Performance Analysis of Intrusion Detection Systems in the Cloud Computing

Mostapha Derfouf, Mohsine Eleuldj
Department of Computer Science
Mohammadia School of Engineers
MOHAMMED V UNIVERSITY IN RABAT
Morocco
mostaphaderfouf@research.emi.ac.ma, eleuldj@emi.ac.ma

Abstract— **Cloud computing is a paradigm that provides access to compute infrastructure on demand by allowing a user to use a big number of virtual machines (VMs) to solve a given computational problem. Before implementing new applications running on the cloud, it is often useful to estimate the different performance/cost of possible implementations. Do it physically on cloud providers will be intensive and expensive, this is why we use simulators. In this paper we will compare the different scenarios of collaborative intrusion detection systems that we have proposed already in a previous paper [1] using a tool called CloudAnalyst that was developed to simulate large-scale Cloud applications with the purpose of studying the behavior of such applications under various deployment configurations. The simulation is made taking into consideration several parameters such as the data processing time, the response time, user hourly average, the request servicing time , the total data transfer and virtual machines costs, the obtained results are analyzed and compared.**

Keywords- IDS, HIDS, NIDS, Cloud Computing,* performance analysis, *CloudAnalyst, CloudSim

I. Introduction

Cloud computing is a new data hosting technology that stands today as a satisfactory answer to the problem of data storage and computing. It can provide processing and accommodation of digital information via a fully outsourced infrastructure allowing users to benefit from many online services without having to worry about the technical aspects of their uses. Cloud computing appears as a tremendous opportunity for companies but logically raises the question of the security of data when they are hosted by a third party.

The increasingly frequent use of Cloud Computing created new security risks. Thereby increasing the interest of hackers to find new vulnerabilities and exposing users to see their data compromised. We have already dealt with the problem of data security in Cloud computing and proposed two scenario of collaborative intrusion detection systems for the cloud , in this paper we will provide an evaluation of each scenario taking into account different parameters like the data processing time , the response time , load balancing ,number of users and data centers and the number of resources in each data center in order to provide the cloud providers and users with information on costs and performance of execution of the different architecture of intrusion detection to be able to choose.

II. The different architectures of intrusion detection systems in the cloud computing

In the "Advanced IDS Management Architecture" [2] the authors proposed an IDS which uses an Event Gatherer combined with the Virtual Machine Monitor (VMM). This IDS is composed of many sensors and a central management unit. The Event Gatherer plugin plays the role of Handler, sender, and receiver in order to provide an integration of different sensors. This architecture uses the IDMEF (Intrusion Detection Message Exchange Format). An interface is designed to expose the result reports for users.

The multilevel IDS concept is proposed by M.Kuzhalisai and G. Gayathri [3] which deals with effective use of system of resources. The proposed system binds user in different security groups based on degree of anomaly called anomaly level. It consists of AAA module which is responsible for authentication, authorization and accounting. When user tries to access the cloud the AAA checks the authentication of the user and based on it, it gets the recently updated anomaly level. Security is divided into three levels: high, medium and low..

Irfan Gul and M. Hussain [4] have proposed a multi-threaded NIDS designed to work in distributed cloud environment. This multi-threaded NIDS contains three modules: capture and queuing module, analysis/processing module and reporting module. The capture module is responsible of reading the network packets and sending them to the shared queue for analysis.

In [5] the authors proposed a framework that integrates a network intrusion detection system (NIDS) in the Cloud. The proposed NIDS module consists of Snort and signature apriori algorithm that is capable of generating new rules from captured packets. The new rules are added to the Snort configuration file.

In [7] the authors proposed an improved Hybrid IDS. The Improved hybrid IDS is combination of anomaly based detection and honey pot technology with KFSensor and Flowmatrix.The Honey pot is used to attract more and more attackers, the detection obtained can be used to create new signatures and update the database. Finally anomaly can be used to detect unknown attack in the whole network.

The Cloud Detection and Prevention System (CIDPS) architecture [8] is illustrated and presented as a workflow scenario. Sensor inputs or alerts generated while monitoring network, host, platform and applications together with the latest CIDPS challenges and enterprise CIDPS policies and their updates, drive through the CIDPS Trust Management system to be analyzed. Inference Engine (IE) is the logical and main part of IDE.

In CIDS architecture [9] 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. This framework contains CIDS components, cloud system components and NIDS components. Table 1 is a comparative table that presents the different architectures of intrusion detection systems.

IDS / Features	Advanced IDS Management Architecture	Cloud Intrusion Detection System	Cloud Detection and Prevention System	Improved Hybrid IDS
Type	Collaborative	Collaborative	Intelligent	Intelligent
The ability to detect unknown attacks	No	No	Could be	Could be
The ability to analyze the content of encrypted streams	Yes	Yes	No	Yes
Encrypting exchanged alerts	No	No	No	No
Diffusion of detected attacks	No	Using alert system message	No	No

Table 1. Synthesis of IDS architectures in the Cloud

III. SMART INTRUSION DETECTION MODEL FOR THE CLOUD COMPUTING

This section describes two architectural scenarios of intrusion detection systems proposed in the previous paper and are : the distributed intrusion detection architecture and centralized intrusion detection architecture.

A. Distributed intrusion detection architecture

The first solution proposed is to set up a distributed intrusion detection architecture in which each HIDS communicate by exchanging IDMEF [10] alerts describing the detected attacks , each HIDS has its own database and is equipped with mining and machine learning module to detect unknown attacks. Using this approach if an intrusion is detected by a SHIDS, it will be communicated to all other SHIDS to update their database as shown in figure 1.

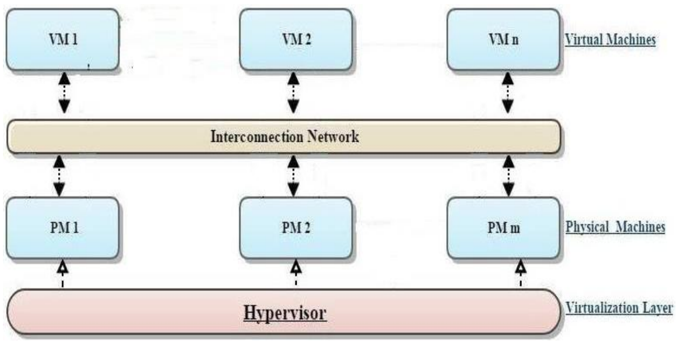

Fig 1. Distributed intrusion detection architecture

B. Centralized intrusion detection architecture

The second scenario is based on a centralized IDS architecture (figure 2) that is based on the principle of collaboration between many SHIDS deployed on the different virtual machines (assuming that we have n virtual machines VM1, VM2 …. VMn and m Physical machines PM1,PM2…. PMm with m # n) in the cloud to detect and protect against attacks targeting applications running on these virtual machines, this approach provides many benefits in terms of portability and costs. The concept of this model is that the different SHIDSs are placed in each VM and cooperate with each other by exchanging alerts about detected intrusions. In our model there is a central agent that is responsible for the reception of notifications from the other SHIDS as well as writing and reading from a central database in order to synchronize the local database of each SHIDS.

In this approach the different SHIDS are synchronized and each detected attack is communicated to other neighbors by the central agent. The model is improved by using the data mining and machine learning techniques to detect unknown attacks.

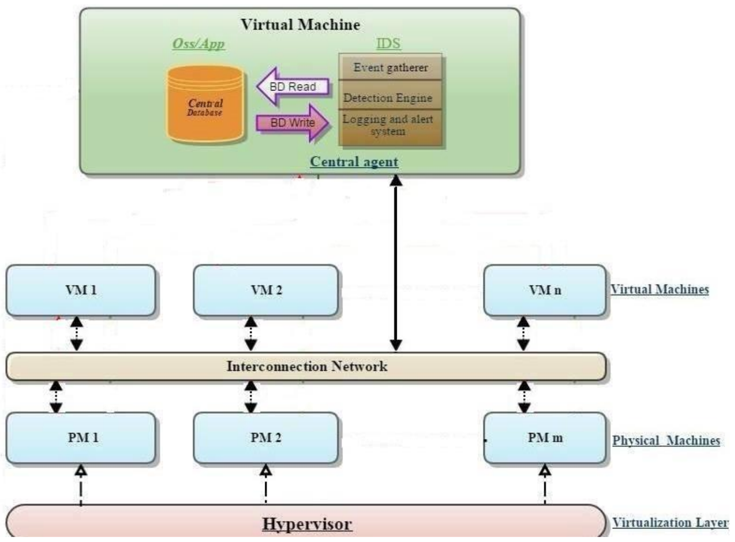

Fig 2. Centralized intrusion detection architecture

IV. IMPLEMENTATION OF THE CENTRALE VIRTUAL MACHINE

In this section we will see the different possible implementations of the centralized intrusion detection system architecture mentioned earlier, taking into account the various parameters such as the data processing time, the response time , load balancing ,number of users and data centers and the number of resources in each data center.

To give an exhaustive list of all implementation, we use the abbreviation "MP" for machine learning processing and “KB” for knowledge base, the following tree shows the different implementations:

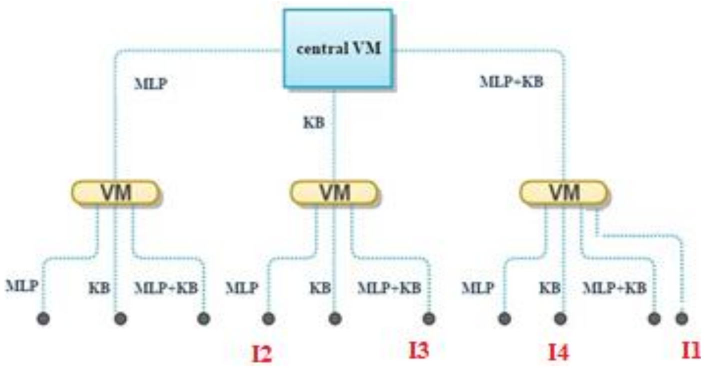

Fig 3. Implementations of the central virtual

Some implementations can be eliminated others can be kept for simulation:

The case of central virtual machine with machine learning and virtual machines with machine learning because it is evident that there is a redundancy since we use the same algorithm in the central VM and in the other VM in addition There will be an overload in treatments since the same processing will be executed in the central VM and in the other VM.

The case of a central VM with KB and learning processing and VM without knowledge base and without machine learning can be kept for simulation. We call it implementation 1 (I1).

The case of a central VM with knowledge base and without machine learning processing and VMs with machine learning processing can be kept for simulation. We call it implementation 2 (I2).

The case of a central VM with KB and without learning processing and VM with knowledge base and machine learning can be kept for simulation. In this case the database is duplicated in the central VM and other VMs, this can be very helpful in case of damage of the knowledge database. We call it implementation 3 (I3).

The case of a central VM with knowledge base and machine learning processing and VMs with knowledge base can be kept for simulation. We call it implementation 4 (I4).

The case of a central VM with machine learning processing and virtual machines with knowledge database and machine learning processing can be eliminated because we use the same algorithm in the central VM and in the other VM in addition there will be an overload in treatments.

The case of a central VM with knowledge database and virtual machine with knowledge database is eliminated because it does not represent a smart IDS since there is no machine learning processing

The case of a central VM with knowledge database and machine learning processing and virtual machines with machine learning processing is eliminated because there is a redundancy in processing.

The distributed architecture is called implementation 5 (I5).

A. *Central virtual machine integrating a central knowledge database*

The central agent is a virtual machine responsible for synchronization and updating of all other SHIDS. In this case the central virtual machine is equipped with a knowledge base and no machine processing so that the other virtual machines integrate machine learning and when a new signature is detected it is automatically sent the knowledge database in the central virtual machine to be stored. If one of the virtual machine receives an attack that is not recognized it asks the knowledge database.

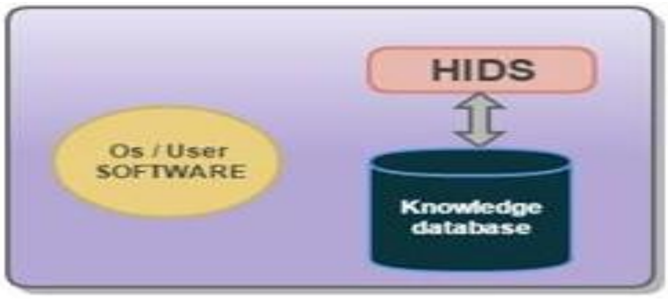

Fig 4. Central virtual machine integrating a knowledge database

B. *Central virtual machine with machine learning engine and integrating knowledge database*

In this case the central virtual machine is equipped with a knowledge base and a machine processing so that the other virtual machines don’t integrate any machine learning processing because this will weigh down performance. When a new signature doesn’t have a matching in the SHIDS of the virtual machine so the signature is automatically sent in an IDMEF format to the central agent , the machine learning is applied on this signature and if an attack is detected it will be stored in the central knowledge database.

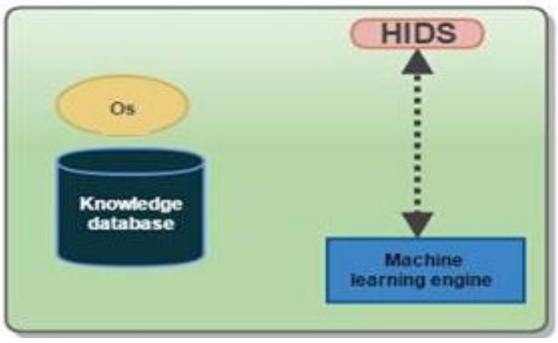

Fig 5. Central virtual machine integrating a knowledge database and machine learning

C. Central virtual machine with machine learning engine without knowledge database

In this case the central virtual machine contains a machine learning engine to detect new attacks and no central database. So when the SHIDS of each VM receives an unknown signature it will be sent via logging and alert system to the central agent to perform machine learning processing and if an attack is detected it will be stored on the knowledge database of each VM.

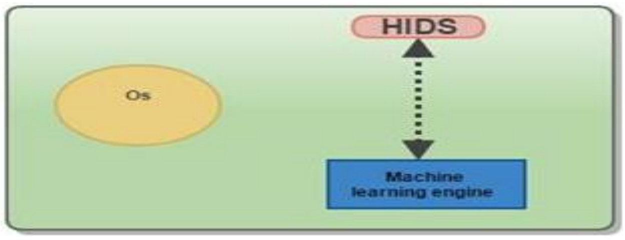

Fig 6. Central virtual machine without a knowledge database and with machine learning

V. IMPLEMENTATION OF THE VIRTUAL MACHINE

A. Virtual machine with machine learning engine and knowledge database

This is a possible scenario of virtual machine in the centralized architecture. In this case the virtual machine is equipped with a SHIDS with a machine learning engine and without knowledge database (figure 7).

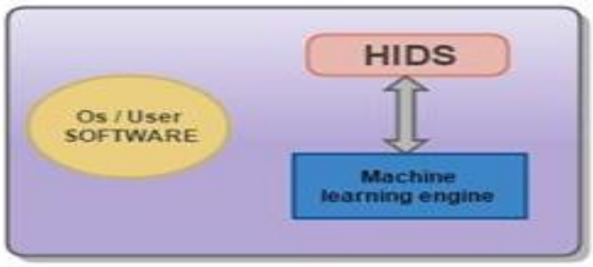

Fig 7. Virtual machine with machine learning and no knowledge database

B. Virtual machine without machine learning engine and integrating a knowledge database

This is the second possible scenario of virtual machine in the centralized architecture. In this case the virtual machine is equipped only with a SHIDS without any machine learning processing and with knowledge database (figure 8).

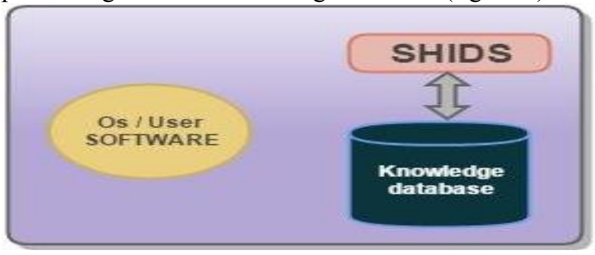

Fig 8. Virtual machine without machine learning and integrating a knowledge database

C. Virtual machine without machine learning engine and and no knowledge database

This is the third possible scenario of virtual machine in the centralized architecture. In this case the virtual machine is equipped only with a SHIDS without any machine learning processing and without a knowledge database (figure 9).

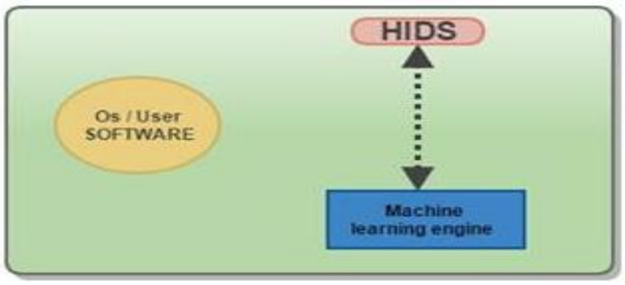

Fig 9. Virtual machine without machine learning and no knowledge database

VI. SIMULATION TOOLS

In this section we focus on the different tools used in the simulation of the different implementation scenarios. We detail the two famous tools CloudAnalyst and CloudSim.

A. CloudAnalyst

To allow control and repeatability of experiments, some simulators such as CloudSim and CloudAnalyst are used. Simulation experiments apply models of both applications and infrastructures [11].

CloudAnalyst is a tool developed in java to simulate large-scale Cloud application, it allows to perform simulations based on different parameters such as the geographic location of users, the number of and data centers and users, the location of data centers and computing resources in each data center and gives information about requests processing time, Datacenter processing time, Total data transfer and others metrics. The results are presented in tables and charts which helps in identifying the important patterns of the output parameters and helps in comparisons between related parameters [12].

We used the CloudAnalyst simulator to model the different scenarios of intrusion detection architecture in the Cloud, analyze the results and choose the best one. The architecture of CloudAnast is presented in the figure 10.

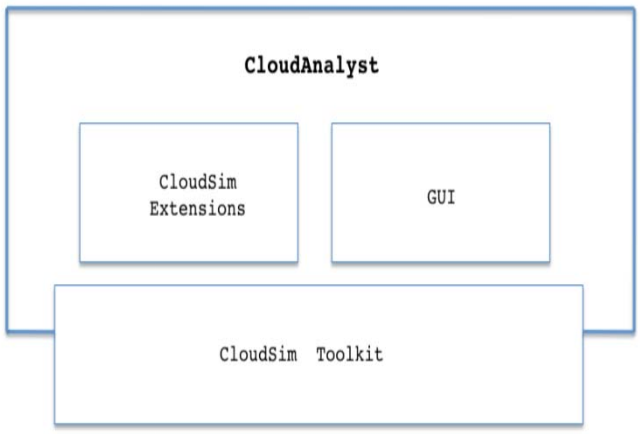

Fig 10. CloudAnalyst Architecture [12]

B. *CloudSim*

CloudSim is a framework used to model and simulate the environment of Cloud computing and its services, was realized in Java. CloudSim supports modeling and simulation of the cloud-based Datacenter environment, such as management interfaces dedicated to VMs, memory, storage, and bandwidth.

VII. SIMULATION OF THE DIFFERENT SCENARIOS

A. *Configuration of the simulation*

We used CloudAnalyst to perform the simulation and define the various simulation parameters. The main Configurable parameters are:
- User bases: defines the users of the cloud applications, their geographic distribution, the number of users, the frequency of usage and the pattern of usage such as peak hours.
- Data Centers: define the data centers used in the simulation. Including hardware and software features.
- Internet Characteristics: Review and adjust the network latency and bandwidth matrices.

For the experiment we create one user base corresponding to a region of the world .

B. *Simulating the distributed intrusion detection architecture*

In this section we simulate the distributed intrusion detection architecture. The distributed intrusion detection architecture is similar to a network in which each node relays data for the network. All nodes cooperate in the distribution of data in the network. Each terminal is connected to all the others. The disadvantage is that the number of connections required becomes very high when the number of terminals increases. the number of requests sent by one virtual machine in one hour is 24*3600=86400 requests.

We suppose that each VM has a machine learning engine and a knowledge database and we use the algorithm EFRID (Evolving Fuzzy Rules for Intrusion Detection) proposed by Gomez and Dasgupta [16] which achieves an overall true positive rate of 95% so the number of requests (packets containing intrusions) will be (86400*95)/100 =82080.

In this scenario we have created three datacenters with one virtual machine (20 Go of RAM) in each datacenter to receive alerts from the users , in this case all alerts from the different users in the world are processed by the different virtual machines in the three datacenters. The data center 1 is created in region 0; the data center 2 is created in region 1 and the data center 3 in region 2. We used Linux as the operating system in these data centers and Xen as hypervisor as shown in figure 6.The memory capacity in DC1, the storage capacity is 100GB.

C. *Simulating the centralized intrusion detection architecture*

In this section we simulate the centralized architecture scenario in which the central virtual machine performs machine learning processing and contains the knowledge database used to store detected attacks using machine learning. The other virtual machines hosting cloud applications will not perform machine learning processing in order to avoid further treatment and do not contain any knowledge database.

To perform the simulation of the centralized architecture , we have created a single datacenter with one virtual machine (20 Go of RAM) that is considered as central IDS with knowledge base and machine learning processing, it receives all alerts from the other VMS , in this case all alerts are processed by this single data center containing the central IDS. The data center is created in region 0, we used Linux as the operating system in this data center and Xen as hypervisor. The memory capacity in DC1 is 20 Go and the storage capacity is 100GB (figure 7)

In the case of the central architecture we suppose that we have 1 virtual machines in the physical server server1, the machine is hosting a cloud service and containing a smart host based intrusion detection system to detect attacks targeting that VM. The VM in this experiment has a size of 100MB, 20 GB of RAM memory and 10MB of available bandwidth.

In this demonstration we consider users belonging to different regions as machines on which the HIDS are implemented because what interests us is the exchanged traffic and response time and we set the data size per request to 3632 octets which corresponds to the size of the IDMEF alert sent to the central IDS.

For simplicity each user base is contained within a single time zone and let us suppose that most users use the application for 12 hours. We assume that 5% of the registered users will be online during the peak time simultaneously and only one tenth of that number during the off-peak hours.

We also assume that each user makes a new request every 5 minutes when online. We suppose that we have 24 requests per second (in case of HTTP) [15] so in 1 hour we have 24*3600 = 86400 requests per hour and for a better estimation it is assumed that 100% of received requests contain intrusions.

We suppose that we use the algorithm EFRID (Evolving Fuzzy Rules for Intrusion Detection) proposed by Gomez and Dasgupta [16] which achieves an overall true positive rate of 95%. We suppose that we have 24 requests per second (in case of HTTP) so in 1 hour we have 24*3600 = 86400 [15] and for a better estimation it is assumed that 100% of received requests contain intrusions , using this algorithm the number of possible detected attack will be 24*3600 = (86400*95)/100 =82080. There are 82486 that will be sent the central VM to update the knowledge base.

VIII. Interpretation and Synthesis

In this section, we will provide a comparative study of the five measurement criteria presented in the table above. Our goal is to determine the performance of each intrusion detection scenario. The table 2 shows the comparison of the results of five measurement criteria (overall response time, datacenter processing time, user base hourly average, the request servicing time, total data transfer and VMs costs) after simulating the centralized and distributed architectures. To have a clear table we use the following abbreviations for the five measurement criteria :

C1= Overall response time in millisecond (ms) which is the total amount of time it takes to respond to a request, C2= Datacenter processing time in millisecond (ms), C3 = user base hourly average, C4 = the request servicing time in millisecond (ms), C5= Total data transfer and VMs costs which calculates the grand total of data center .

Criteria/ Impleme -ntations	C1 (ms)	C2 (ms)	C3 (hours)	C4 (ms)	C5 ($)
I1	298.10	5.06	< 2	5.062	VM cost : 0.1 transfer cost :1029.92
I2	600.97	17.13	< 2	28.02	VM cost : 0.6 transfer cost : 639.45
I3	700.10	12.05	< 2	35.062	VM cost : 0.8 transfer cost : 300.92
I4	300.97	6.13	< 2	12.03	VM cost : 0.3 transfer cost :1340.45
I5	300.44	17.05	> 5	51.7	VM cost : 1 transfer cost :16195.42

Table 2. Comparative table

A. *Overall response time:*

In the first measurement criteria, as we can see, the overall response time value in the first centralized architecture is less than the one in the second centralized architecture. To clarify, in the centralized architecture, the SHIDSs placed in each VM has a machine learning engine and corporate and exchange alerts for each detected intrusion with each other and with the central agent containing the central knowledge base that is responsible for many tasks, such as the reception of notifications from other SHIDS as well as writing and reading from the central database since the other VMs don't have a knowledge database so they are forced to question the central base. These treatments of bandwidth resources, which will increase the datacenter response time. That's what makes the difference between the centralized architecture and the distributed architecture; where all of these treatments are implanted into each VM.

B. *Datacenter processing time:*

In the first centralized architecture, the alerts exchanging and machine learning treatments are centralized in the central agent and not in each VM (as in the second centralized and distributed architecture). This will allow the data center to save resources used to accomplish these treatments in terms of memory and CPU. This is why the datacenter processing time in the first and fourth centralized architecture is lower than the one in the in the second and third centralized scenario the distributed scenario.

C. The request servicing time:

The request servicing time of the first centralized scenario is 5.06 which is lower than the second scenario of the centralized architecture which is 17.13 ms this is because each VM performs the machine learning treatment on only received packets and this is consuming in term of memory and CPU. In the 3th scenario (35.062 ms) the request servicing time is higher than in the 4th scenario because the machine learning treatment is done in each VM hosting cloud services however in the 4th scenario all treatment are done only one time and one central VM the other VMs will only send packets to this central VM for processing.

D. Total data transfer and VMs costs:

The data transfer cost of the first and fourth centralized scenario is respectively 1029.92 and 1340 ms which is higher than the one in the second and third centralized scenario. Because in the second and centralized the exchange is done twice: during the detection of the vulnerabilities and sending them to the central node and also during the interrogation of the central database for update.

Regarding the transfer and VMs costs results we find it normal in the two centralized scenarios since all the communications are donne with only one host that is the central VM, however in the distributed architecture all VM communicates with each other is why the cost is very high.

To be more precise and to give a simulation that is very close to the reality we carry out the simulation for different number of virtual machines ranging from 25 to 100 and we try to give a synthesis (table 3). Since the processing time is the most important criteria for the simulation, the simulations are performed and the tracing time is compared for each scenario.

We use the term scenario which refers to an implementation with a varied number of virtual machines (scenario 1 for the implementation 1, scenario 2 for the implementation 2…) , the time is in milliseconds.

Number of VMs/ Scenarios	**100**	**75**	**50**	**25**	**10**	**5**
Scenario 1	298.10	260.25	220.08	140.78	58.22	30.17
Scenario 2	600.97	420.96	380.00	215.78	140	94.56
Scenario 3	700.10	480.23	396.10	232.23	164.12	113.05
Scenario 4	300.97	286.15	227.25	149.02	75.80	40.17
Scenario 5	300.44	284.15	270.10	200.24	140.	90.52

Table 3. Comparative table of the overall response time (C1) for the different scenarios

We decided to translate the table 2 obtained into a graphic chart to better see the most suitable scenario and the least expensive in terms of processing time as shown in figure 11. From the charter presented in figure 11 we can deduce that the scenario 1 is the best scenario in terms of time processing.

IX. CONCLUSION

We have presented the different architecture of intrusion detection system in the Cloud Computing , proposed the different architectures and implementation of the smart intrusion detection systems we have also presented the tools used to perform the simulation then we have described all possible architectural scenarios of intrusion detection systems. Then, we have displayed a simulation and testing of the different scenarios. Ultimately, we have provided a comparative study and an interpretation of the simulation results of the centralized and distributed intrusion detection architectures. The scenario 1 is the best one, it refers to the central VM with knowledge database and machine learning processing and VM without knowledge base and without machine learning.

REFERENCES

[1] M.Derfouf, M.Eleuldj, S.Enniari and O.Diouri, “Smart intrusion detection model for the Cloud Computing” Middle East and North Africa Conference on Technology and Security), Saidia, Morroco, 3-5 October 2016.

[2] S. Roschke, F. Cheng, and C. Meinel, “An advanced ids management architecture,” Journal of Information Assurance and Security, vol. 5, pp. 246–255, 2010

[3] M. Kuzhalisai and G. Gayathri, "Enhanced Security in Cloud with Multi-Level Intrusion Detection System", IJCCT, Vol. 3, Issue 3, 2012.

[4] I. Gul and M. Hussain, “Distributed cloud intrusion detection model,” International Journal of Advanced Science and Technology, vol. 34, pp. 71–82, 2011.

[5] C. N. Modi, D. R. Patel, A. Patel, and M. Rajarajan, “Integrating signature apriori based network intrusion detection system (nids) in cloud computing,” Procedia Technology, vol. 6, pp. 905–912, 2012.

[6] https://www.sans.org/reading-room/whitepapers/detection/idmef-lingua-franca-security-incident-management-1080

[7] Ajeet Kumar Gautam, Dr. Vidushi Sharma, Shiv Prakash and Manak Gupta, "Improved Hybrid Intrusion Detection System (HIDS): Mitigating False Alarm in Cloud Computing", JCT, 2012.

[8] A. Patel, M. Taghavi, K. Bakhtiyari, and J. C. J´unior, “Taxonomy and proposed architecture of intrusion detection and prevention systems for cloud computing,” in Cyberspace Safety and Security. Springer, 2012, pp. 441–458.

[9] H. A. Kholidy and F. Baiardi, “Cids: A framework for intrusion detection in cloud systems,” in Information Technology: New Generations (ITNG), 2012 Ninth International Conference on. IEEE, 2012, pp. 379–385.

[10] J. Gustedt, E. Jeannot, and Martin Quinson, “Experimental methodologies for large-scale systems: a survey,” Parallel Processing Letters, vol. 19, Sep. 2009, pp. 399-418.

[11] Bhathiya Wickremasinghe and Rajkumar Buyya. Cloud-analyst: A cloudsim-based tool for modelling and analysis of large scale cloud computing environments. MEDC Project Report, 2009.

[12] https://www.researchgate.net/figure/281764373_fig3_Figure-3-Cloud-Analyst-Architecture

[13] http://students.cec.wustl.edu/~azinoujani/ accessed on Feb, 2015

[14] Debar, H., Curry, D., Feinstein, B.: The Intrusion Detection Message Exchange Format, Internet Draft. Technical Report, IETF Intrusion Detection Exchange Format Working Group (July 2004).

[15] Stacy Joines, Ruth Willenborg, Ken Hygh :Performance Analysis for Java Web Sites ,chapter 6.

[16] Jonatan Gomez and Dipankar Dasgupta. Evolving fuzzy classifiers for intrusion detection. In Proceedings of the 2002 IEEE Workshop on Information Assurance, West Point, NY, USA, 2002.

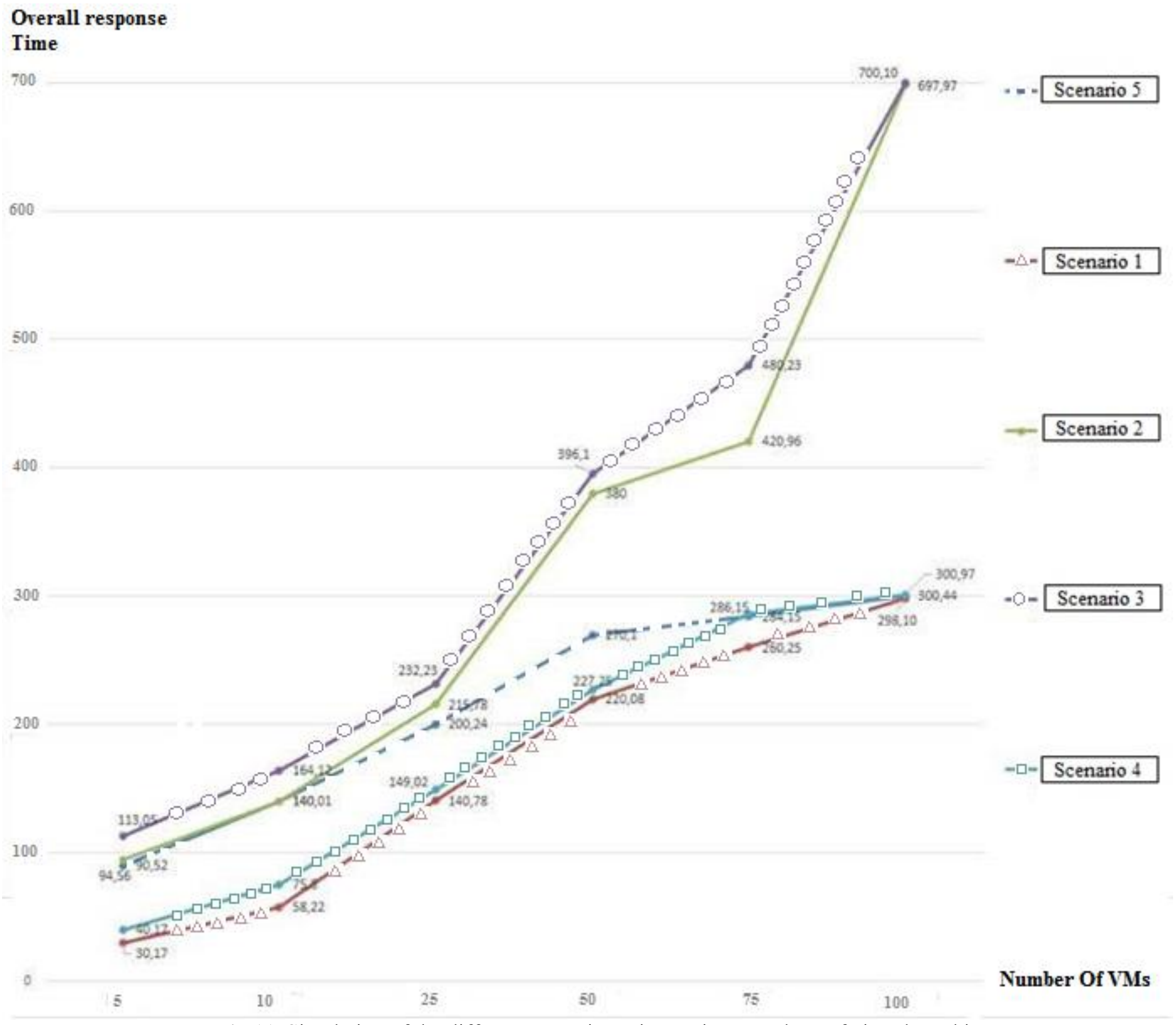

Fig 11. Simulation of the different scenarios using various numbers of virtual machines