects/cids