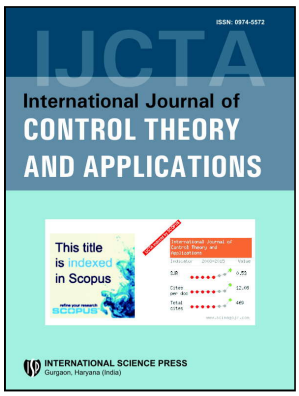

International Journal of Control Theory and Applications

ISSN : 0974-5572

Volume 10 • Number 14 • 2017

Malware Anti Defense Infrastructure Cloud Network

Surabhi Soni[1], Gaurav Goel[1] and Manish Mahajan[1]

[1] *Department of Computer Science Engineering CGC Landran, Mohali, Punjab, India, Emails: surabhisoni69@gmail.com, cgccoe.cse.gaurav@gmail.com,* hod.coecse@*cgc.edu.in*

Abstract: Nowadays the major challenge of cloud computing is security. The systems which are based on cloud become a target for malware due to rising of cloud computing technology. In an infrastructure-as-a-service cloud, due to the presence of malicious software in the virtual machine not only the customers but cloud infrastructure as well as other co-hosted customers are affected directly. As the quantity and complexity of malware threat are continuously transforming and increasing, it becomes too difficult for the malware authors to detect and analysis the malware threat. This paper presents an architecture that contains decentralized analysis framework which provides a guarantee of protection of cloud environment from the attackers. This architecture supports both lightweight intrusion monitoring and heavyweight in-depth analysis and helps in detection of attacks, collection of evidence, analysis of virtual machine. All on-demand decisions taken by decision engine have control over the suspicious events considering two restrictions therefore quality of service and cost-efficiency.

Keywords: Malware, Virtual machine, Cloud security, Cloud computing, lightweight intrusion monitoring, heavyweight intrusion monitoring.

I. INTRODUCTION

In the domain of computer science, Cloud computing is an important template in terms of computing. The big challenge of cloud computing is security. So there is need of some security measures to protect the data from unauthorized users that is stored on the cloud. The cloud providers offered pliability and scalability which produces more and more services that farm out to the cloud. In this paper, some security challenges that are produced in Infrastructure-as-a-service cloud environment will be discussed. Now cloud becomes high homogeneous, so it is the first target for the attackers; as the attackers know the places from where they can find the profitable information. As a consequence, the major engineering challenge is viewed as the cloud security [13].

Nowadays malware threat scenario is quickly transforming. Malware is becoming more cosmopolitan. For governments, enterprises and end-users, the detection of malware threat is the topmost choice for them. They are trying to get more knowledge about how to detect and avoid the attacks [8,17,21].There are two detection approaches such as signature based and static analysis based techniques can be easily equivocate by

techniques which can be seen in the wild such as encryption, obfuscation, polymorphism. To analysis and detection of these attacks some dynamic malware tools are become more popular here [1,4,20]. The detection of attacks accurately, malware analysis and collection of evidence depends upon the selected information sources and different monitoring mechanisms. Therefore, gathering information in detailed occur higher runtime cost. An intractable tradeoff is to be produced between performance effects on analysis cost, quality of the collected data and a production system. The collected data inside a virtual machine is easy target for the attackers to change it.

There are many existing policies that defined solutions of the problem. But there is one problem which has no solution therefore malware analysis, collection of evidence and detection of intrusion togetherness cannot define in joint architecture. In this paper, a system MADICN is defined that monitors the activities of virtual machines using lightweight detection technique as well as heavyweight detection technique.

The residue of the paper contains some sections which are as follows. Section II summarized threat model. Section III organized the CloudIDEA architecture. Section IV explained the system overview. Section V summarized the Honey-pot MapReduce model .Section VI described the granularity levels of honey pot .Section VII summarized the discussion on how to protect the data. Section VIII described the related work.

2. THREAT MODEL

In an infrastructure-as-a–service cloud threat model, there are three fundamental attack goals in a cloud environment: the execution of VMs in the cloud, the cloud management system, the hypervisor and the node management.

As much as like other software, the cloud management system is also not much secure therefore it contains many defects. In this research, the main goal is not being avoided or detection of some attacks. The main function of subsystem such as the hypervisor/node management is to handle the virtual machines and the control of this subsystem is taken by the cloud management system. To be suppose that the external attackers are not able to access the management interface, as it is commonly hidden inside a VLAN such as dedicated management

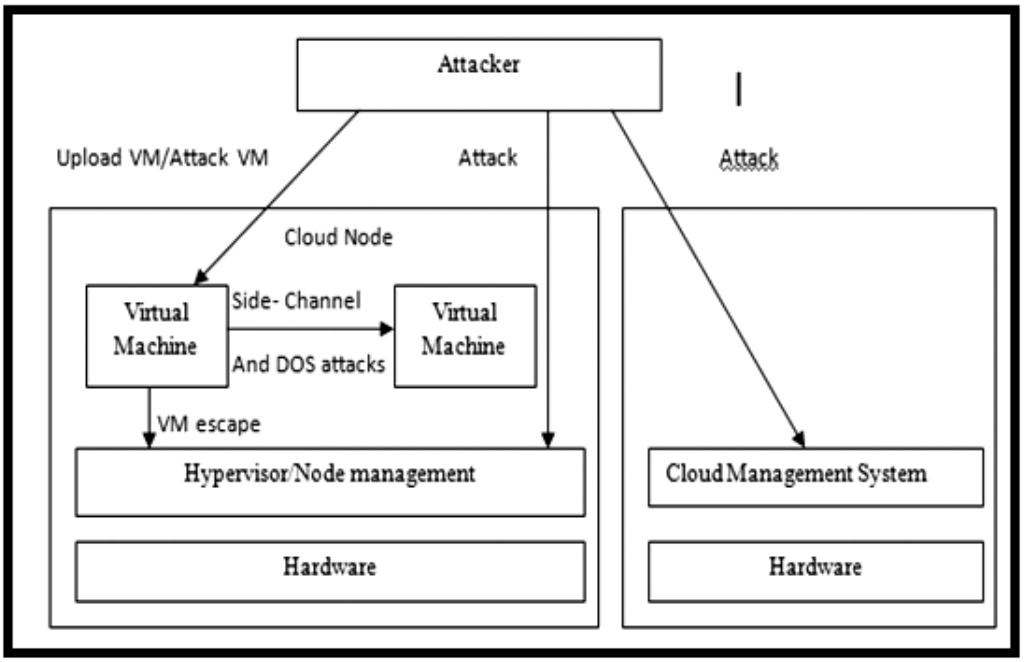

Figure 1: Iaas Cloud Threat model[7]

and against this subsystem it does not consider direct external hits. If the virtual machine is controlled by the spike, then the potential attack vector is the hypercall interface of the virtual machine [15].

Both the source of attack and the target is the customer virtual machine. As an objective, the striker can directly be exploited susceptibility of running software inside the virtual machine. These hits are the main cause of the effected cloud customers, and the main aim is to protect the cloud customers from these such attacks. Furthermore, any opponent can upload the malware virtual machine in a public cloud. There are two types of attacks which can be done by attackers on customer virtual machine i.e., Cross-virtual machine side channel attacks and denial of service attacks. The denial of service attacks can be done by loading the physical resources of a cloud node in excess or through the virtual network [19]. The second attack cross-virtual machine side channel attacks can be done by observing the behavior of virtual or physical hardware modules therefore L2 cache [22], and deduction of the information that contains in the other virtual machine. Moreover, by using different techniques such as memory brute force attacks or privilege escalation, effected virtual machines are tried to elude the hypervisor. After eluded the hypervisor, the attacker can easily gain the control of other virtual machines, the hypervisor or the cloud node completely [18].

3. PROPOSED WORK

3.1. Proposed Framework

The proposed framework is shown in figure 2. In this architecture, lightweight monitoring and heavyweight monitoring is done and controlled by the cloud management component. When any infected virtual machine is analyzed by different mechanisms then this suspected virtual machine is moved to dedicated analysis environment. Each virtual machine is observed by a localised analysis framework which is component of each class node.

For introspection and tracing of virtual machines therefore hypercall tracing and network traffic monitoring is done by using plug-ins. In a production environment, to minimize the overhead and notice the current threat

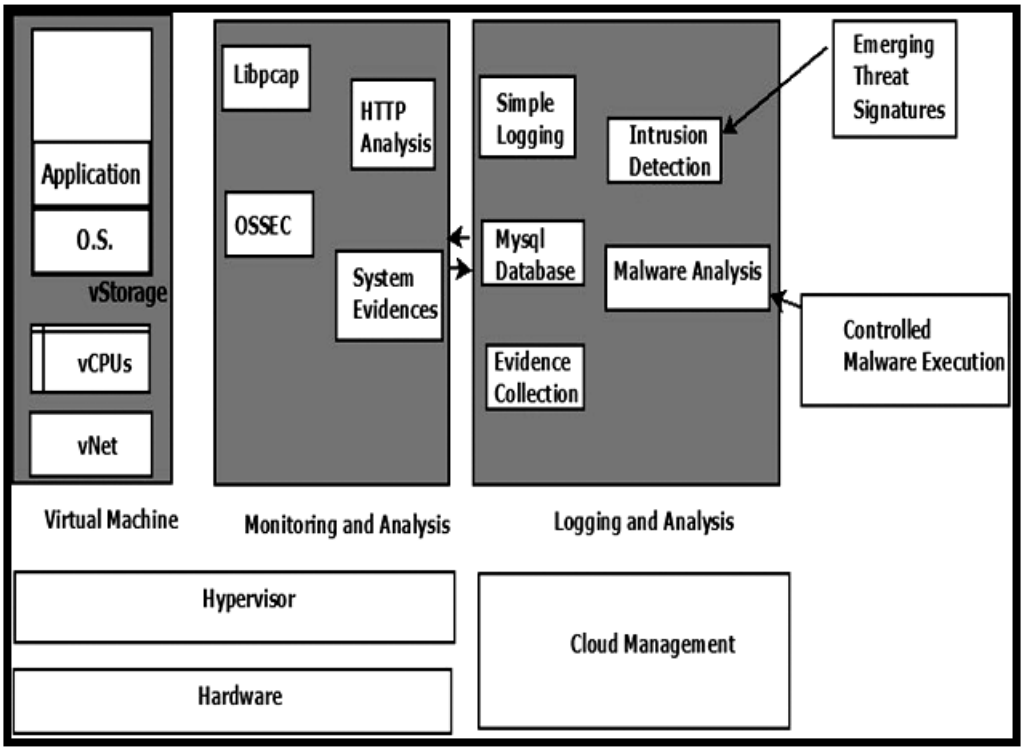

Figure 2: Proposed Framework

level some monitoring mechanisms has been tailored, on demand these plug-ins can be deactivated and activated at runtime.CloudIDEA management is the component that one helps to processing the traces of the system. It adds other components in it that are responsible for collection of evidence, detection of attacks, mitigation that helps to compare and learn all the patterns for all virtual machines in a cloud data centers.

3.2. Flowchart

The flowchart of work is shown in figure 3. The first step is to set up the cloud networks or virtual machine to host public/private cloud. The second step is to set up the cloud using any open source tool such as Linux and then providing the network to access the virtual machine. The next step is to download and compilation of libpcap. The next step is to monitoring and capturing of traffic. After monitoring of traffic, system will be traced. After tracing the system, if packet found any malicious software then alerts will be generated otherwise the implementation of signature based as well as behavior based analysis engines is done.

4. EXPERIMENTAL RESULTS

Tentative outcome evaluates the performance by concealing the information in a cover image using SLINUX.

To assess the execution of the proposed model, different performance parameters are used. In this paper, the results of extract and write parameters are shown in which the comparison of proposed and already present system has been done.

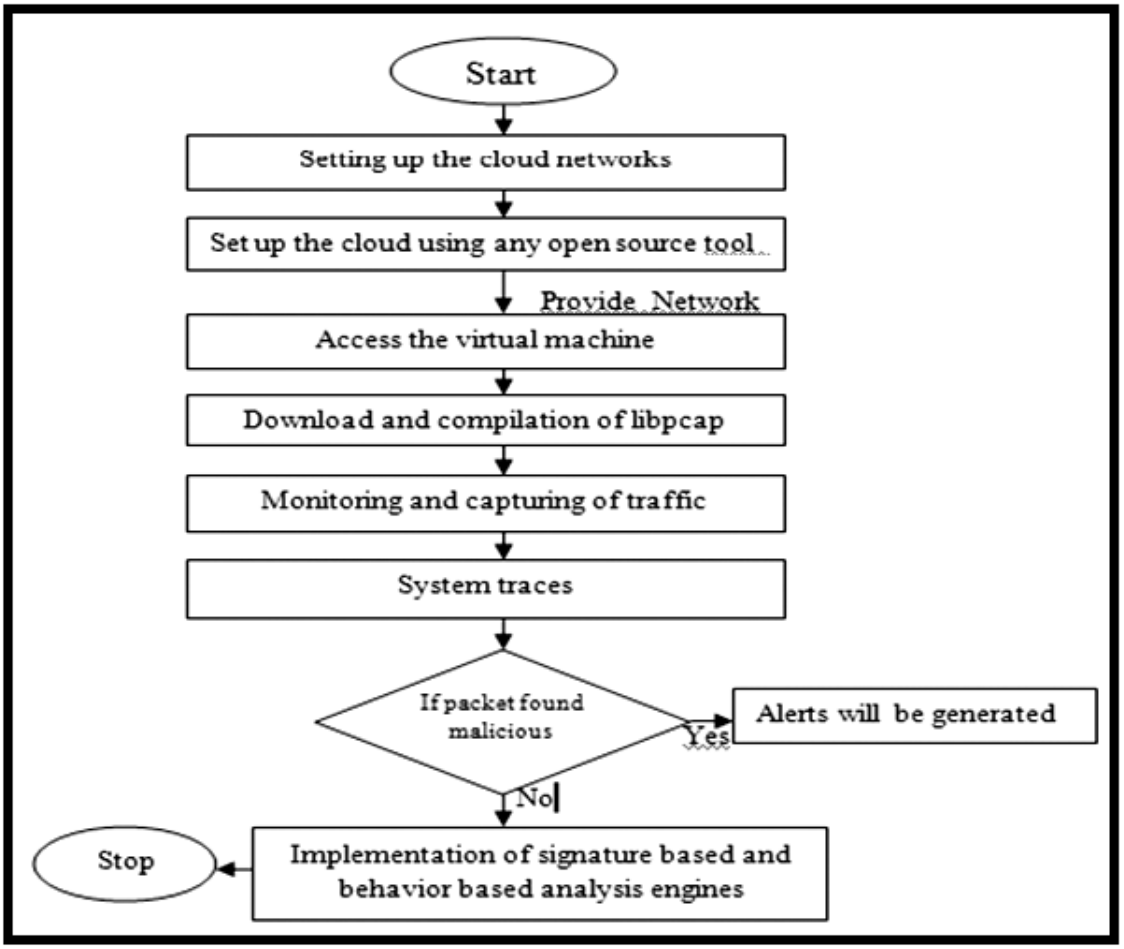

Figure 3: Flowchart of work

Table 1
Performance Metrices

Performance Metrics	*Attribute*	*Descriptions*
	Treal	Total execution time
	Tuser	The time where the programming was running in user space
	Tsys	Tsys is the time spent in system calls

4.1. Results and Evaluation

4.1.1. Treal-overhead comparison with CloudIDEA Treal

In table 2 extract use case has to be taken and the comparison between the total execution time of proposed and existing system has to be done.

Table 2
Treal overhead our system vs Cloud IDEA

Use case: Extract (VM Readiness)- Linux Kernel Extraction	*#syscalls*	*Treal (s) overhead*	*Treal- CloudIDEA*
Extract (VM Readiness)			
None	0	3.58	11.35
Execve	1	6.18	13.038
Open	129	22.13	11.81
All	36379	87.04	116.664

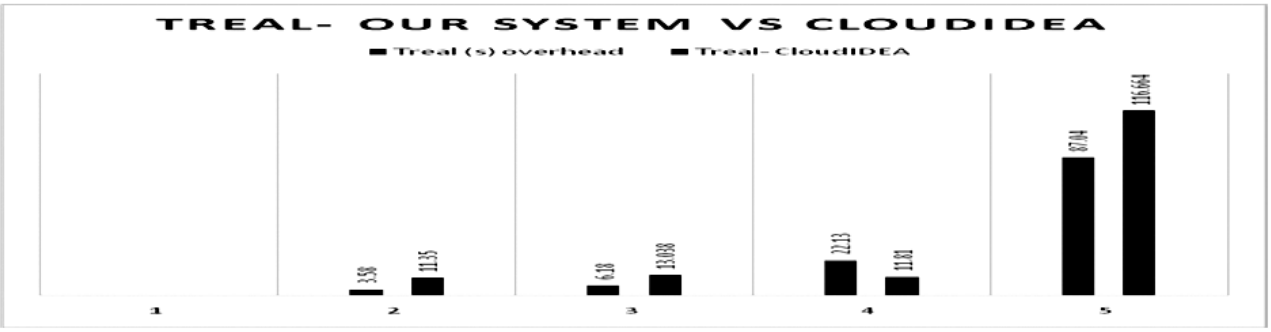

Figure 4: Treal (Overhead) w.r.t. number of syscalls

The figure 4 shows the comparison between the proposed system and cloudIDEA system. The proposed model has been taken less time in comparison with the existing model, which shows the robustness of the proposed model.

4.1.2. Tuser-overhead comparison with CloudIDEA Tuser

In table 3 Tuser overhead using extract use case has to be described and the comparison between the proposed system and the existing system is to be done there.

Table 3
Tuser our system vs CloudIDEA

Use case: Extract (VM Readiness)- Linux Kernel Extraction	*#syscalls*	*Tuser (s) overhead*	*Tuser- CloudIDEA*
None	0	0.34	6.508
Execve	1	2.9	6.56
Open	129	0.259	6.532
All	36379	12.81	6.94

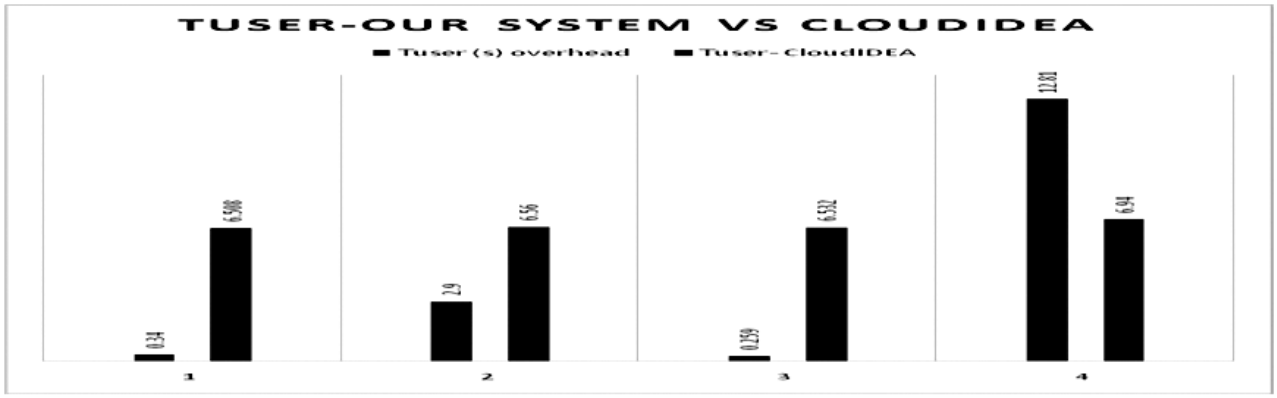

Figure 5: Tuser-our system vs cloudIDEA

The figure 5 shows the comparison between the proposed system and the existing system .The proposed model takes less time than existing system.

4.1.3. Tsys-overhead comparison with CloudIDEA

In table4 the comparison between the Tsys of our system and the existing system using extract use case.

Table 4
Tsys our system vs CloudIDEA

Use case: Extract (VM Readiness)- Linux Kernel Extraction	*#syscalls*	*Tsys overhead*	*Tsys- CloudIDEA*
None	0	0.16	1.664
Execve	1	8.46	1.692
Open	129	0.241	2.384
All	36379	96.62	108.936

In figure 6 the comparison between our system and existing system with Tsys attribute and it shows that our system is faster than existing system.

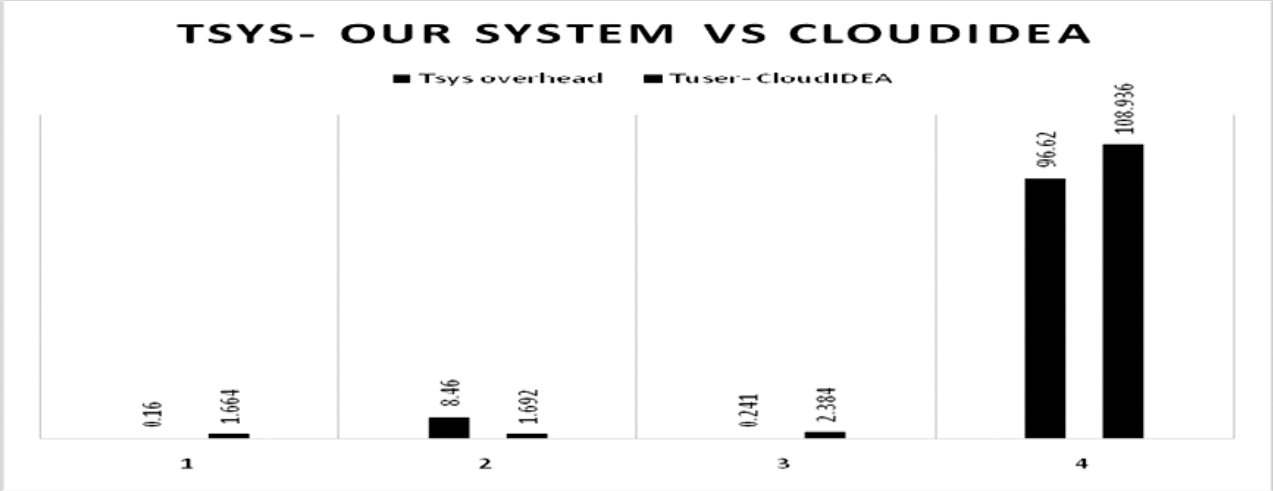

Figure 6: Tsys comparison graph between CloudIDEA and Our Implementation

4.2. Use case: Write a 1.2 Gb file into VM

The table 5 shows the overall readings of use case write. In this table read system call is also taken.

Table 5
Use case- write overall readings

Use case- Write (write a 1.2Gb file into VM)				
None	0	6.27	0.18	0.05
Execve	1	16.81	0.21	0.19
Open	12	6.43	0.27	0.17
Read	1054	59.76	0.26	0.14
All	607k	84.47	0.25	82.55

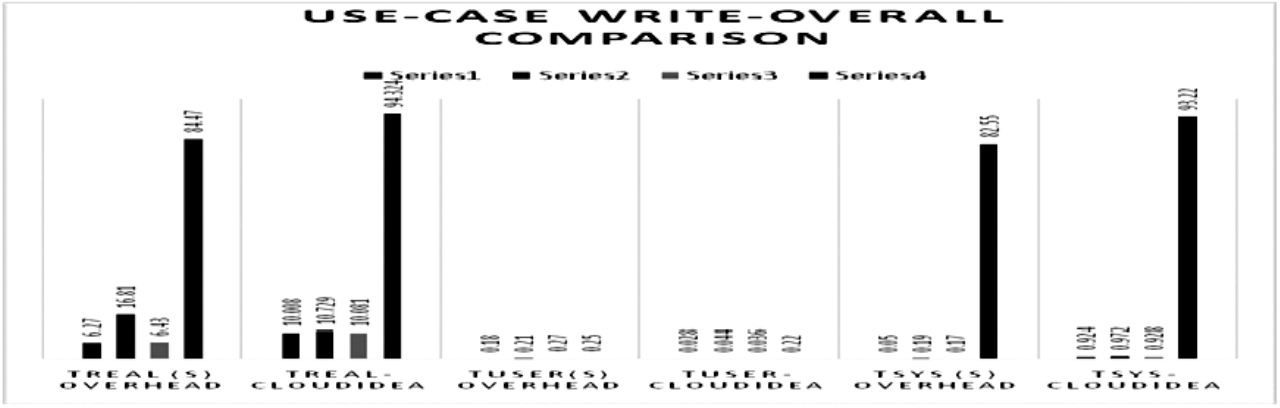

Figure 7: Use case-Write overall comparison with CloudIDEA

The figure 5 shows the overall comparison of all attributes of our system and existing system using use case write.

4.2.1. Treal-overhead comparison with CloudIDEA Treal

The table 6 shows the readings of Treal overhead and Treal CloudIDEA using write use case.

Table 6
Use case- write Treal results

Use case: write	*#syscalls*	*Treal (s) overhead*	*Treal-CloudIDEA*
None	0	6.27	10.008
Execve	1	16.81	10.729
Open	12	6.43	10.081
All	607k	84.47	94.324

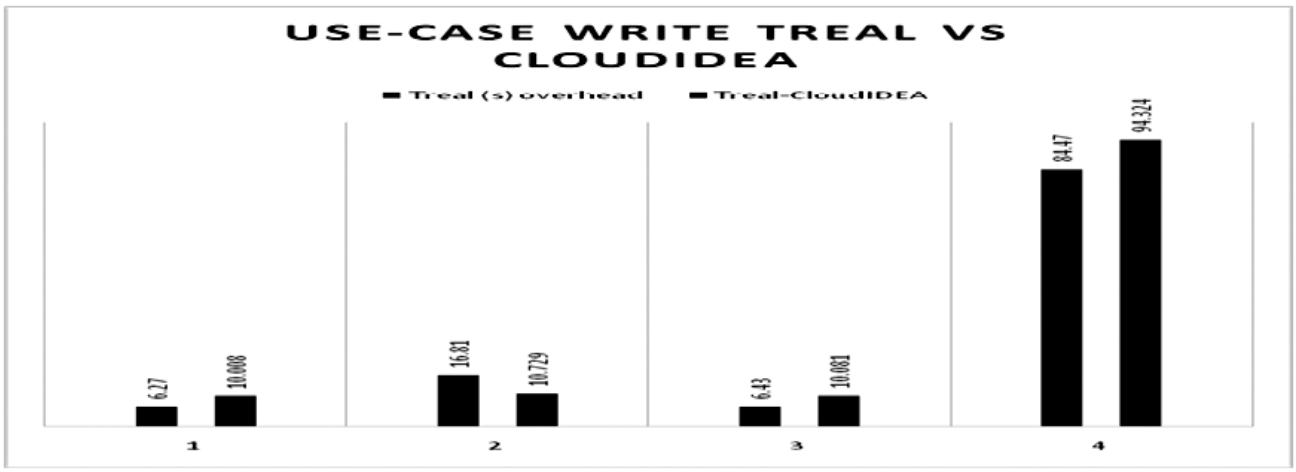

Figure 8: Use case-Write Treal comparison with CloudIDEA

The figure 8 shows the comparison between the Treal attribute of our system and the existing system using write use case.

4.2.2. Tuser-overhead comparison with CloudIDEA Tuser

In table7 the comparison between Tuser overhead and Tuser CloudIDEA is done using write use case.

Table 7
Use Case write- Tuser results

Use case: write	*#syscalls*	*Tuser(s) overhead*	*Tuser-CloudIDEA*
None	0	0.18	0.028
Execve	1	0.21	0.044
Open	12	0.27	0.036
All	607k	0.25	0.22

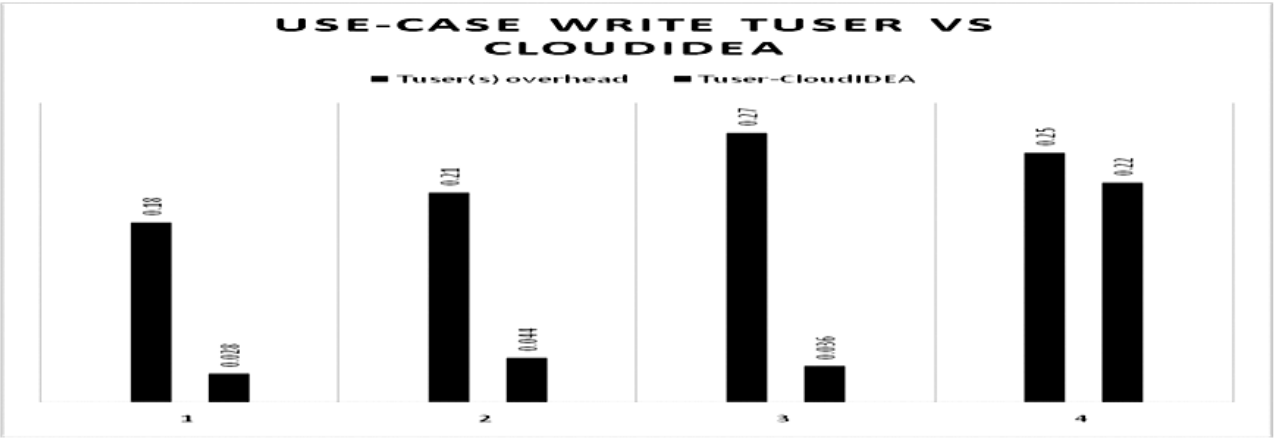

Figure 9: Use case-Write Tuser comparison with CloudIDEA

The figure 9 shows the comparison between our system and existing system of Tuser attribute using write use case.

4.2.3. Tsys-overhead comparison with CloudIDEA Tsys

In table 8 comparison between the Tsys overhead proposed and the existing system using write use case is to be done.

Table 8
Use Case write- Tsys results

Use case: write	*#syscalls*	*Tsys (s) overhead*	*Tsys-CloudIDEA*
None	0	0.18	0.028
Execve	1	0.21	0.044
Open	12	0.27	0.036
All	607k	0.25	0.22

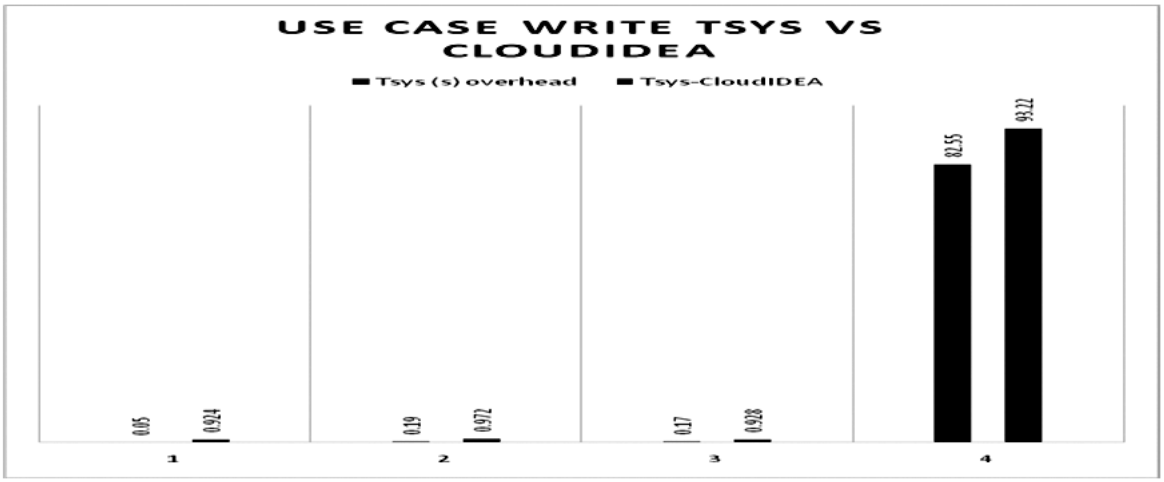

Figure 10: Use case-Write Tsys comparison with CloudIDEA

The figure 10 shows the comparison between use case write proposed system and existing system.

4. CONCLUSION

This proposed design shows scalability therefore it is able to work on any Platform with virtual box. It also shows robustness with provide actual malwares/exploits detailed system logs. It also satisfies the reconfigurability features such as time boundary. As from results and evaluation, we can also conclude that our system is better than base mechanism. In our implementation, it is performing better somewhere as in the case of writing the text file of 1.20B into guest machine. The total of 12762 system calls were observed by monitoring the guest VM, and total execution time was 84.47 whereas in the research paper, the 65.7k system calls were recorded with 94.324 time seconds. The implemented system is over-performing with respect to the research base paper in terms of less system calls (lightweight monitoring) and less time consumed. As future work, our system can be improved if it observed less system calls with less time than our system .

REFERENCES

[1] Anubis. http:// anubis.iseclab.org.

[2] Anwar, F., & Anwar, Z. (2011, December). Digital forensics for eucalyptus. In*Frontiers of Information Technology (FIT), 2011* (pp. 110-116). IEEE.

[3] Deng, Z., Zhang, X., & Xu, D. (2013, December). Spider: Stealthy binary program instrumentation and debugging via hardware virtualization. In *Proceedings of the 29th Annual Computer Security Applications Conference* (pp. 289-298). ACM.

[4] Dinaburg, A., Royal, P., Sharif, M., & Lee, W. (2008, October). Ether: malware analysis via hardware virtualization extensions. In *Proceedings of the 15th ACM conference on Computer and communications security* (pp. 51-62). ACM.

[5] Dolgikh, A., Birnbaum, Z., Chen, Y., & Skormin, V. (2013, June). Behavioral modeling for suspicious process detection in cloud computing environments. In *2013 IEEE 14th International Conference on Mobile Data Management* (Vol. 2, pp. 177-181). IEEE.

[6] Dykstra, J., & Sherman, A. T. (2013). Design and implementation of FROST: Digital forensic tools for the OpenStack cloud computing platform. *Digital Investigation*, *10*, S87-S95.

[7] Fischer, A., Kittel, T., Kolosnjaji, B., Lengyel, T. K., Mandarawi, W., de Meer, H., ... & Weishäupl, E. (2015, October). CloudIDEA: A Malware Defense Architecture for Cloud Data Centers. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"* (pp. 594-611). Springer International Publishing.

[8] Franklin, J., Perrig, A., Paxson, V., & Savage, S. (2007, October). An inquiry into the nature and causes of the wealth of internet miscreants. In *ACM conference on Computer and communications security* (pp. 375-388).

[9] Harrison, K., Bordbar, B., Ali, S. T., Dalton, C. I., & Norman, A. (2012, September). A framework for detecting malware in cloud by identifying symptoms. In *Enterprise Distributed Object Computing Conference (EDOC), 2012 IEEE 16th International* (pp. 164-172). IEEE.

[10] Kirat, D., Vigna, G., & Kruegel, C. (2014). Barecloud: bare-metal analysis-based evasive malware detection. In*23rd USENIX Security Symposium (USENIX Security 14)* (pp. 287-301).

[11] Lengyel, T. K., Maresca, S., Payne, B. D., Webster, G. D., Vogl, S., & Kiayias, A. (2014, December). Scalability, fidelity and stealth in the drakvuf dynamic malware analysis system. In*Proceedings of the 30th Annual Computer Security Applications Conference* (pp. 386-395). ACM.

[12] Marnerides, A. K., Watson, M. R., Shirazi, N., Mauthe, A., & Hutchison, D. (2013, December). Malware analysis in cloud computing: Network and system characteristics. In *2013 IEEE Globecom Workshops (GC Wkshps)* (pp. 482-487). IEEE.

[13] Mather, T., Kumaraswamy, S., & Latif, S. (2009). *Cloud security and privacy: an enterprise perspective on risks and compliance*. " O'Reilly Media, Inc.".

[14] Martini, B., & Choo, K. K. R. (2012). An integrated conceptual digital forensic framework for cloud computing.*Digital Investigation*, *9*(2), 71-80.

[15] Perez-Botero, D., Szefer, J., & Lee, R. B. (2013, May). Characterizing hypervisor vulnerabilities in cloud computing servers. In *Proceedings of the 2013 international workshop on Security in cloud computing*(pp. 3-10). ACM.

[16] Poisel, R., Malzer, E., Tjoa, S.: Evidence and cloud computing: The virtual machine introspection approach. Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA) 4(1), 135–152 (2013).

[17] Radianti, J. (2010, July). Eliciting information on the vulnerability black market from interviews. In*2010 Fourth International Conference on Emerging Security Information, Systems and Technologies*(pp. 154-159). IEEE.

[18] Studnia, I., Alata, E., Deswarte, Y., Kaâniche, M., & Nicomette, V. (2012, November). Survey of security problems in cloud computing virtual machines. In *Computer and Electronics Security Applications Rendez-vous (C&ESAR 2012). Cloud and security: threat or opportunity* (pp. p-61).

[19] Shea, R., & Liu, J. (2013). Performance of virtual machines under networked denial of service attacks: Experiments and analysis. *IEEE Systems Journal*,*7*(2), 335-345.

[20] Willems, C., Holz, T., & Freiling, F. (2007). Toward automated dynamic malware analysis using cwsandbox.*IEEE Security and Privacy*, *5*(2), 32-39.

[21] Zhuge, J., Holz, T., Song, C., Guo, J., Han, X., & Zou, W. (2009). Studying malicious websites and the underground economy on the Chinese web. In*Managing Information Risk and the Economics of Security*(pp. 225-244). Springer US.

[22] Zhang, Y., Juels, A., Reiter, M. K., & Ristenpart, T. (2012, October). Cross-VM side channels and their use to extract private keys. In *Proceedings of the 2012 ACM conference on Computer and communications security*(pp. 305-316). ACM.

[23] Zhang, Y., Juels, A., Oprea, A., & Reiter, M. K. (2011, May). Homealone: Co-residency detection in the cloud via side-channel analysis. In *2011 IEEE Symposium on Security and Privacy*(pp. 313-328). IEEE.

[24] Willems, C., Hund, R., Fobian, A., Felsch, D., Holz, T., & Vasudevan, A. (2012, December). Down to the bare metal: Using processor features for binary analysis. In *Proceedings of the 28th Annual Computer Security Applications Conference*(pp. 189-198). ACM