A Secure and LightWeight Approach for Critical Data Security in Cloud

Sanchika Gupta
Department of E&CE
Indian Institute of Technology, Roorkee
Uttarakhand, India
dr.sanchikagupta@gmail.com

Anjali Sardana
Department of E&CE
Indian Institute of Technology, Roorkee
Uttarakhand, India

Padam Kumar
Department of E&CE
Indian Institute of Technology, Roorkee
Uttarakhand, India
padamfec@iitr.ernet.in

Ajith Abraham
IT For Innovations - Center of Excellence
VSB-Technical University of Ostrava, Czech Republic
*Machine Intelligence Reserch Labs (MIR Labs), WA, USA
ajith.abraham@ieee.org

Abstract— **Cloud computing is a model that provides ubiquitous, on demand access to a shared pool of computing resources including networks, servers, storage, application and services that can be easily provisioned and released. As Cloud is a shared and distributed environment, the need for ensuring security of its critical infrastructure that includes computing, network and storage is immense. One of the critical resources to look after in cloud environment is the data which is stored in files. The files can be configuration file at servers, or private user confidential files at users own work space but they all have a risk of data modification associated with them. If user data is modified through an attack then it will decline the trust of user on cloud services and if the important configuration files are modified, they will disrupt the functioning of cloud environment, like attacker can escalate its privileges and access to critical resources through such tampering and modifications to important files.**

The paper solves the problem addressed and focuses on a proposal and prototype implementation of a tool built for Cloud File integrity establishment and monitoring that establishes and checks file Integrity periodically. The novelty of the approach lies in the fact that the tool does not require any database for storing the integrity of files and the integrity of the file is the compressed encrypted hash of the data stored in the file that can't be reverse engineered by an attacker easily. The tool is lightweight and initial results dictate that it is scalable and efficient. The Real time deployment and analysis of tool is under progress.

Keywords:- Intrusion detection, attacker, Cloud, Security, Monitoring, Integrity, file, signature, hash, monitoring, establishment, Compression.

I. INTRODUCTION

Cloud computing is not a new topic to buzz upon, it was known to the industry since past. But now it is emerging out as a great platform for computing and data storage. The effectiveness of cloud is because of the services it provides to end users. Cloud provides the usage of facilities such as computing and data storage remotely on a pay per usage model. The services are generally provided at software, platform and infrastructure layer and are generally known as software as a service, platform as a service and infrastructure as a service.

Cloud according to NIST is a model that provides access to a configurable pool of shared resources that can be easily provisioned and released on demand. The access to such resources including computing, data storage, application and services is through remote network access facility[1]. The important and remarkable feature provided by cloud to the new era of computing is the easy provisioning of resources and release of them that indirectly increases the efficiency and utilization of important computing and storage resources.

Cloud based on the type of service it provides can be broadly classified into Compute Cloud and Storage cloud. However cloud provides offering of its services at three different layers which are SaaS (Software as a Service), PaaS (Platform as a Service) and IaaS (Infrastructure as a Service).In the model of cloud computing the services are provided by a well defined set of cloud service providers and are availed by a set of cloud service users. The whole concept of cloud computing takes as its base the concept of virtualization [2]. The concept of virtualization provides the facility of creation of virtual replicas of resources including computing, storage , operating system, network etc. [3].
Virtualization can be applied on hardware level to provide virtual copies of hardware resources such as disk, processor etc in which case it is known as hardware virtualization. Virtualization takes many forms including network, desktop and memory virtualization. It provides the way through which physical cloud resources can be efficiently provisioned individually to its users transparently. The word transparently dictates that such virtualization of physical resources in cloud will be seen by the cloud vendors only with users having no information of what is the infrastructure behind.

Currently because of the facilities Cloud provides there is a huge demand of such services in the industry both by individual users and small corporate entities that can't purchase such services but require them on demand. Some of the great providers of such services include Google, Rack space, IBM, Microsoft, Symantec, HP.

As Cloud is getting deployed at a fast pace rate because of the facilities it provides there has been a little focus on the security aspects of it. As cloud is a remotely used shared facility it requires Focus on security of its critical Infrastructure and resources. The Infrastructure of cloud and its resources can be broadly classified into the following four categories:

1. VM Host, Hosts
2. Virtual and Private Network
3. Critical Data or Information.

Whenever a Cloud User request provisioning of computing and operating system resources a VM is assigned to the user that remain allocated to till a predefined amount of period requested by the user. Hence until the VM is released it is the responsibility of cloud vendor to provide Security to the Virtual Machine so that malicious programs must not interrupt with the proper functioning of system virtual entity assigned to the user. It is not only the VM that needs to be secured Cloud should also provide security to other VM in the environment from the attack that can be launched through malicious system calls on other Virtual Machines.

This Problem is solved with the use of Host based Intrusion detection system that detects well-known attacks on VM including VM Escape, VM Hoping etc. Host based IDS are however of many types, based on the kind of technique used for malicious system call detection [19-27]. Such IDS are generally deployed at individual hosts in Cloud environment, or a single IDS is implemented at the privileged node where the activities from all nodes are captured and analyzed [4]. In virtualization environment many of the attacks are VM based attacks that includes but are not limited to VM Escape, VM hoping etc. Such attacks are also handled with the use of host based IDS [5].

Network Infrastructure is also one of the important resources to look after in Cloud as the whole access to the shared pool of resources depends on it. With respect to network we require that the data of the cloud user flowing through the network must remain safe. Also the network should remain available because unavailability of Network will decrease the trust of user in using Cloud services.

Network Availability is attacked majorly through Denial and Distributed Denial of Service attacks. Data confidentiality is tampered through Network and Packet sniffers that sniff the data going on to the network. However many solutions are proposed for taking care of network based attacks including Network intrusion detection systems. Also for preventing secret data loss through sniffing the data in the environment generally flows in encrypted form. In one of our research paper we have also analyzed these intrusion detection systems and found that their effectiveness varies according to their placement and configuration in Cloud computing environment [4].

The third important resource to look after in Cloud environment that people are less focused upon are the system specific or configuration specific files which are responsible for proper functioning of Cloud or the files that are given to each VM users during VM allocation or the important files that resides in user private workspace and which need assurance of integrity. These are not only the configuration files, but the files that are important and existing remotely, so that anybody can have access to them.

The importance of file integrity establishment and checking is necessary because a user trust plays an important role in cloud. For example, if a user has taken cloud services for storing, managing and working on its private data and if that data gets modified by some malicious entity it will destroy the trust of that user on the services of cloud vendor. This is not the only reason for providing security to the files stored in Cloud; one of the other reasons is that Cloud is a automatic provisioning, management and release based service hence it has a collection of configuration files that are assigned and allocated to VM users and which represent the access privileges and configuration specifications, if such files gets manipulated through any malicious entities then it will harm the proper functioning of Cloud.

However such an access for modification of files can be easily prevented in the distributed environment through assigning access roles and privileges. But the problem of data integrity management is not simple in a distributed environment like Cloud. We are taking into picture the problem that such access and privileges can be obtained by malicious users through exploitation of operating system specific vulnerabilities. We identified that the file integrity establishment and reporting tools, which are used today are complex and so cannot be directly applied to the Cloud environment. The complexity of the current file integrity Tools lies in the usage of external and specific databases for integrity management of files. Like many Integrity establishment tools make use of database that stores the integrity of the files.

During integrity checking the integrity of the file is checked for equality with that stored in the database. But the problem remains there only as now it requires additional resources for providing maintenance and security to the databases that stores the integrity information. We have analyzed that the problem of integrity establishment and monitoring can be simplified to be applied in cloud environment by reducing the amount of additional resources, as that will not only decrease the complexity but reduce external dependency with increased level of security. One of the important thing we noted is that in Cloud environment, the amount of files stored are large, hence the storage requirements becomes complex because a separate database for storing file integrity hash key will take additional and a decent amount of extra storage space..

We have also found that some files doesn't require a huge auditing in them, they only require that there data will remain unaltered and if it gets altered by any unauthorized user, the file must be replaced by its replica. Hence in such cases a solution that unnecessarily increases the complexity by introducing severe analysis of files access and integrity checking will not be an optimum solution. We have analyzed that some tools popular in the market do a thorough analysis of file activities but for certain environment and files such a deeper analysis may not be worthy if the goal is only just to maintain integrity. Moreover if we make the detection process complex, a large number of resources are being utilized for providing security and users are left with a small amount of resources at their disposal. And if you see Cloud a large proportion of files that are distributed across (responsible for configuration) the cloud and are responsible for proper functioning of Cloud, requires that they will remain correct as any modification in them may lead to breaches in Cloud security, availability and integrity.

In this paper we have taken as issue the secure and lightweight implementation of integrity establishment and monitoring tool for securing critical data in Cloud environment. We have solved it by proposing and implementing a very lightweight file monitoring tool called as Secure Cloud based File Monitoring Tool also known as SFMT. We have currently focused on securing important configuration and system specific files of VM's in cloud environment. SFMT is a centralized file monitoring tool which runs on privileged domain and is capable to establish and monitor integrity of files residing in other domains in Cloud environment. This tool can also remotely check the integrity of configuration and system specific files to verify that they are not manipulated or tampered by anybody inside or outside the Cloud. This is a periodic task that establishes and verifies file integrity after specified interval of time.

The novelty of the solution lies in the usage of cryptographic checksum over compressed data of file for establishing and verifying the integrity of files. This techniques improves security of integrity (hash) generated for a file so that it can't be reproduced by a malicious entity easily and is lightweight as it does not require any database to store it as the integrity generated remain in the file itself.

II. Related Works

File Integrity Monitoring is a critical instance for the protection and security of unauthorized breaches in the file system. Moreover file integrity monitoring is an essential requirement for NIST 800-53 [6], PCI Compliance [7] and also according to Consensus Audit Guidelines. Many file integrity solutions are proposed in virtual machines and Cloud environment.

The concept of checksum based security measures to verify integrity of file is not new. Previously researchers have proposed techniques for checking integrity of files on Cloud. Hai Jin et al. proposed VMFence [6] which is used to monitor network flow and file integrity in real time. An important well known tool known as Tripwire also do file system monitoring by storing the hash keys of files on a regular~~y~~ basis and storing them in a database.

Pennington et al. [8] proposed storage-based intrusion detection system which allows the storage systems to watch for data modification characters. I3FS [9] intercepts file system calls and injects its integrity checking operations in the kernel mode. It performs checksum comparison in the critical path. NOPFIT [10] is another file system integrity tool for virtual machine which uses multi-byte NOP injection. XenFIT [11] is a file integrity monitor which is implemented on Xen. In XenFIT, the monitored system consist of breakpoints, which intercepts file system calls e.g. open, close, and write. It records the system call log, and sends it to the privileged VM. It is necessary to put an intercepting system call module in the monitored VM, which is easily disabled by the attackers.

There is a great need to monitor the server performance in Cloud environment. Various tools for monitoring data breaches in Cloud are present. File loggers are also proposed by researchers for accountability and transparency in Cloud environment. These file loggers intercepts file access calls and gather the data that can be used to identify the modification and alterations in files distributed across the Cloud.

Flogger [12] is a file centric logger for monitoring file access and transfers within Cloud computing environment. It provides a tool for the end user to check if their files have been tampered. It can be implemented in both virtual and physical spaces in the Cloud providing full transparency.

Tripwire [13] is a host based IDS which alerts on macro changes to the files and folders. Tripwire stores the hash values of the files in a database and compares the current hash values of files with the database values to identify any intrusion. It works in 4 modes and does the actual checking in the check mode. The other 3 modes of operation of tripwire are init, update and test mode.There are various file monitoring tools that are offline and checks system integrity of host files. But those tools cannot be deployed in Cloud environment as the trust on each host cannot be quantified. Tools such as iNotify, File alteration manager which are proposed for verifying file integrity of systems on LAN's, doesn't provides an appropriate solution for vast and scalable Cloud.

In our study we found that the architecture of Cloud requires a lightweight solution that highly efficient and optimized in terms of storage databases and can hence decrease the communication and computational cost in Cloud environment. Solutions such as file loggers and integrity monitors with need for extra resources looks a complex solution for Cloud environment. Our proposed solution separates out as a low complexity solution that stores cryptographic checksum in the files itself. Hence our solution doesn't require any centralized database to store the checksum of files. This provides Cloud a lightweight tool to look at the integrity of files with minimal resources.

III. Proposed Work

The broad overview of proposed scheme is shown in the diagram given in figure1. The working of our proposed file monitoring tool is periodic in nature. The tool works under the control of administrator on privileged domain and its settings such as periodic time interval for checking integrity of files can be set. The tool retrieves files from a location and read it to generate a checksum of its content (which are compressed through compressing techniques using Bzip2 Module in perl) which are then encrypted and are stored in the file itself between specific tags. In Integrity Monitoring the tabs are searched for the extraction of previous checksum, which are then compared with the checksum of the contents (generated in the same way but excluding checksum data in file). If they are found equivalent then it means, no changes have been made to the file and the file is unaltered. However if some changes are made to the file the two checksum (one stored in file and other one currently calculated) will not match. The tools response manager and alert generation component generates alert for the scenario and take necessary action such as replacing the old file with latest version with new integrity established.

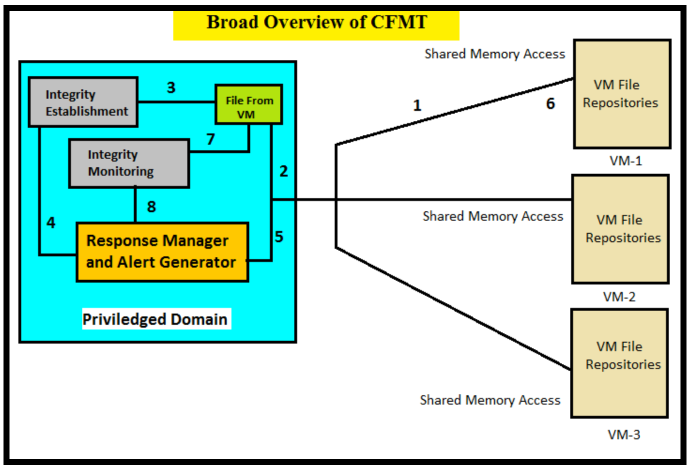


Figure 1. Overview of SFMT

The working of our tool can be explained through the diagram shown in figure1 and is divided into steps given below [14].

1. The first step is to locate the folder and files present in a particular VM (for example VM1). These files can be located through various means. One way is to locate the shared folder that contain folder and files specific to a particular VM.
2. The files are then accessed by the SFMT module.
3. The tool allows selection of the task that can be Integrity Establishment or Integrity Monitoring. In integrity checking the encrypted hash codes are generated for the files and appended into them.
4. The results of the integrity establishment and the new file (with added encrypted hash) are sent to the Response Manger and Alert Generator component.
5. Response Manager sends the new file obtained from Integrity Establishment Component for storing at its location.
6. The file destination is located and the new file is then stored in VM repository of the VM it belongs to.
7. In Integrity Monitoring the encrypted hash codes are captured from the file and are checked with the currently generated encrypted hash code to identify manipulations if any.
8. The results of Integrity Monitoring are given to Response Manager and Alert Generation Module which then report alerts pertaining to manipulation in files if any through sending emails and SMS to respective persons and also take necessary countermeasures.

This Integrity Establishment and Integrity Monitoring process is periodic and is applied on each VM in a sequential manner. The process runs in background and hence requires a very less computational power for carrying out operations. The cryptographic checksum of compressed file contents is appended to the file on periodic basis. The integrity establishment and verification can be set as one time per day or one time per week based on the needs of a specific configuration and environment and is an admin controlled parameter. Whenever the integrity of a file needs again reestablishment the previous line containing the cryptographic checksum of the file is truncated and a new line is inserted to the file containing the new valid cryptographic checksum of the file with its start and end tags.

IV. Implementation Details

We have implemented the proposed scheme of file integrity monitoring with the use of Perl on windows platform. The tool is created as an independent application that can reside centrally on a privileged domain.

The tool uses md5 digest scheme for creating the digest of the file content, which are first compressed through Bzip2 Module of Perl and the crypt scheme that takes the md5 digest as plaintext and specified salt value to return a string. The file is compressed so that the complexity of generating the same hash for two files with different contents can be increased for the attacker. The important quality of the crypt scheme is that the same plaintext and salt generate the same crypt value but there is no known way to get the original plaintext from the generated hash string.

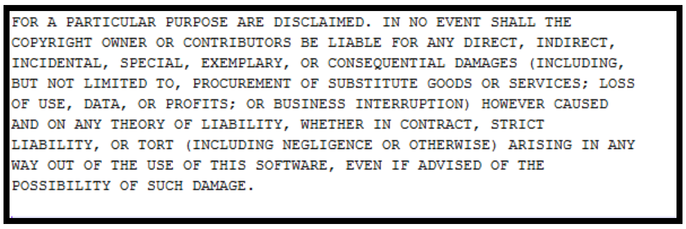
The md5 hash scheme is the traditional scheme of creating a 128 bit or 16 byte digest value. It takes as input any message of arbitrary length and generate 16 byte digest of the message given as input. The implementation is a prototype however more advanced algorithms for hash generation can be used with good collision resistant properties.

During file integrity establishment, after the file is accessed from the VM file repositories it is read as a string and is given to the md5 hash generation module. The generated hash value is then fed into the crypt module which generates the encrypted hash value of the md5 hash value. These values are then appended to the original file between two identical tags. These tags are <Secure> tags which help the monitoring module in locating the encrypted hash codes when the file is checked for integrity monitoring. During file integrity monitoring step, the file is searched for these two identical tags from where the encrypted hash value is fetched. The original file's (after discarding tags and data) encrypted hash value is then calculated in the same way it is calculated during file integrity establishment step.

The two values one which is fetched between the tags and the other currently generated are checked for equality. If the results of equality are true it indicates that the file has not been altered since its last value. However if it's unequal then an alarm is generated and responses are taken by Response Manager and Alert Generator module. One of such responses is to log such an occurrence and replace the modified file in the VM with the original file with added integrity to it.

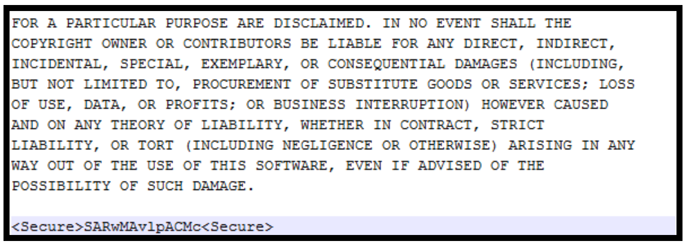
V. Results And Discussions

We have applied the scheme proposed in Cloud for checking and verifying the integrity of files stored on various VM's. The snapshot given in Figure 2 is a part of a file before file integrity checking is applied over it. Figure 3 provides a snapshot of the same file after integrity is established on it. The information between the starting and ending tags is the cryptographic checksum generated for the file.



```
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS
OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY
WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
```

Figure 2: A File snapshot before Integrity Establishment

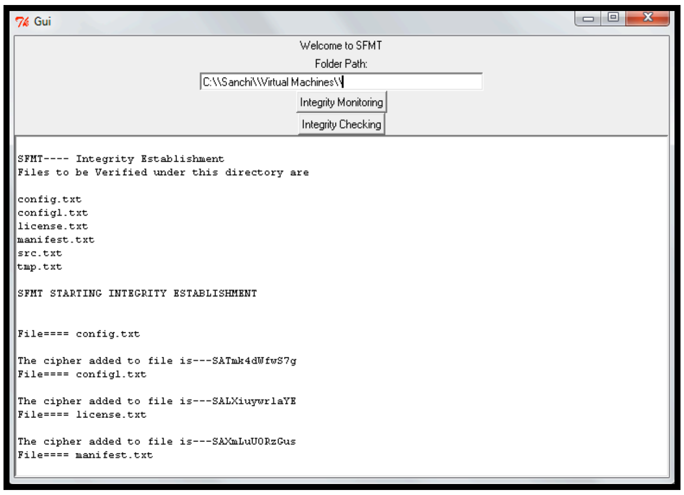


```
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS
OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY
WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.

<Secure>SARwMAv1pACMc<Secure>
```

Figure 3: A File snapshot after Integrity Establishment

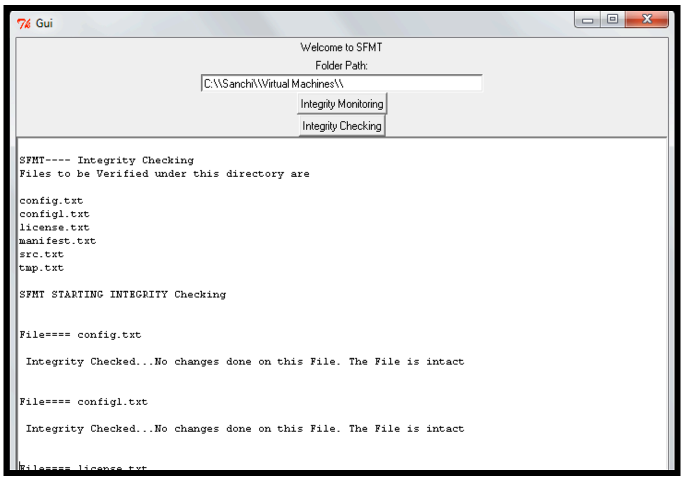
If integrity establishment is specified than all the files in that folder location are retrieved and the cryptographic checksum are appended to them. Snapshot in Figure 4 describes how SFMT extracts each file from folder and provides integrity establishment by appending a cryptographic checksum.

However if integrity monitoring is the requirement then all the files at the particular location are extracted and their integrity checking is performed. Figure 5 provides a snapshot of Integrity monitoring by SFMT on files of a folder shared with a VM.If some variations are found then the list of the files with variations are properly informed and reported by the tool on the console after which the responses can be fired by the administrator to back up the latest files at those locations and creating change logs. Figure 6 describes how a manipulated file is reported through alarm by SFMT. The reason for getting alarm for 'license' file in Figure 6 is that we have manipulated the file by just one character adding to it.
The secret of this scheme lies in the salt, which is used for the creation of encrypted hash from md5 hash calculated and it must be kept securely at the centralized domain. In our case it is fixed however in real implementation with many VM it can have different values for each independent VM for security concerns. The integrity for files is reestablished after periodic interval of time. However integrity monitoring is a recursive process that occurs more frequently over files stored at VM locations so as to maintain integrity of files to its best possible and report alerting instances if any. With encrypted hash added to the files it is nearly impossible for any VM user to manipulate the file for his benefit and generate the encrypted hash value to it so that it will go undetected by the integrity monitoring module. The reason lies in the fact that unauthorized manipulator do not know which secret salt value is used for the creation of encrypted hash even if he knows the hash algorithm used.



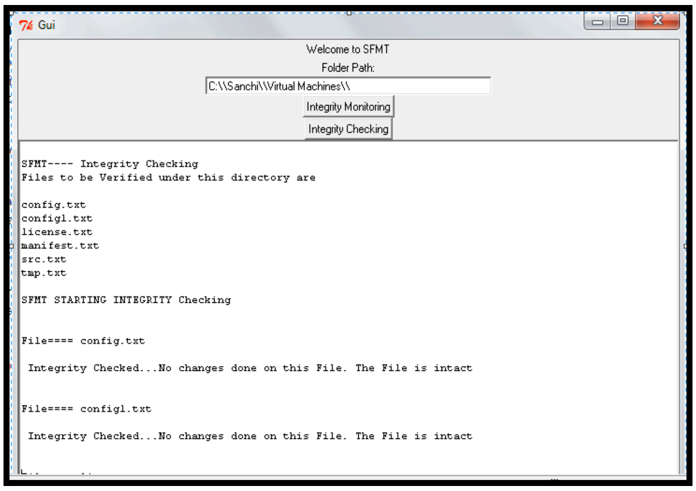
```
SFMT---- Integrity Establishment
Files to be Verified under this directory are

config.txt
configl.txt
license.txt
manifest.txt
src.txt
tmp.txt

SFMT STARTING INTEGRITY ESTABLISHMENT


File==== config.txt

The cipher added to file is---SATmk4dWfwS7g
File==== configl.txt

The cipher added to file is---SALXiuywrlaYE
File==== license.txt

The cipher added to file is---SAXmLuUORzGus
File==== manifest.txt
```

Figure 4. Snapshot of Integrity establishment by SFMT



```
SFMT---- Integrity Checking
Files to be Verified under this directory are

config.txt
configl.txt
license.txt
manifest.txt
src.txt
tmp.txt

SFMT STARTING INTEGRITY Checking


File==== config.txt

 Integrity Checked...No changes done on this File. The File is intact


File==== configl.txt

 Integrity Checked...No changes done on this File. The File is intact


File==== license.txt
```

Figure 5: Snapshot of Integrity Monitoring by SFMT



```
SFMT---- Integrity Checking
Files to be Verified under this directory are

config.txt
configl.txt
license.txt
manifest.txt
src.txt
tmp.txt

SFMT STARTING INTEGRITY Checking


File==== config.txt

 Integrity Checked...No changes done on this File. The File is intact


File==== configl.txt

 Integrity Checked...No changes done on this File. The File is intact
```

Figure 6: Snapshot of Alarm generation by SFMT

VI. CONCLUSIONS

This paper illustrated a prototype implementation of a tool built for Cloud File integrity establishment and monitoring that establishes and checks file Integrity periodically. The algorithm is light weight as it can run in background for file monitoring and require minimal operations on the file for generating and verifying the integrity. Currently we have done our testing with normal text files which are generally used as configuration files and it was found that for 50 files each of 200kb our scheme just takes four and a half seconds for Integrity Establishment and near to five seconds for Integrity Monitoring. This means that the scheme is quite light weight and can be used in Cloud environment effectively with low implementation cost and changes.

References

1. R. Buyya, Y. Chee Shin, and S. Venugopal, "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities," *High Performance Computing and Communications, 2008. HPCC '08. 10th IEEE International Conference on*, pp. 5-13.

2. B. Paul, D. Boris, F. Keir, H. Steven, H. Tim, H. Alex, N. Rolf, P. Ian, and W. Andrew, "Xen and the art of virtualization," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, 2003, pp. 164-177.

3. L. Flavio, and P. Roberto Di, "Secure virtualization for cloud computing," *ACM Journal of Network and Computer Applications* vol. 34, no. 4, 2010, pp. 1113-1122.

4. S. Gupta, S. Horrow, A. Sardana, M. Parashar, D. Kaushik, O.F. Rana, R. Samtaney, Y. Yang, and A. Zomaya, "A Hybrid Intrusion Detection Architecture for Defense against DDoS Attacks in Cloud Environment Contemporary Computing," Communications in Computer and Information Science 306, Springer Berlin Heidelberg, pp. 498-499.

5. H. Jin, G. Xiang, D. Zou, S. Wu, F. Zhao, M. Li, and W. Zheng, "A VMM-based intrusion prevention system in cloud computing environment," *The Journal of Supercomputing*, 2011, pp. 1-19.

6. S.P. Nist, "800-53 Rev. 2," *Recommended Security Controls for Federal Information Systems*, 2007.

7. T. Bradley, *PCI compliance: implementing effective PCI data security standards*, Syngress Media Inc, 2007.

8. G.P. Adam, D.S. John, G. John Linwood, A.N.S. Craig, R.G. Garth, and R.G. Gregory, "Storage-based intrusion detection: watching storage activity for suspicious behavior," *Book Storage-based intrusion detection: watching storage activity for suspicious behavior*, Series Storage-based intrusion detection: watching storage activity for suspicious behavior, ed., Editor ed.^eds., USENIX Association, 2003, pp.

9. S. Patil, A. Kashyap, G. Sivathanu, and E. Zadok, "I3FS: An in-kernel integrity checker and intrusion detection file system."

10. K. Junghan, K. Inhyuk, and E. Young Ik, "NOPFIT: File System Integrity Tool for Virtual Machine Using Multi-byte NOP Injection," *Computational Science and Its Applications (ICCSA), 2010 International Conference on*, pp. 335-338.

11. Q. Nguyen Anh, and T. Yoshiyasu, "A novel approach for a file-system integrity monitor tool of Xen virtual machine," *Book A novel approach for a file-system integrity monitor tool of Xen virtual machine*, Series A novel approach for a file-system integrity monitor tool of Xen virtual machine, ed., Editor ed.^eds., ACM, 2007, pp.

12. R.K.L. Ko, P. Jagadpramana, and L. Bu Sung, "Flogger: A File-Centric Logger for Monitoring File Access and Transfers within Cloud Computing Environments," *Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on*, pp. 765-771.

13. H.K. Gene, and H.S. Eugene, "The design and implementation of tripwire: a file system integrity checker," *Book The design and implementation of tripwire: a file system integrity checker*, Series The design and implementation of tripwire: a file system integrity checker, ed., Editor ed.^eds., ACM, 1994, pp.

14. Sanchika Gupta, Anjali Sardana, and P. Kumar, "A light Weight Centralized File Monitoring Approach for Securing Files in Cloud Environment," *The 7th International Conference for Internet Technology and Secured Transactions (ICITST-2012)*, IEEE [Accepted].

19. Alvaro Herrero, Emilio Corchado, Maria Pellicer and Ajith Abraham, MOVIH-IDS: A Mobile-Visualization Hybrid Intrusion Detection System, Neurocomputing Journal, Elsevier Science, Netherlands, 72(15), pp. 2775-2784, 2009.

20. A. Abraham, C. Grosan and C. Martin-Vide, Evolutionary Design of Intrusion Detection Programs, International Journal of Network Security, Vol.4, No.3, pp. 328-339, 2007.

21. Y. Chen, A. Abraham and B. Yang, Hybrid Flexible Neural Tree Based Intrusion Detection Systems, International Journal of Intelligent Systems, John Wiley and Sons, USA, Volume 22, pp. 1-16, 2007.

22. S. Peddabachigari, A. Abraham, C. Grosan and J. Thomas, Modeling Intrusion Detection System Using Hybrid Intelligent Systems, Journal of Network and Computer Applications, Elsevier Science, Volume 30, Issue 1, pp. 114-132, 2007.

23. A. Abraham, R. Jain, J. Thomas and S.Y. Han, D-SCIDS: Distributed Soft Computing Intrusion Detection Systems, Journal of Network and Computer Applications, Elsevier Science, Volume 30, Issue 1, pp. 81-98, 2007.

24. S. Chebrolu, A. Abraham and J. Thomas, Feature Deduction and Ensemble Design of Intrusion Detection Systems, Computers and Security, Elsevier Science, Volume 24/4, pp. 295-307, 2005.

25. S. Mukkamala, A. Sung and Ajith Abraham, Intrusion Detection Using Ensemble of Soft Computing and Hard Computing Paradigms, Journal of Network and Computer Applications, Elsevier Science, 28(2), pp. 167-182, 2005.

26. A. Abraham, C. Grosan and Y. Chen, Cyber Security and the Evolution in Intrusion Detection Systems, Journal of Engineering and Technology, ISSN 0973-2632, I-Manager Publications, Vol. 1, No. 1, pp. 74-81, 2005.

27. S. Mukkamala, A. Sung, A. Abraham and Vitorino Ramos, Intrusion Detection Systems Using Adaptive Regression Splines, Enterprise Information Systems VI, Seruca, I.; Springer-Verlag, ISBN: 1-4020-3674-4, pp. 211-218, 2006.