Cloud Log Forensics Metadata Analysis

Sean Thorpe
Faculty of Engineering and Computing
University of Technology
Kingston, Jamaica
sthorpe@utech.edu.jm

Indrajit Ray
Department of Computer Science
Colorado State University
Fort Collins, USA
indrajit@cs.colostate.edu

Tyrone Grandison
IBM Research
Yorktown Heights
NY, USA
tyroneg@us.ibm.com

Abbie Barbir
Bank of America
abbie.barbir@bankofamerica.com

Abstract—**The increase in the quantity and questionable quality of the forensic information retrieved from the current virtualized data cloud system architectures has made it extremely difficult for law enforcement to resolve criminal activities within these logical domains. This paper poses the question of what kind of information is desired from virtual machine (VM) hosted operating systems (OS) investigated by a cloud forensic examiner. The authors gives an overview of the information that exists on current VM OS by looking at it's kernel hypervisor logs and discusses the shortcomings. An examination of the role that the VM kernel hypervisor logs provide as OS metadata in cloud investigations is also presented.**

Keywords-Hypervisor,Cloud,Metadata,Logs,Forensics

I. Introduction

While cloud forensics is a field that is still in its infancy, it is gaining traction in the face of proliferate criminal activities taking advantage of the insecurity of these abstract domains. By definition a virtual cloud domain represents a service oriented architecture (SOA) based technology that unlocks the economies of scale benefits for providing the traditional web hosted services. In other words the cloud as a service model offers on demand, elastic and scalable provisions to its networked end users. The cloud deployment model is categorized using networked communities of public, private and hybrid domains of users. Underlined to these cloud service deployment models is the fact that each deployment has a generic set of service layers -: namely the Infrastructure as a Service (IAAS), Platform as a Service (PAAS) and Software as a Service (SAAS) layer. To date vendors like Amazon with its Elastic Cloud Provisions (EC2) and Google are major IAAS providers. Windows Azure, VMWare and Xen Citrix represent the major PAAS and SAAS providers [2]. These service layer designs however inherently lack Forensics and Security within the existing virtualization stack which unfortunately has become an urgent need by law enforcement personnel.

Cloud forensics at this point still does not have a universally accepted definition, but current practices borrow heavily from the existing digital forensics literature in how information retrieval can be supported within these logical domains. Traditional digital forensics is usually split into two separate tasks: disk imaging and disk analysis. Although numerous tools exist that assist an investigator with these tasks [15, 18, 16, 8], there is still plenty of research to be done as it relates to cloud forensics and virtual disk imaging analysis across distributed virtual cloud networks. The challenge in existing digital forensics that concerns disk imaging is well documented; however cloud forensics will still require further risk assessment for these concerns. For all intent and purposes, to understand the concerns in cloud forensic analysis using the kernel hypervisor logs one will need to offer some categorization for which the desired log data can be useful.

One can roughly categorize desired information into the following groups:

1) Information that is available to the VM host operating system and recorded on non-volatile media
2) Information that is available to the VM host system but is not recorded
3) Information that is not currently available to the VM host operating system but could be made available
4) Information that is impossible to be obtained by a VM host operating system

Obviously, a cloud forensic investigator will only have access to the first kind of information suffice to say he has control of the private cloud Local area network(LAN) physical system resources, or, if a live system analysis is performed using well known VM memory introspection techniques within that private LAN. Most of the literature and development in the field of cloud forensics is still a conjecture based on information that is actually present in the first category. One holds the opinion that it is the duty of future research to explore in what manner more forensically relevant information can be provided to an examiner in a feasible fashion. In considering the design of future cloud systems, it is useful to first evaluate what the desired information is, whether and how it can be obtained by a physical computing system, and if it can be stored in a reasonable fashion. This way, some or all information from the second and third categories could be moved to the first one, plus one can then gain an understanding of what is possible.

The exact kind of information and the scope of its cloud storage will differ from one VM host operating system to the other. One can hardly require all VM operating system vendors, especially the large vendors like VMWare, Amazon, Windows Azure, Xen Citirx and so on to start modifying their products to record more data for forensics, or force VM users to enable such logging. However, there are many cases where extra information is desirable, in order to be able to show due diligence, or to more quickly discover if a system was compromised or accessed in an unauthorized fashion.

In the subsequent section, we examine what kind of system information is currently available, which extra information is desirable from a forensics point of view, analyze how the current existing information satisfies those wishes, and how feasible it is to obtain and store information that is not collected on the VM system.

II. Towards Cloud Digital Forensics

To ground the theories that this paper puts forward for cloud forensics, one can consider Casey and Palmer's definition of forensics as "... a characteristic of evidence that satisfies its suitability for admission as fact and its ability to persuade based upon proof (or high statistical confidence)." [11].

When applying this definition to cloud digital forensics, one can see that the area consists of members from a wide spectrum of disciplines and backgrounds. Apart from computer science, digital forensics is relevant to practitioners of disciplines such as law, law enforcement, politics, or standardization bodies. The existing literature in the field of digital forensics has a lot of material to cover and needs to address a diverse audience including digital crime scene technicians, digital evidence examiners, digital investigators [11], lawyers, attorneys, judges, politicians, developers, and researchers. By analogy the same must occur in any reasonable evaluation of cloud forensics as a new field.

Much of the current literature and guidelines for digital forensics focuses primarily on data retrieval and what information is present on existing systems. Given the diversity of the target audience and their different level of computing expertise, one of the main objectives is to teach practitioners the basic procedures of evidence retrieval and analysis on today's cloud computing systems.

Naturally, future research in the field of cloud digital forensics cannot be addressed too thoroughly, although more recent publications also discuss research [23]. Most of the current work explains how to recover data from physical operating system in one form or the other, but not for VM operating systems. In respect to metadata, some of the work also discusses the forensic value and/or quality of the information that is found. By cloud forensic value one means the possibility to draw conclusions about VM events on the VM host operating system where the hypervisor kernel log data repositories are situated. For example, timestamps have a high value from an event reconstruction perspective because they allow an ordering of file operations into a VM event timeline according to [20,21, 22]. This however is dependent on the fact that the VM log timestamps have not been tampered with and that the system's bios clock is correct. Access control information on its own, however, holds less value from a VM event reconstruction point of view, because it generally only reflects static system policies, but does not provide information about individual VM events. The information that can be derived from access control information is a (group of) VM user(s) that may have had access to a VM object on the VM host operating system. At this point further evidence (e.g. in the form of hypervisor kernel syslog timestamps or login data) is needed to draw any conclusions. Under quality one understands how believable the information is.

Is it easy to tamper with the information on the VM host system? For example, on some VM host operating systems a VM file's access and modification timestamps can be arbitrarily set by its private cloud owner.

In some cases, the discussion of VM log evidence retrieval is limited to a description of where important VM kernel system files are located and how to use tools that recover deleted files. Metadata is not discussed at all or only in the form of timestamps [20, 21, 22]. However no critical discussion is given about the value of the forensic information for virtualized cloud system environments. Other publications focus in great detail on the issue of information hiding and retrieval without mentioning file system metadata or issues such as how to obtain time, user, or location information [5].

Some of the current forensics literature actually addresses the value of file system metadata in the form of MAC times and user information [10]. However, a critical discussion about the quality of the information is lacking. At some part of the discussion timestamps are presented as a powerful means to reconstruct events. However, no critical discussion is given [10]. A more thorough discussion about event reconstruction is given by Casey [11]. The actual techniques in terms of functional, relational, and temporal analysis are described on a higher level than what information a system may (reliably) provide. However, the authors makes it very clear that timestamps may be altered and discusses techniques to detect the tampering or deduce the correct times of VM events[20,21,22]. Carrier and Spafford [9] use the term characteristics of a digital object, the set of data and metadata associated with the object, in their event reconstruction model. This reflects the need for reliable metadata information for event reconstruction. They do not, however, discuss what the nature of these characteristics could or should be. The purpose of this paper is to analyze what data can be added and how its forensic quality can be maintained in a virtualized operating system environment.

All of the surveyed literature only describes the information that can be obtained from existing systems and this is their intended purpose. File recovery is the main focus, but a few documents elaborate on the value of timestamps or user information. In a survey of the literature there is no discussion about the requirements of future systems with respect to traditional digital forensics much less cloud digital forensics and what type of meta information beyond MAC times and user information is desired. The discussion shows the increasing level of awareness within the digital forensics community and the tasks that need to be performed in recognition of the volatile nature of digital evidence. What is lacking is an analysis of what information is necessary to perform these tasks or make them easier to perform, and further what kind of desired information can actually be obtained from a VM hosted operating system albeit a private or public cloud provider.

III. History of Metadata

Traditional physical computing systems have some type of long-lived data storage that may be examined for evidence. Even though it need not be the case for every system, by extension the usual organization of VM storage is comprised of hypervisor log files, directories, and metadata abstracted from a physical solid state device with an embedded operating system. For the remainder of this paper one can assume such an organization. VM metadata define all the data in the hypervisor log file system that describes the layout and attributes of the regular VM files and VM directories. This includes attributes such as timestamps, access control information, file size, but also information on how to locate and assemble a VM file or directory in the hypervisor file system. This latter information contains multiple level pointers indexing to data blocks, or even entire blocks used as internal nodes of lookup data structures such as in B-trees.

VM File system metadata was not originally designed to be used for the purpose of reconstructing events that occurred on the VM host system. Anderson [1] was the first to utilize data for threat monitoring within physical operating systems. He proposed to utilize System Management Facilities (SMF) records, which were kept by mainframe servers, such as those running IBM's OS/360. In the 1960s most computing tasks were performed on mainframe computers, with OS/360 one of the dominating operating systems. Information stored on the servers' disks described the entire batch job of a user. The data for the jobs came from punch cards or tape media. The batch job information was kept in records, which described different aspects about the job, some describing the user data (which can be seen as a file). This included file type, minimum and maximum size, creation, access, and modification times, but also information about the job itself such as running times, duration and resources utilized. Compared to today's systems metadata there was more available information for forensic purposes. The information is still present on today's systems but usually not recorded or only in a temporary fashion such as the proc files system. Multics was the first well known operating system that supplied a hierarchical file system, which is generally considered as the ancestor of today's most common file systems. Daley and Neumann describe in the Multics file system design paper [13] the need for users to store their data within the computing environment itself as opposed to storage media such as cards and tape. The user would have complete control and ownership of his data as well as the metadata. They formulated the following design objectives: “Little-used information must percolate to devices with longer access times, to allow ample space on faster devices for more frequently used files. Furthermore, information must be easy to access when required, it must be safe from accidents and maliciousness, and it should be accessible to other users on an easily controllable basis when desired."[13]. To determine how frequently information was used they proposed an access timestamp. The need for modification and creation times came from the file system's backup system, which would commit newly created and modified files to tape backup. This is the original motivation for the use of today's MAC (“Modified, Accessed, Changed") times. To be able to allow other users to access files they proposed the inclusion of an access control list plus permissions (modes) for each file. All remaining metadata had to do with the actual on-disk layout of a file. UNIX was introduced in 1970 and its file system was strongly influenced by Multics. The metadata for a file was stored in an inode and it contained the file's location and size, its type (directory or file), the three timestamps, and the access control information. The latter was comprised of the user and group identifier and protection bits as all modern UNIX variants and derivatives use them today. MS-DOS emerged in the early 1980s. Its file systems, it is well known that File Allocation Tables(FAT), keep’s track of the file type, size, location, and the timestamps. The space reserved for timestamps varies between 2 and 4 bytes, which results in differences in granularity. For example, the access time is only measured in days. Because DOS did not have any notion of a user, no user or permission information is stored with FAT. The Windows operating system at first inherited the FAT file system, but when the limitations of FAT became too much of a problem, NTFS was introduced. NTFS carries detailed user and permission information as well as modified, accessed, created, and changed timestamps.

IV. The Relevance of Metadata for Cloud Forensics

If it were possible to record a system's state register values, memory, timers, network events, interrupt information, etc. for every single clock step, one could use that information to deterministically replay all state events that took place on the VM hosted system. The answers to most questions a cloud investigator might have could be answered. Even if it were possible to record all that information it still would not be feasible as the amount of time necessary to record the information on a VM storage device would slow the system down several orders of magnitude. As this approach is not feasible, one has to utilize snapshots of the VM host system's state instead. A snapshot reflects the VM host system state at a given discrete point in time as a function of the hypervisor kernel logging activities. In addition to knowing the actual state of the system for those points in time, one might be able to draw conclusions about the state changes that occurred between two given snapshots. Taking a snapshot of the entire VM host system's state or parts thereof might be feasible for critical VM host systems within a private cloud domain. In the general case, limited VM storage capacity and performance considerations prohibit this practice. For this reason one needs to consider a further reduction of information quantity and frequency of recording, preferably through an already existing mechanism on the VM host system itself. The hope is that through an audit trail of individual changes to parts of the system (small deltas in the VM host system state) one will obtain sufficient information to understand the changes in the VM host system's meta data state leading up to the current one.

Files play an important role in the operation of most computing systems generally. Usually the operating systems itself as well as the boot mechanism are comprised of files. Program executables, configuration data and startup scripts, user information, as well as application data are stored in files. Therefore looking at the VM hypervisor log files is a good indicator of what actions took place on a VM hosted system: it gives a rough view of information flow, file accesses can show what programs were executed when, and file modifications show what was altered on the VM host system. The operations on the VM host system's hypervisor log files are only a subset of the VM host operating system state. However, for the reasons discussed above, they can yield answers to many questions of interest to a cloud forensic examiner. Furthermore, recording multiple meta information about a VM host file's operations as they occur is a mechanism that introduces little computational overhead. Thus a VM log file's metadata seems a logical place to record the subset of the VM host operating system's state. The metadata associated with a VM file can be seen as the metamorphism characteristics of a digital object as discussed by Carrier and Spafford [9].

Information recorded by the VM host system or user programs may aid the VM forensics investigator during the analysis phase of an investigation. System logging facilities such as the VM hypervisor syslogs or shell history files and third party programs such as the well known Tripwire or even the popular tcp wrappers provide valuable data sources for a forensic investigation. However, programs running outside of the VM host system's hypervisor kernel space may not have access to the necessary information.

Information traditionally stored in log files, are subject to deletion or tampering. Other approaches that modify the kernel [3, 4, 7] either do not store the added information on a permanent basis, as log files have a quantity problem in terms of space considerations, and this becomes particularly observed within the local data center. Other approaches, such as the well known Sun's Basic Security Module for Solaris store extensive audit information in sequential log files using an “audit token" to relate records to each other. The audit records can become quite complicated and large in size and space management can become very complex. Furthermore, the information is not stored directly at the object of interest (i.e. the file) but rather operations on files have to be reconstructed from all of the audit records on the system. By storing the desired information directly as file system metadata one generally gains the following benefits:

- The information is automatically collected and stored by the system: all the information that is available to the system is available to be recorded.
- The information is collected automatically with no extra cost for setting up logging mechanisms.
- The information is directly stored with the object of interest. It is not necessary to correlate various system logs to obtain the desired information.

Tampering with the information is not as simple as tampering with a file. If the VM mapped raw source disk access is not allowed by the hypervisor, the recorded information is protected from all users. Even if VM raw disk access is allowed a malicious VM user still has to navigate the VM host file system to get to the information. When modifying or deleting it he needs to be careful not to destroy any data that is crucial to the successful operation of the VM host system.

V. Desired Information in a Log Cloud Based Investigation

Noted from the overview in Section III, current operating systems and file systems were not designed with digital forensics in mind much more cloud forensics. On most Unix-like systems[6,19], the metadata associated with a file that holds forensically usable information is only 22 bytes, of which 16 are timestamps.

We now provide in this section a discussion on the types of information that is desired from a cloud forensics view point. When performing a forensic investigation on a VM hosted operating system the cloud investigator needs to reconstruct as many VM hypervisor log events and actions that took place on the kernel as a source of historical trace in order to minimize the uncertainty in conclusions[21,22]. In its most complex form a VM investigation will attempt to reconstruct all VM events that took place on a VM host as extrapolated from the kernel logs co-located across distributed Storage Area Network (SAN) disks within say the same private cloud as the issues of reconstruction within a truly public cloud is an NP hard problem and beyond the scope of our work. The main questions a cloud digital investigator has to ask are: who, what, when, how, where and why did a VM event occur. The who question is concerned with which VM user or set of VM users are responsible for certain actions on the compromised VM host. What addresses the actions that were actually performed on the VM host, and over which time interval they took place, and in what manner those actions were executed? This assumes that NTP cloud servers maintain a synchronized and consistent clock mechanism for the physical machine (PM) and VM system boot times.

The where question is to determine both where the responsible VM users were located when they initiated the actions, as well as where the data files on the VM host system came from. This assume that we can use some GPS based coordinate system like a global unique identifier (GUID). By this assumption we can use the SAN LUN otherwise referred to as the CPUID/CPU_WORLDID. Considering however that no established standardization format exist for these GUIDs on the SAN disk, our assumptions here are more conceptual than a reality. Finally, the why question is concerned with the motives that lie behind the actions. The VM hosted system cannot know the intentions of its VM users. Basically the answer to this question is one that the VM investigator must infer from the answers to all the other ones. Due to space limitations an analysis for the who question only is summarized and presented.

A. *Who did it?*

The question of who is responsible for certain actions or the existence of data can be important in the course of a cloud digital investigation. This holds especially true for VM hosted operating systems with a large number of active VM users, such as a virtual server in a thin-client environment or VM hosted systems that offer virtual network portal services. Most VM hosted systems utilize some sort of authentication mechanism usually a login procedure requiring a user name and a password to bind a user identifier and often a group identifier to a process or a VM session.

The VM user and group identifiers for processes and files are also commonly referred to as the “owners" of those instances. It is thus tempting to associate everything that bears such an identifier as the result of that particular VM user's action. However, the original purpose of those identifiers in today's operating systems lies in access control, not true ownership. The only thing that may be deduced by looking at the identifiers is that a process bearing a particular id has been granted the permissions to an object that is associated with that identifier. Thus, in most of today's VM host it is difficult or even impossible to determine the VM user id of the subject that is truly responsible for actions even with suitable access control methods. The information might be gained by correlating other information, such as login times or typical user activities, with the times at which the VM file operations occurred, but this process is tedious and may not even lead to the correct conclusions. If the correct information is stored directly with the file, no correlation work will be necessary.

From a digital forensics perspective the question of who “owns" a file is irrelevant. One wants to know who created, modified, accessed, and deleted it. So who performed those operations on a file? In many cases the VM user id of the file will be equivalent with the identity of the user responsible. There are exceptions, though. For example, a file may be created by VM User A who then changes the VM user id of the file to VM User B. In a Xen Citrix environment running on Unix, this may be done with the chown and touch commands. Such commands are used to transfer permission rights for an object from one VM user to the next, but once executed any notion of the creator of a VM file is lost to the VM host system. For the remaining operations (modify, access, delete) it would make sense to look for the responsible VM users within the set of users that hold the proper permissions for that VM file. This set may be quite large (up to any VM user on the system), and also when the permissions or VM user and group identifiers change, the information deduced from that would be incorrect.

By introducing new fields that associate VM user and VM group identifiers with the various timestamps can help solve the problems addressed above. However, this will not be enough to solve other ones where a user's actions affect a file. For example consider two VM processes, one controlled by VM User A and the other controlled by VM User B as illustrated in Figure 1. Process A read data which is record as a part of a VM log File 1. The two processes communicate using inter process communication (IPC) and VM User A supplies the data it read from the file to B's process, which writes the data into VM log File 2. The creator of the VM file clearly is VM User B. However, VM User A also played an important role in its creation and content. Current VM hosts have no intelligence as to how they can detect say VM User A's effect on the file.

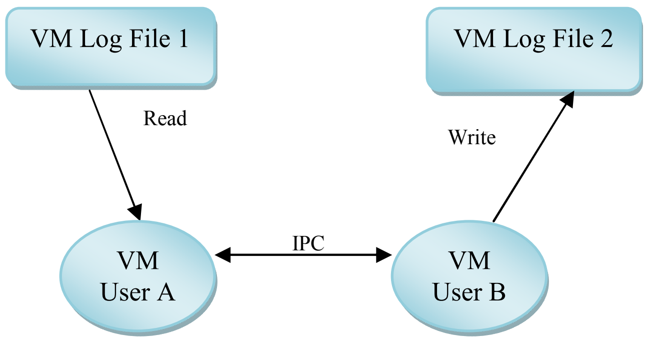

Figure 1. The contents of VM Log File 2 are influenced by VM User A

If one extends the example to more VM processes and VM users that play a role in the creation of the VM file one can observe that a fixed size field to hold user information for a file is not sufficient (unless it is large enough to hold information about all VM users). Additionally, if multiple VM users' processes communicate with each other, how can the VM host tell which ones played a role in a VM file operation and which ones did not? It turns out that this problem is actually intractable for the general case. As the VM host system is generally unaware of the information flow within a process. The only reasonable thing that can be done is observing its inputs (IPC in this case) and outputs (the creation of the VM files). Deciding which input caused an output to occur is equivalent to solving the well known “Halting problem”.

While this problem cannot be directly solved, there are two alternatives. The first is to allow VM programs say within a private cloud to run on the VM host system whose information flow has been predetermined. In this case it is known what data from which input affects the output and the VM host system has this information readily available via the hypervisor kernel logs. But while one obtains the correct and desired information, one loses the ability to run any general program because of the requirement to perform information flow analysis for each VM program independently.

Another technique is to use estimation. Because one cannot tell the actual inputs that affect the output, one simply assumes that all of them had an effect. This approach will result in a redundancy problem i.e. some inconsistent additional information being kept, but it also assures that no correct information is discarded. From a digital forensics perspective this is a good approach for two reasons: if information about a particular VM user is not associated with a file, then one could be sure that the VM user did not have anything to do with the file's operation; also information that is present and might be false is better than no information being present at all. An approach using label sets bound to processes and system objects in [3] could be observed and adopted. One also needs to keep in mind that the information of which VM users played a role in the operations on a file cannot be determined in the general case, given the multiplicity of complex VM events that may compromise even the most basic information requirement needed by the investigator. Ideally if this was possible, such information could be valuable for the cloud forensic investigator.

VI. Conclusion

In this paper the authors have analyzed the role VM host hypervisor kernel log file system metadata plays in cloud forensic investigations. In general one realizes that some of the desired information is impossible to obtain on VM host systems that run arbitrary programs, and hence deemed an intractable problem.

Against this background the future of cloud file system designs will need to be driven by forensics and security playing a more pivotal role within the software engineering requirement and implementation architectures of VM hosted operating systems. Ongoing work at the authors academic and industrial research labs explore the design of private cloud VM log auditing and security tools that can be used to address the set of concerns raised in this paper. Tool analysis specifically focus on visualization and correlation of potential evidence between the kernel log registries of the PM host OS and the VM hypervisor instances running on top of the host(s) .

References

[1] James P. Anderson. Computer security threat monitoring and surveillance. Technical report, James P. Anderson Co., April 1980.

[2] Peter **Mell**. Timothy **Grance** .NIST Definition of Cloud Computing. Retrieved from: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf, September 2009.

[3] Florian Buchholz . Pervasive Binding of Labels to System Processes. In Proceedings of the published Phd thesis, August 2005. Retrieved from : https://users.cs.jmu.edu/buchhofp/publications/buchholz_thesis.pdf2.

[4] Florian Buchholz and Clay Shields. Providing process origin information to aid in computer forensic investigations. Journal of Computer Security, 12(5):753-776, September 2004.

[5] Michael A. Caloyannides. Computer Forensics and Privacy. Artech House, Norwood, MA, 2001.

[6] R.Card, Theodore Ts'o, and Stephen Tweedie. Design and implementation of the second extended file system. In Frank B. Brokken et al, editor, Proceedings of the First Dutch International Symposium on Linux, 1994.

[7] B. Carrier and C. Shields. A Recursive Session Token Protocol for use in Computer Forensicsand TCP Traceback. In Proceedings of the IEEE Infocomm 2002, June 2002.

[8] Brian Carrier. Sleuthkit and autopsy forensic browser. http://www.sleuthkit.org.

[9] Brian D. Carrier and Eugene H. Spafford. Defining event reconstruction of digital crime scenes.Journal of Forensic Sciences, 49(6), 11 2004. CERIAS TR 2004-37.13

[10] Eoghan Casey, editor. Handbook of Computer Crime Investigation. Academic Press, San Diego, CA, 2002.

[11] Eoghan Casey. Digital Evidence and Computer Crime. Academic Press, San Diego, CA, second edition, 2004.

[12] Franklin Clark and Ken Diliberto. Investigating Computer Crime. CRC Press, Boca Raton, FL, 1996.

[13] R.C. Daley and P. G. Neumann. A general-purpose file system for secondary storage. In FallJoint Computer Conference, 1965.

[14] Dorothy E. Denning and Peter F. MacDoran. Location-based authentication: grounding cyberspace for better security. In Internet besieged: countering cyberspace scofflaws, pages 167-174. ACM Press/Addison-Wesley Publishing Co., 1998.

[15] Encase forensic software. http://www.guidancesoftware.com.

[16] Dan Farmer and Wietse Venema. The coroner's toolkit. http://www.porcupine.org.

[17] J. Chapman Flack and Mikhail Atallah. A toolkit for modeling and compressing audit data. Technical report, COAST, Purdue University, 1998. COAST TR 98-20.

[18] The forensic toolkit. http://www.accessdata.com/Product04_Overview.htm.

[19] S. Garfinkel, G. Spafford, and A. Schwartz. Practical Unix and Internet Security. O'Reilly, third edition, 2003.

[20] Sean Thorpe, Indrajit Ray. File Timestamps in a Cloud Digital Investigation. Journal of Information Assurance and Security. ISSN 1554-1010 Volume 6 (March 2011) pp. 495–502

[21] Sean Thorpe, Indrajit Ray. Detecting Temporal Inconsistency in Virtual Machine Activity Timelines. Proceedings of Journal of Information Assurance and Security (JIAS), Volume 7. No.1,April 2012.

[22] Sean Thorpe, Indrajit Ray, Indrakshi Ray, Tyrone Grandison."A Formal Temporal Log Data Model for the Global Synchronized Virtual Machine Environment". International Journal of Information Assurance and Security (JIAS), Volume 6, No 2. 2011.

[23] D.Riley,C.Wren,T.Berry."Cloud Computing :Forensic Challenges for Law Enforcement.Proceedings of the International Conference for Internet Technology and Secured Transactions,London,UK,Nov.2010

[24] Paul J. Leach and Rich Salz. Uuids and guids. http://www.webdav.org/specs/draft-leach-uuids-guids-01.txt.