International Journal of Information Science and Computer Application (IJISCA) Vol 1 (2012), pp 1-10

Formal Hash Compression Provenance Techniques for the Preservation of the Virtual Machine Log Auditor Environment

Sean Thorpe
Faculty of Engineering and Computing, University of Technology, Kingston, Jamaica
thorpe.sean@gmail.com

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: In this paper we provide tamper proof mechanisms for auditing old log entries as a part of lineage provenance within the virtual machine (VM) environment. For each VM provenance log record we apply SHA1 hash checksums, all encapsulated as huffman compressed codes to enforce log preservation against tampering. Our contribution establishes new formal definitions for the VM log provenance. Additionally we have performed base line experimental case studies of how we have started to corroborate these prescribed concerns. For example the significance of using compression, demonstrates the ability of our virtual machine(VM) log auditor disk to preserve the aging log entries for 33% longer than before on the existing VM disk host. We believe these considerations within our research are useful for both the digital investigator and the system administrator who have the arduous tasks for managing the security of these new logical perimeters. This work is motivated by the ongoing work of the authors to demonstrate new security and forensic models for managing log data clouds.

Key words: Provenance, Hash, Log, Cloud, Auditor, Security

1. INTRODUCTION

Database provenance chronicles the history of updates and modifications to data and has received much attention due to its role in scientific data management. Many data security administrators would argue that its acceptance however requires a leap of faith. However the use of provenance information in of itself offers little protection to database records as they are susceptible to accidental data corruption, malicious forgery, a set of problems is often prevalent in loosely coupled environments like the nefarious VM data clouds we have today.

We provide in this paper a tamper proof mechanism for providing provenance to the VM log data running on the physical host operating systems as one technique for providing enforceable audit trails and by extension security within the VM system administered environment. We propose a checksum compression based approach, which is well suited to the

problem of database provenance, including non linear provenance objects and provenance associated with multiple fine granularities for the virtual machine hypervisor kernel log data. We apply this work in the context of our software prototype-: which is a virtual machine log auditor that profiles and manage the VM application identities, events, states and processes, within the this abstract domain. We demonstrate that the proposed solution satisfies a set of desirable security properties, and that the additional time incurred by the checksum approach is manageable, and a feasible approach in practice for auditing virtual machine log entries.

2. RELATED WORK

Provenance describes the history of creation and modification of data. Problems of recording, storing and querying provenance information are increasingly important in data intensive scientific environments, where the value of the data is tied to the method of creation of this data in the first place, and by whom **(Annis et.al, 2002)** . In the decentralized and multi-user environments, we observe that individuals who obtain and use data, do so at that their own risk. They need to trust that the provenance information associated with the data accurately reflects the process by which it was created and refined, but we explore the Cloud trust formalisms in a separate paper. Unfortunately this very data is subject to forgery and data corruption from disparate sources albeit accidental or willful. To this point, integrity has not been a well addressed solution by database provenance. While recent work in the context of file systems exists, the proposed solutions are not applicable to databases within a virtual machine cloud. In particular **(Hassan et.al, 2009)** only considered provenance as a total ordered chain of events on an atomic object (i.e. a file). Traditionally, for databases, provenance is treated as a set of partial ordered events on a set of compound objects (e.g. records, tables or even a whole database).

Throughout the paper, we consider an abstract set of VM participants (users, processes, transactions etc.) that contribute to one or more data objects through insertions, deletions, updates, aggregations. Information about these modifications is collected and stored in the form of a synchronized logical provenance record as the basis of qualifying an auditable log entry of an aging VM record. Various system architectures have been proposed by **(Bhagwat et.al, 2004)** and **(Buneman et.al,2007)** for collecting and maintaining provenance records, from attaching provenance to the data itself as a form of annotation to depositing provenance in one or more repositories . Thus, one of our chief goals is to develop a cross platform solution for enabling tamper resistant provenance for security of logs for our synchronized virtual machine log database schemas. We recognize that it is pointless that since data is collected and shared in a decentralized and loosely coupled manner, it is impractical to use secure logging tools like that of our virtual machine log auditor that rely, for example on trusted hardware or other system level assumptions about secure operations.

Occasionally, a data recipient will request and obtain one or more of these data objects. In keeping with the vision of using provenance in the first place, each database log object running on the auditor's back end database is accompanied by a provenance object. We seek to establish this by providing basic cryptographic proofs and well founded assumptions to the System administrator that the provenance object has not been maliciously altered or forged. The closest work to ours is that described **(Hassan et.al,2009)** which focused on security problems (integrity and confidentiality) that arise when tracking and storing provenance in a file system . While our work utilizes a similar threat model and integrity checksum approach, we must deal with a significantly more complicated VM data model (i.e. compound VM data objects) and provenance model in order to apply these techniques in a database setting.

A recent vision paper by **(Miklau et.al,2005)** considered the problem of data authenticity on the web, and described a pair of operations (signature and citation) for tracking the authenticity of derived data. One of the main differences between that work and our work is the structure of participants' transformations. The previous work assumed that transformations were structured in a limited way (specifically, as conjunctive queries), whereas we consider a more dynamic scenario of VMs in an arbitrary black box of such transformations. The general problem of logging and auditing for cloud databases has become increasingly important in recent years. Research in this area has focused on developing queryable VM audit logs and tamper evident logging techniques like those discussed in the papers by **(Snodgrass et.al,2004)**and **(Tan et.al,2006)**.

In addition, there has been considerable recent interest in developing authenticated data structures to verify the integrity of query results in dictionaries, outsourced databases, and third party data publishing as discussed in papers like those of **(Devanbu et.al, 2000)**, **(Frew et.al,2008)**, and **(Foster et.al,2002)**.

Finally, the provenance community has begun to think about security issues surrounding provenance records and annotations. Authors like **(Braun et.al, 2008)** and **(Tan et.al, 2006)** motivate the need for understanding the complications in provenance systems. Several systems have implemented a provenance system to protect information from unauthorized access: for provenance in a service oriented environment (SOA); and for annotations. The Cloud computing community which happens to be a more recently introduced community within the SOA environment is like other groups seeking to find secure methods of releasing information. For an abstract domain as the virtual machine cloud this is a challenging problem.

3. CONTRIBUTIONS AND PAPER OVERVIEW

This is the first study as far as the literature reveals on the integrity and tamper proof mechanisms provided for database provenance within the compute cloud environment. While related work has focused on security (integrity and confidentiality) for file system provenance, we extend the prior work in the following important ways:

Non Linear Provenance for virtualized log databases - Database operations often involve the integration and aggregation of objects. One might consider treating an object produced in this way as if it were new (with no history), but this discards the history of the objects taken as input to the aggregation. Thus, in databases it is common to model provenance in terms of data aggregation (DAG) or non-linear provenance.

Compound VM log objects - In a virtualized database, it is critical to think of provenance associated with multiple granularities of data, rather than to simply associate provenance with atomic objects. For example, in the relational data model, each table, row, and cell has associated provenance, and the provenance of these objects are inter-related.

The remainder of this paper is organized as follows: In Section 4, we lay the ground work by describing the database provenance model and integrity threat model for a virtual machine log auditing environment. We then present tamper proof provenance techniques for an atomic and compound VM logic object.(i.e. Section 5). In Section 6, we provide the type of VM threat model that impacts the provenance arguments from Section 5.In Section 7 and 8 we present the cryptographic basics of our arguments. In Section 9 and 10, and 11 we present the proof of concept experiment. And finally in Section 12 we provide the conclusion and future work.

4. PRELIMINARIES

We begin with the preliminary building blocks of our work, which include the basic VM provenance model and integrity model. As a short precursor definition, the VM data clouds is predicated on the service oriented architecture design principle of providing IT resources as an on demand, elastic service to end users (**Mell et.al., 2009**).The VM data cloud supports three (3) deployment layers of service namely-: Software as a Service (SAAS), Infrastructure as a Service (IAAS), and Platform as a Service (PAAS). Both the IAAS and PAAS are system layer services, while SAAS is an application layer service within the virtualization stack. In our work for log audit provenance, we focus at the PAAS system layer of this virtualization stack, since we handle the error, system, and application logs stored on the hypervisor kernel. Throughout this paper, we will consider a VM log system database D, consisting of a set of VM log data objects. Each VM object has a unique identifier, which we will denote using a capital letter, and a value. We will use the notation A.val to refer to the current value of a VM log object A. Our log auditor's database supports the following common operations:

- Insert (A,val): Add a new object A to D with initial value val .
- Delete (A): Remove an existing object A from D.
- Update (A, val^1): Update the value of A to the new value val^1
- Aggregate ({A_1 ……..A_n },B); Combine objects A_1 ……..A_n to form a new object B.

5. VIRTUAL MACHINE LOG PROVENANCE MODEL

In this section we formally define the intuitive definitions for the virtual machine log provenance. Log provenance describes the relationship between the result of a transformation and the inputs that contributed to it. In a relational setting, this is usually interpreted as the input tuples of some VM hypervisor log query q that contributed to an output tuple t of q. In our study to date there is still no evidence where log

provenance for the VM environment has any sufficient use cases that address its application. For this reason our approach is still subject to an enterprise implementation of these concerns. Notwithstanding a conceptual design outlook should provide initial specification models for which we could seek to assess the implementations. We adopt our ideas based on existing techniques used within the PI-CS (Perm influence Contribution Semantics) and Lineage CS literature for applying data provenance.

We scale these contribution semantic definitions to define new ones for the hybrid virtual machine environment. For the purposes of this work we refer to such contribution as *VM- LCS* (virtual machine log contribution semantic). Let's assume that the *VM-LCS* models the provenance of a result tuple t of a query q as a list W (q,t) = $<Q^*_1 \ldots\ldots Q^*_n>$ of subsets Q^*_i of the inputs Q_i of the query(where the inputs could be base relations or the result of other log queries) that contribute to t. One should not seek to model provenance as an independent set of tuples as it has the distinct disadvantage that the information about which input tuples were combined to produce a result is not at all modeled. In other words input tuple correlation is critical and complicit within our VM-LCS. Hence to explicitly model which tuples were used together in the creation of an output tuple, we consider the provenance representation from a list of subsets of the input relations to a set of *log witness list*. Thus a log witness list w is an element from $<Q^{\in}_1$ x…..x $Q^{\in}_n>$ with $Q_i{}^{\in} = Q_i \cup \perp$. Thus a log witness list w contains a tuple from each input of an operator or the special value $\perp$. The value $\perp$ indicates that no tuple from an input relation belongs to the log witness list. This is particular useful if we consider the relational algebra implication, in that one can model outer joins and unions which are both important in the integration of the VM log data sources and targets. Each log witness list represents a combination of input relation tuples that were used together to derive a tuple. We can now represent this contribution as a formal definition

Definition 1: (VM-LCS provenance). For an algebra log operator op with inputs $Q_1 \ldots\ldots Q_n$, and a tuple $t \in op(Q_1 \ldots\ldots Q_n)$ a set $P(op,t) \subseteq W = (Q^{\in}_1$ x…..x $Q^{\in}_n)$ where $Q_i{}^{\in} = Q_i \cup \perp$.is the VM-LCS of t if it fulfills the following conditions:

(1) $op(P(op,t)) = \{t\}$
(2) $\forall w \in P(op,t) : op(w) \neq 0$
(3) $\neg\exists P^1 \subseteq W : P^1 \supset P\ (op,t) \wedge P^1 \models (1),(2),(4)$
(4) $\forall w, w^1 \in P(op,t) : w \curlyvee w^1 \Rightarrow w \notin P(op,t)$

The first condition (1) in the Definition 1 checks that the provenance produces exactly t and nothing else by computing the result of the operator op over the provenance. The second condition (2) guarantees that each log witness list w (combination of tuples) in P contributes something to t (removes false positives). The third condition (3) checks that P is the maximal set with these properties, meaning that no witness lists that contribute to t are left out. The provenance of a VM log query is computed recursively applying Definition 1 to each operator of the query. The fourth condition (4) is necessary to produce precise provenance for outer joins and set union. This condition states that we will exclude a witness list w from the provenance, if there is a smaller witness list w^1 in the provenance that subsumes w. A witness list w is subsumed by a witness list w^1 (denoted by : $w \curlyvee w^1$) iff w^1 can be derived from w by replacing some input tuples from w with $\perp$.

The second formalism which we'll discuss is transformation provenance. We model the parts of transformation provenance that contribute to an output tuple. In contrast to traditional data provenance, transformation provenance is operator centric. In retrospect one could say that transformation provenance is similar to approaches in workflow management systems. We articulate the VM log transformation provenance as transformation q using an annotated algebra tree for q. For an output tuple t and a witness list w in the data provenance of t, the transformation provenance will include 1 and 0 annotations on the operators of the transformation q. A 1 indicates this operator on w influences t, a 0 indicates it doesn't.

Definition 2: (Annotated Algebra tree). An annotated algebra tree for a VM log transformation q is a pair ($Tree_q$, θ) where $Tree_q$ = (V,E) is a tree that contains a VM log node for each algebra operator used in q (including the base relations as leaves) and $\theta : V \in Tree_q \rightarrow \{0,1\}$ is a function that associates each operator in the tree with an annotation from {0,1}. We define a preorder on the nodes to give each node an identifier(and to order the children of binary operators). Let I (op) denote the identifier of the VM node representing operator op.

Intuitively each witness list of the log data provenance of tuple t represents one evaluation of an algebra expression q . For each witness list w, each part of the algebra expression has either contributed to the result of the evaluation q on w o not. Therefore, we present the transformation provenance as a set of annotated algebra trees of q with one member per witness list w. We use data

provenance to decide whether a log operator op in q should get a 0 or a 1 annotation. Essentially if evaluating the subtree sub_{op} under op on w results in an empty set ($sub_{op}(w) = \varnothing$) , then op has contributed nothing to the result tuple t and should not be included in the transformation provenance. These formalisms can be summarized below in Definition 3:

Definition 3: (Transformation Provenance). The transformation provenance of an output tuple t of q is a set T(q,t) of annotated trees defined as follows:

$T(q,t) = \{ (Tree_q , \theta) \mid w \in P(t) \}$

$\theta_w (op) = \{ 0 \text{ if } sub_{op}(w) = \varnothing \text{ else } 1 \}$

As a precursor to Definition 4, let's look at the characteristics required for VM log mapping provenance.. In a mapping scenario, transformations may be derived from a set of declarative schema mappings. For most non trivial mappings several transformations exists that implement these mappings correctly (they produce a target instance that satisfies $\sum_{st}$ and $\sum_{t}$). A single transformation may implement more than one mapping or vice a versa. For debugging specifications we do not only like to know what parts of the transformation produced a target tuple t, but also from what mappings these transformations were derived.

Hence we, define mapping provenance based on transformation provenance and the relationships between transformations and mappings. The relationship between a mapping and part of a transformation is modeled by adding new annotations (specifically mapping identifiers) to the algebra tree for transformation. Now for creating the VM algebra tree for such mapping transformations, let's use the formalism that $Tree_q = (V,E)$, where we introduce one new annotation function , μ_m , per mapping $M \in \sum_{st.}$ The function μ_m is 1 for each operator that implements this mapping and 0 otherwise. For example, consider the mapping $M_1 : S(a) \Rightarrow \exists b: T(a,b)$ and $M_2 : S(a) \wedge R(a,b) \Rightarrow T(a,b)$ for target relation T with schema T = (d,e), and R, and S are relation sets on M. We summarize this formalism in Definition 4 below:

Definition 4: (VM log Mapping Provenance). The VM log mapping provenance M(q,t) for a tuple t from the result of a log query q is defined using the mapping annotation functions μ_m over the VM log transformation provenance T(q,t) as follows :

$M(q,t) = \{ M_w \mid w \in P(q,t) \}$

$M_w = \{ M \mid \forall op \in V: \mu_m(op) = \theta_w (op) \}$

Definition 5: (VM Log Provenance Object) - The VM log provenance of a data object, A, consist of a set of VM Log provenance records, which are partially ordered by a sequence identifier (seqID). Alternatively it is easy to think of the provenance object as a data aggregation (DAG). Each VM log data object has a single most recent record , with the greatest value seqID. For simplicity, we will assume that a seqID values are assigned in the following way: When a new VM Log data object is inserted its initial seqID is equivalent to the timestamp of the logged entry on the VM kernel \var\essx3i\log directory. For each such entry we keep a secondary index field called logseqID which starts at zero (0) to record the first entry in the log. Each newly recorded time-stamped log entry increments by 1 to mark the latest or most current entry on the VM log data stack.

With the exception of deletion, each operation is documented in the form of a VM log provenance record. (For the purposes of this paper, after a VM log object has been tagged deleted, its VM log provenance record is no longer relevant.) The VM log auditor explores generating "Deleted provenance" databases that manage such VM log entries in the face of its use in cloud forensic investigations. We model each log provenance record as a quintuple of the form (seqId,LogseqID,e,{ (A_1 , v_1) ……, $(A_n v_n)$). E represents the entity that performed the operation i.e. that is a person based on login a/c and MAC address). (A_1 , v_1) ……, $(A_n v_n)$ describes the (set of) input object(s) and their values. (A,v) describes the output object and its value. seqID and logseqID as composite keys helps to describe and confirm the relative order of the provenance records associated with specific objects. We don't worry about the SAN disk performance overhead of using composite key indices, as this data set is a compressed provenance record entry. If two (2) records say rec1 and rec2 involve the same object (with the same id) as either input or output,then rec1.seqID.logseqID < rec2.seqID.logseqID indicates that the operation described by rec1 occurred before the operation performed by rec2.

6. VM LOG THREAT MODEL

In the absence of additional protections, the VM log provenance records and objects described in the previous section are vulnerable to illegal and unauthorized modifications that can go undetected. Throughout this paper our objective is to develop an efficient scheme for detecting such modifications. In

this section we outline our threat model and desired guarantees which are a variation of those offered by **(Hasan et.al,2009)** . In particular , consider a data object A and its associated provenance object P. Suppose that P accurately reflects the provenance of A, but that a group of one or more attackers would like to falsify history by modifying A and or P. Worse case the attackers of the virtual machine cloud networks are themselves insiders(participants). We set out the following desired guarantees with respect to a single attacker :

R1: An attacker (participant) cannot modify the contents of other participants' provenance records (albeit input or output values) without being detected by a data recipient.

R2: An attacker cannot remove other participants' provenance records from any part of P without being detected by a data recipient.

R3: An attacker cannot insert provenance records (other than the most recent one) into P without being detected.

R4: If an attacker modifies (updates) A without submitting a proper provenance record to P documenting the update, then this will be detected by the data recipient.

R5: An attacker cannot attribute provenance object P (for data object A) to some other data object, B , without being detected by a data recipient.

In summary, we should be able to detect an attack that results from modifying a VM Log provenance records that has an immediate successor. Also, we must be able to detect any attack that causes the last provenance record in P to mismatch the current state of object A. In addition, it may be the case that multiple participants collude to attack the provenance object. In this case, we seek to make the following guarantees:

R6: Two colluding attackers cannot insert provenance records for non colluding participants between them without being detected by a data recipient.

R7: Two colluding attackers cannot selectively remove provenance records of non colluding participants between them without being detected by a recipient.

R8: Participants cannot repudiate provenance records.

In our work it is important to point out the distinction between threat detection and threat prevention. Our log auditor prototype only handles threat detection , and hence our goal enables detection of tampering; and not denial of service types attacks. Take for example the case example, of an attacker who nefariously modifies data or provenance objects to prevent the information from being used. We also do not address in our current work issues of forged authorship (piracy) in which an attacker makes copies of a data object, and claims to be the original creator of the data object.

7. CRYPTOGRAPHY BASICS

We will make use of some basic primitives. We assume a suitable public key infrastructure, and that each VM log auditor participant is authenticated by a certificate authority.

Hash Functions: In our current log auditor design we use cryptographic hash functions e.g. SHA1. We denote this function as h(). Generally speaking, h() is considered secure if it is computationally difficult for an adversary to find a collision. i.e. $m_1 \neq m_2$ such that $h(m_1) = h(m_2)$.

Public Key Signatures - We assume that each participant p has a public and secret key , denoted PK_p and SK_p. p can sign a message m by first hashing m, and then encrypting h(m) with this secret key. We denote this as $S_{skp}(m)$. RSA is a common public key cryptosystem.

8. LOG PROVENANCE INTEGRITY FOR ATOMIC AND COMPOUND OBJECTS

We begin with the simple case in which we have a VM Log auditor database D comprised of atomic event log objects. In this case, we propose to provide tamper evidence by adding a provenance checksum to each provenance record. This checksum provision applies both Merkle Hash trees and Huffman compression to enforce losslessness on the data set. In the case of linear provenance (operations comprising of only insertions, updates, and deletions).

9. EXPERIMENTS

Section (10) and (11) describe our experimental setup and analysis respectively.

10. EXPERIMENTAL SET-UP

At the University of Technology [UTECH] we demonstrate the design of a virtual machine log auditor using Windows 7 based VMWare essx3i data centre. We setup an Oracle 11g relational database to run an

independent Windows 7 hosted physical server with its own Storage Area Network (SAN) terabyte disk. Our VM log audit server runs periodic ftp sessions to the production SAN to retrieve the system event and application event logs and sends this log data back to our test SAN. We use a SENDER script to complete this process. Note this is a shell script. The log auditor runs a LOADER batch file script to parse the retrieved ftp logs to the Oracle 11g relational database (**Thorpe et.al 2011**) . It is at this stage we assign a provenance checksum as well as the compression of this hash. Presently the auditor checks the provenance logs for frequency of occurrence and anomalies in such log data occurrences.

The frequency of the VM log data anomalies is critical to forensic analysis, but we'll not discuss that here. We had been particularly concerned about how the log auditor's database will maintain its own security if these provenance logs experience exponential growth on the SAN disk. To this end, we have applied Huffman compression codes to the hashed log auditor database.

The log auditor's front end application runs on a Java Web enabled browser. We use the java.security.message Digest ("algorithm SHA") which generates a 20 byte message digest. And we use java.crypto.Cipher to generate a 128byte signature (given a 1024 byte key). Our Huffman compression algorithm is built using C++ code which we import into our Java Application as a remote procedure call. For the purposes of this paper we take a sample of system event logs and application logs over different time points to demonstrate our proof of concept. These event logs can be viewed as compound objects of several provenance record entries. Each VM log event transaction is a time-stamped transaction and hashed based on the sequence of each record loaded to the log auditor.

11. EXPERIMENTAL ANALYSIS

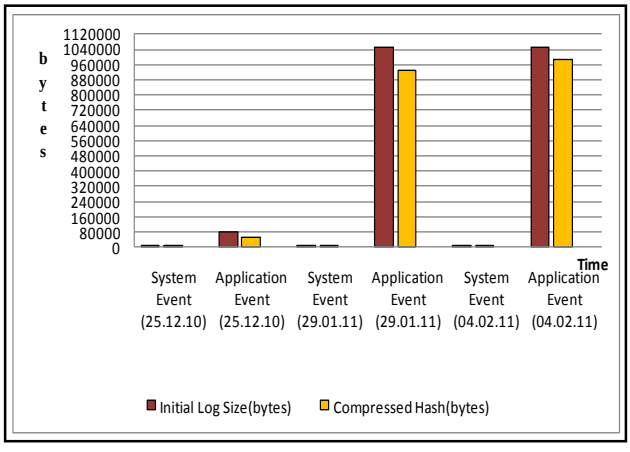
We analyze the impact of our approach, by running a series of hash compression on these logs as seen in *Table 1*. We use three (3) System log events and three (3) Application log events over similar time intervals as pairs of events to correlate log database performance with the disk before and after we apply the compressed provenance hash. The logs analyzed are snapshots of the actual sizes of the files we captured from the VM host. Here are the actual results generated in Table 1:

TABLE I. : REPRESENTS THE TABLE OF SYNCHRONIZED SYSTEM AND APPLICATION LOG EVENTS STORED BY OUR VM LOG AUDITOR DATABASE.

Log Type	Log Run Date	Initial Logsize (Bytes)	Compress-ed Hash (Bytes)	Log Disk Rate Optimization (%)
System Event	25/12/10	1823	1243	33%
System Event	29/01/11	3475	2273	35%
System Event	04/02/11	3475	2273	35%
Application Event	25/12/10	81093	54610	33%
Application Event	29/01/11	1047565	930190	37%
Application Event	04/02/11	1052043	985347	35%

From table 1 the hash compression on log entries shows a general performance improvement on the Storage Area Network (SAN) disk utilization by a range of 33% to 37%. Although the disk utilization is not principal to our log study in this paper, the results are useful for two reasons. (a) As the rate of the logs grow we recognize that the optimality on the disk improves as we process the logs for provenance (b) For the smallest size comma delimited log files , the optimality of performance is comparable with the largest log file databases also available within our black box. Hence the throughput of the log audit provenance processing has little overhead to our results. In *Figure 1* below we use the graph tabulation to provide further empirical analysis discussion to these results.

Figure 1: Snapshot Analysis of System Event Logs versus Application Logs



Following on from our observations in Table *1*, we use *Figure 1* to capture the consistency of performance improvement between system and applications logs for the different log sizes from the sample in this study. The application logs denote the changes in student related registration and assessment periods within the University. We decided to assess two periods-: Peak and Off Peak. For e.g. the period 25/12/10 is an off peak period when school is on recess and little traffic over the network is expected. Since the student assessment system accounts for the largest use of the VMWare Vcenter system resources, our log auditor application demonstrates a relatively smaller log file utilization for system events as compared to the larger application events. During the period 29/01/11 to 04/02/11, a peak period, in student registration for the Spring term, when log traffic is at its highest. This log traffic is somewhat consistent for both the System and application log instances in both the peak and off peak periods. Notwithstanding our compression hash technique show that for both log type occurrences, performance overhead is cut by approximately 1/3 when using the compressed log provenance hash for auditing. This result is useful as it demonstrates our ability to improve the SAN disk life by the same margin, to support new log provenance entries. Independent work will discuss how synchronized log compression on the VM can impact capacity planning on the SAN disk as a set of use case scenarios.

12. CONCLUSION

In this paper the authors have initiated a study of enabling virtual machine log security using compressed hashed provenance techniques. The main contribution is a set of primary formal protocol definitions and case arguments for establishing correctness and authenticity for the preservation of log data collected in these logical domains. The work is further corroborated by an experimental University case study on same. From the literature this appears to be one of the first few papers that have done any evaluation for the work prescribed. Future work includes, looking at the history of data ownership on the VM and how one can provide suitable access control mechanisms for these VM audit log entries. Secondly, we seek to explore how one can use these preserved logs as a part of case evidence in cloud digital investigations.

REFERENCES

Secure hash standard. Federal Information Processing Standards Publication [FIPS PUB]. 180 (1), April 1995

R.Agrawal, R.Bayardo, C.Faloustos, J.Kierman, R.Rantzau, and R.Skrikant. Auditing Compliance with Hippocratic database. In VLDB 2004.

J.Annis, Y.Zhao, J.-S. Volcker, M. Wilde, S.Kent, and I. Foster. Applying chimera virtual data concepts to cluster finding in the sloan sky survey. In *Proceedings of the ACM/IEEE Conference on Supercomputing, 2002*

O.Benjelloun, A.Das Sarma, A. Halevy, and J. Widom. ULDBs: Databases with uncertainty and lineage. In VLDB, 2006.

D. Bhagwat, L.Chiticariu, and W.-C.Tan., and G.Vijayvargiya. An annotation management system for relational databases. In VLDB 2004.

U. Braun, A. Shinnar, and M. Seltzer. Securing Provenance. In USENIX, July 2008

P. Buneman, A. Chapman, J. Cheney. Provenance management in curated databases. In ACM SIGMOD, 2006.

P.Buneman, J.Cheney and S. Vansummeren. On the expressiveness of implicit provenance in query and update languages. In ICDT, 2007.

P.Buneman, S. Khanna, and W.-C. Tan. What and where. A characterization of data provenance. Lecture Notes in Computer Science, 2001

S.P.Callahan, J.Freire, E.Santos, C.E. Scheidegger, C.T. Silvand, and H.T.Vo. Vistrails. Visualization meets data management. In ACM SIGMOD 2006

A. Chapman, H.V. Jagadish, and P. Ramanan. Efficient provenance storage. In ACM SIGMOD, 2008

A.Chebotko, S.Chang, S.Lu, F. Fotouhi, and P.Yang. Scientific workflow provenance querying with security views. In WAIM, 2008.

A.Cirillo, R.Jagadeesan, C.Pitcher, and J.Riely. Tapido: Trust and Authorization viaProvenance and Integrity in Distributed Objects. Lecture Notes in Computer Science, Programming Languages and Systems Edition, 2008.

S.Davidson, S.Cohen-Boulakia, A.Eyal, B. Ludascher, T.McPhillips, S.Bowers, and J.Freire. Provenance in Scientific Workflow Systems. IEEE Data Engineering Bulletin, 32(4), 2007.

P.Devanbu, M.Gertz, A.Kwong, C.Martel, G. Nuckolls, and S. Stubblebine. Flexible authentication of XML documents. *Journal of Computer Security*, 12(6), 2004.

P.Devanbu, M.Gertz, C.Martel, and S. Stubblebine. Authentic third party data publication. In

Proceedings of the IFIP 11.3 Workshop on Database Security, 2000

I. Foster, J.Vockler, M.Eilde, and Y.Zhao, Chimera: A Virtual data system for representing and querying, and automating data derivation. In SSDBM, 2002.

J.Frew, D.Metzger, and P.Slaughter.Automatic Capture and reconstruction of computational provenance. Concurr. Comput: Pract. Exper., 20(5): 485-496, 2008

P.Groth, S.Miles, and W.Fang, S.Wong, K. – P. Zauner, and L.Moreau. Recording and using provenance in the protein compressibility experiment. In IEEE Symposium on High Performance Distributed Computing, 2005.

P.Groth, S.Miles, and L.Moreau. PReServ: Provenance recording for services. In *Proceedings of the UK OST e-Science second All Hands Meeting 2005(AHM'05), 2005*

R.Hasan, R.Sion, and M.Winslett. Introducing secure provenance: Problems and challenges. In International Workshop on Storage Security and Survivability, 2007.

R.Hasan, R.Sion, and M. Winslett. The case of the fake Picasso: Preventing history forgery with secure provenance. In FAST 2009.

I. Khan, R.Schroeter, and J.Hunter. Implementing a Secure Annotation Service. LectureNotes in Computer Science. Springer Berlin/Heidelberg, Provenance and Annotation of Data, edition 2006.

F.Li, M. Hadjieleftheriou, G.Kollins, and L. Reyzin. Dynamic authenticated index structures for outsourced databases. In ACM SIGMOD, 2006.

G.Miklau and D.Suciu. Managing integrity for data exchanged on the web. In WebDB, 2005.

K. Muniswamy-Reddy, D.Holland, U. Braun, and M. Seltzer. Provenance aware storage systems. In USENIX 2006.

M.Noar and K.Nissim. Certificate Revocation and Certification update. In USENIX, 1998.

T.Oinn, M.Greenwood, M. Addis, M.N Alpedimir, J.Ferris, K.Glover, C.Goble, A. Goderis, D.Hull, D.Marvin, P.Li, P.Lord, M.R. Pocock, M.Senger , R.Stevens,A.Wipat, and C.Wroe.Taverna. Lessons in creating a workflow environment for the life sciences. Research Articles. Concurr. Comput.:Pract. Exper, 18 (10), 2006

OpenProvenanceModel. http://twiki.ipaw..info/bin/view/Challenge/OPM, 2008

H.Pang, A.Jain, K.Ramamritham and K.Tan. Verifying Completeness of relational query results in data publishing. In ACM SIGMOD 2005

A. Rosenthal, L. Seligman, A. Chapman, and B. Blaustein. Scalable access controls for lineage. Workshop on the theory and Practice of Provenance, 2009

R.Snodgrass, S.Yao, and C.Collberg. Tamper detection in audit logs. In VLDB, 2004.

V.Tan, P.Groth, S.Miles, S.Jiang, S.Munroe, S.Tsasakou, and L. Moreau. Security Issues in a SOA Based Provenance System. Lecture Notes in Computer Science. Springer Berlin/Heidelberg, Provenance and Annotation of Data Edition, 2006.

W.T.Tsai, X.Wei, Y.Chen, R.Paul, J.-Y.Chung, and D.Zhang. Data Provenance in SOA: security, reliability, and integrity. *Journal of Service Oriented Computing and Applications*, 2007.

S. Thorpe, I. Ray, and T.Grandison. Using Schematic Mapping Techniques to synchronize virtual machine logs. To Appear 4th International Conference on Information Systems Secure (June 2011) - *Proceedings to be forwarded to the Lecture Notes in Computer Science-Springer Verlag edition*

S. Thorpe, I. Ray, and T.Grandison. Enforcing Data Quality Rules for Transformation mapping within a synchronize virtual machine log cloud. To Appear 4th International Conference on Information Systems Secure (June 2011) - *Proceedings to be forwarded to the Lecture Notes in Computer Science-Springer Verlag edition*.

P.Mell,T.Grance. NIST Definition of Cloud Computing Retrieved from: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf, September 2009.

Author Biographies

Sean Thorpe holds an M.S. and B.S. degrees in Information Security and Computer Science respectively from the University of Westminster, London, UK in November 2002 and from the University of the West Indies, Mona Campus Jamaica in November 2000. He joined the University of Technology (UTECH) as a Lecturer in Systems Security since January 2003.. Mr. Thorpe has worked extensively as a System Programmer Analyst and Oracle DBA, and now on study leave pursuing a PhD in Cloud Forensics at UTECH since 2010.

Indrajit Ray is an Associate Professor at Colorado State University since 2002. Prior he was an Assistant Professor at the University of Michigan Melbourne from 1997 to 2001. He earned his PhD from George Mason University, Virginia in summer 1997. He obtained his B.S. Computer Science Degree from Bengal Institute in India in 1984 and then his M.S. from Jadvapur University in 1991, also in India. His primary research interest is digital forensics, security policies, access controls and intrusion detection

Abbie Barbir holds a PhD from Arizona State University since 1991. Hecurrently heads the OASIS Security Study group 17 responsible for policyformations and standards for open access environments. He has significant industry experience within Nortel and the ITU. He is currently a consultant to Bank of America

Tyrone Grandison is a Research Staff Member at the Thomas J. Watson Research Center. He received a B.S. degree in Computer Studies (Computer Science and Economics) from the University of the West Indies in 1997, a M.S. degree in Software Engineering in 1998 and a Ph.D. degree in Computer Science from the Imperial College of Science, Technology & Medicine in London. He then joined IBM at the Almaden Research Center in California, where he worked on data privacy and security. In 2010, he joined the Global Healthcare Transformation team as Program Manager for Core Healthcare Services. Dr. Grandison is a Distinguished Engineer of the Association of Computing Machinery (ACM), a Senior Member of the Institute of Electrical and Electronics Engineers (IEEE), a Fellow of the British Computer Society (BCS), has been recognized by the National Society of Black Engineers (as Pioneer of the Year in 2009), the Black Engineer of the Year Award Board (as Modern Day Technology Leader in 2009 and Minority in Science Trailblazer in 2010) and has received the IEEE Technical Achievement Award in 2010 for "pioneering contributions to Secure and Private Data Management". He has authored over 70 technical papers and co-invented over 20 patents.