Use of Schema Associative Mapping for Synchronization of the Virtual Machine Audit Logs

Sean Thorpe[1], Indrajit Ray[2], and Tyrone Grandison[3]

[1] Faculty of Engineering and Computing, University of Technology
Kingston, Jamaica
thorpe.sean@gmail.com
[2] Colorado State University
Fort Collins, USA
indrajit@cs.colostate.edu
[3] IBM Almaden Research
Silicon Valley, USA
tyroneg@us.ibm.com

Abstract. Compute cloud interoperability across different domains represents a major challenge for the System administrator community. This work takes a look at the issues for enabling heterogeneous synchronization of virtual disk log attributes by use of an associative mapping technique. We explore this concern as a function of providing secure log auditing for the virtual machine (VM) cloud. Our contribution provides novel theoretical foundations that can be used to establish these synchronized log audit requirements supported by practical case study results.

Keywords: Log audit, Attribute, Synchronized, Virtual Machine, Associative.

1 Introduction

Enabling security within the compute cloud presents serious challenges which the VM system administrator has to face. These challenges range from third party trust, malicious insiders, and distributed storage just to name a few. Arguably the benefits of economies of scale for cloud deployment is matched by the security risk.

This paper articulates using a formal approach, the specific components required in designing an audit compute cloud. We achieve this by synchronizing the log file data between the actual physical disk and the virtual disk using associative mapping techniques of the adopted cloud components [7]. The rest of the paper is divided into four (4) sections. Section 2 looks at related work, Section 3 looks at an Attribute Log Graph context model, Section 4 looks at Experimental Results and Section 5 provides the conclusion and future work.

2 Related Work

Cloud computing arguably is seen by many as virtualization on steroids. The first deployment of virtual machines started as early as 2003, and by 2006 it's acceptance as a mainstream service oriented computing model became evident. It is well understood that virtual compute clouds provide tremendous economies of scale for users.

We look at the work defined by Arenas et.al. [6] and others [2] [4] [8] [9] to develop our own algorithms. We posit the use of schema mapping techniques to correlate log file designs for the synchronized Virtual log auditing functions. Schema mapping composition is a fundamental operation in data management and data exchange. The mapping composition problem has been extensively studied for a number of mapping languages most notably source to target tuple generating dependencies (s-t tgds). We adopt these constraints as apart of our own specification semantics for a Global Virtual Machine Attribute Policy Auditor (GVMAPA) which is currently ongoing work within our research group. An important class of s-t tgds are local as view (LAV) tgds. This class of mapping is prevalent in practical data integration and exchange systems. We use these ideas to develop heterogeneous source-target Virtual Machine log maps, given the rich and desirable structural properties that these mappings possess and we think it can offer to such abstract domains.

3 Audit Log Graph Context Model

The authors posit a synchronized log file based audit mechanism for the cloud environment. The following formalisms below represent the attribute parameters required for defining these types of digital log footprints:

Definition 1: Virtual log (VL) footprint contains the attributes: We adopt the formal definition from [7].

$$c_a = \{e_1^t, \ldots\ldots, e_n^t)$$

$$c_a \in C, e_x^i \in E, 1 \le i \le l, 1 \le x \le m$$

l represents the maximum number of active log context and m represents the maximum number of VM environments defined for any given active context. Each context environment must be sensitive to the identified data, machine, and time allocated to a specified context

$$e_x^i = \{t_x^i, \{m_1^{i^x}, \ldots, m_a^{i^x}\}, \{d_1^{i^x}, \ldots, d_b^{i^x}\}, \{cn_1^{i^x}, \ldots, cn_e^{i^x}\}\}$$

Where

$$m_a^{i^x} \in M, d_b^{i^x} \in D, cn_e^{i^x} \in CN, 1 \le a \le \infty, 1 \le b \le \infty, 1 \le e \le \infty.$$

M is the set of all machines, D is the set of all related data and CN is the set of all connected nodes. *M* is assumed to be a collection of physical machines only a is the maximum number of machines, *b* is the maximum number of data transactions and *e* is the maximum number of nodes within a data center. As the number of machines comprises virtual machines and connections may exist between machines of different data centers, the value of e may be significantly larger than n-1, i.e. the number of other data centers in this cloud instance The values for a and b are assumed to have no upper limit as the number of machines and storage units for a particular cloud will grow or shrink as needed The location of the xth data center of the ith cloud, l x is a latitude-longitude pair. A machine consists of at least one central processing unit indexed by a unique CPU_ID and bandwidth.

The storage component is either a physical storage unit or a virtual storage unit. Note that S = PS $\cup$VS where PS is the set of all physical storage and VS is the set of all virtual storage. The next connected data center contains the source node bandwidth and the next or target node. Both source and target nodes are either a data center, a physical machine address (i.e. MAC address and associated I.P. address) or a virtual machine (i.e. a logical address mapped to a binding I.P address).

Due to the fact that a few of the base constructs in our model are composite data types, we define a base function called in (Composite Object O, Sub-Element Type T) that returns a set of the elements of type T in object O. The implicit assumption is that all types in a composite object are unique. In our articulation of a cloud log instance, we declare that the data centers are connected as a reminder.

Definition 2: Each VM node is stored as apart of a cloud log graph G .

Algorithm 1. CloudLogGraph (Cloud ci)

```
1: Create an empty graph G
2: For each data center dx in ci
3: Create a group graph node (ggn) in G for dx
4: For each machine m contained in dx
5: Build a node m in ggn dx for machine m
6: For each dx in ci
7: For each next element e contained in dx
8: Find the associated node or ggn associated to e's source
9: Find the associated node or ggn associated to e's target
10: Create a link between source and target
11: Assign bandwidth to the weight of the edge
12: return G
```

Definition 3: The Virtual Log Data Set Graph GD is a set of labelled graphs, GD={G1,G2, . . . ,Gn}, and |M| denotes the number of log graphs in a log graph dataset, that is the size of the graph dataset. |E| denotes the number of edges of the log graph dataset.

Definition 4: The size of a Virtual Log Graph(G) is the number of edges in E(G).

Definition 5: A log sub-graph of a Virtual log graph G1 = {V, E} is a log graph G2 = {W, F}, where W $\subseteq$ V and F $\subseteq$ E. A sub-graph G2 of G1 is a proper log sub-graph of G1 if G2 $\not\in$ G1.

Given two labelled graphs G1 = $\{V,E,\Sigma_v,\Sigma_e, l\}$ and G2 =$\{V,E,\Sigma_v,\Sigma_e, l\}^1$, G2 is a log sub graph of G1 iff

1. $V^1 \subseteq V$.
2. $\forall u \in V^1 \quad ((l(u)) = (l^1(u)))$
3. $\forall u, v \in V, (u, v) \in E, (l(u,v) = l^1 f(u),f(v) \in E^1)$

Definition 6: A labeled log graph G2 is sub-graph isomorphic to labeled sub graph G1, denoted by $G2 \subseteq G1$.

Definition 7: Virtual Log graph assumes the use of the Frequency Graph mining algorithm. G is frequent iff $supG > \infty$. The frequent subgraph threshold is set to∞. where $0 \leq \infty \geq 1$ and a graph database GD, is used to find all frequent sub-graphs in GD. To obtain the frequency of a sub-graph G2, we should count the number of graphs which contains this sub graph. During the frequent subgraph search step, the expansion of sub-graphs may cause similar sub-graphs to grow in different ways. The use of canonical order techniques is adopted to evaluate log graph consistency among similar graphs.

In the next section we demonstrate how we have developed a software prototype called a virtual machine log auditor that synchronizes these log formalisms within the actual working environment.

4 Experimental Results

In order to evaluate the effectiveness of our approach we have implemented a prototype tool called the "Global Virtual Machine Log Auditor (GVLMA)." We have applied GVLMA within the context of a University wide deployment for monitoring it's private VCentre cloud. GVLMA as a software application maps the VMware essx3i host ran on a Windows 7 operating system. The coordinated mapping task involves making a read and copy of the source log on the VM kernel host. GVLMA polls the VCentre disk through a secure ftp session (SENDER shell script) to the VCentre production storage area network (SAN). At this point it should be noted that our SENDER script actually does a semantic mapping of the log source schema on the kernel. We also deploy a LOADER script which initiates an Oracle 11g stored procedure, and this allows for transformation mapping between the source and target schema. In this case the target schema is the log auditor's database.

We use JASPER reporting (i.e. Java's web-based GUI reports) to show the outcome of the semantic translation. In the test environment, we maintain GVLMA also running on Windows 7, using our own VCentre essx 3i test host. The production source SAN runs a 1 Terabyte disk storage whereas our test SAN runs a 100GB disk storage. The production VCentre cloud runs fifteen (15) virtual machine server instances connected to the local VM host.

The mapped log events are polled over different time points, to determine a suitable VM log footprint of consistent transactions ran by this host. In Figure 1 below we show a snapshot of these time points for the System Log Events as of 9/11/10. The table frequency shows that between the time interval 9:51 a.m. to 16:36 p.m. a consistent signature of failed disk starts. These results corroborate a chain of basic evidence to suggest that the security administrator should have swapped out these disks on the production SAN as well as to perform further forensic analysis on the same at that instant of noted failure.

The summary evaluation of these failed disk events also lead to obvious application instances errors on the SAN. For example failed buffer writes of the application logs, antivirus shutdown on the host etc.

LOG_ DATE	**EVENT**	**LOG-SOURCE**	**DESCRIPTION**	**OCCURRENCE**
			Most Frequent Errors	
VM Log Type: Error		Log type: Error	**Date loaded 20110128 /first vm-log dump**	
11/102010 9:51	15	Symmpi	The \Device\Scsi\symmpi1 is not ready for access yet.	36
11/10/201 0 9:51	11	Disk	The driver detected a controller error on \Device\Harddisk0.	36
11/10/201 0 9:50	15	Symmpi	The\ Device\Scsi\symmpi1 is not ready for access yet.	34
11/10/201 0 9:50	11	Disk	The driver detected a controller error on \Device\Harddisk0.	34
11/9/2010 16:36	11	Disk	The driver detected a controller error on \Device\Harddisk0.	26

Fig. 1. Shows the synchronized VM Host disk over the time period 9/11/10 between 9:00 a.m. to 16:36 p.m. for a sequence of time events and their Frequency of occurrence

What we cannot say from these results is why the VM system administrator took so long to detect the failures, but a reasonable assumption is that before we had done this automated prototype to perform the audit, the administrator appeared to have been doing manual entries to perform security validations on the VM kernel. And hence may have somehow missed these entries at that point in time. Notable to the production environment however at that time point, is the fact that there was no archival security analysis done on these system event logs. The commissioning of our automated log auditor since the 28/11/2011 is what the University currently uses to demonstrate these mapped log errors on the synchronized VM kernel. And hence we could argue that the administrator task could have been made easier, if this automation was done earlier.

It is also useful for the reader to understand that the automated synchronized logs audits are read from the VMware essx3i kernel \var\log directory. Hence for this reason we can generate a schema for both the device and application instances running within this virtualization stack.

The results captured in this study have been limited to the VMware cloud; however ongoing work explores further testing on the Citrix Xen-Kernel host, as well as VCentre newer infrastructure 4, to demonstrate further schematic mapping constraints by our log auditor. Also, we use static snapshot analysis of the log events to perform this system study.

5 Conclusion and Future Work

In this paper the authors have presented formal structural properties for auditing a synchronized VM cloud environment. We used schematic associative mapping formalisms to link these properties. We substantiated our formalisms by a preliminary experimental study. Further work explores associative mining rules that not only correlate log frequency on a synchronized event but also classifies the behavior of these events as forensic concern for this virtual machine log auditor.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB Conference, pp. 487–499 (1994)
2. Dehaspe, L., Toivonen, H.: Discovery of frequent, datalog patterns. Data Mining. Knowledge. Discovery 3(1), 7–36
3. Ordonez, C.: Models for association rules based on clustering and correlation. Intelligent Data Analysis 13(2), 337–358 (2009)
4. Yan, X., Cheng, H., Han, J., Yu, P.S.: Mining significant graph patterns by leap search. In: SIGMOD, Conference, pp. 433–444 (2008)
5. http://www.usenix.org/events/osdi99/full_papers/wang/wang_html/node8.html
6. Arenas, et.al.: Inverting Schema Mappings: Bridging the Gap between Theory and Practice. PVLDB 2(1), 1018–1029 (2009)
7. Grandison, T., Maximillen, E.M., Thorpe, S., Alba, A.: Towards a formal definition of Cloud Computing. In: Proceedings of IEEE Services (2010)
8. Mahavan, J., Harvey, A.: Composing Mappings among Data Sources. In: VLDB, pp. 251–262 (1996)
9. ten Cate, B., Kolaitis, P.: Structural Characterizations of Schema Mapping Languages. In: ICDT, pp. 63–72 (2009)