Detection of malicious activities on tracking to ensure security and transparency in cloud

Ashwini.S, Moorthi.K, Sakthi Vel

Abstract— Mechanism of tracking data flow in cloud computing is to provide transparency and increase security. The flow of data's along with its activities is monitored on tracking with a logging mechanism in order to increase trust and security. Event logging mechanism are involved which captures analysis and visualize data events in the cloud from the data point of view. Critical data-related cloud security problems such as malicious actions, attackers, and data leakages, data policy violations can be detected by analyzing the data provenance. Various data leakage threats and attacks are analyzed and alerted to data administrators and owners; thereby increasing the needs of trust and security for customers' data, with various provenance algorithms. Event logger Records file centric access and transfer information from within the kernel spaces of both virtual machines (VMs) and physical machines (PMs) in the Cloud, giving full transparency of the entire data landscape. Using the techniques such collecting system call from the kernel, maintaining and capturing provenance, and by maintain log details of the system ,both file centric and system centric tracking is done.

Index Terms— Virtual machine, Physical machine, security,event logger,provenance,xen hypervisor.

1 Introduction

Cloud computing is a way to provide customers with Information Technology services, with virtualization technologies in the background.In cloud computing there is increasing in data related security issues. Hence there is lack of trust in cloud computing. In order to increase trust in Cloud computing, we need to increase transparency and accountability of data in the Cloud for both enterprises and end-users.

Tracking mechanism is done on basis of two methods file centric and system centric [8]. In traditional one-system or local area network (LAN) environments, it is common to find user-space file monitoring tools or extensions of file systems. To be widely used for monitoring the single- or multiple-file activities within a single machine. Tools are also available for monitoring packets in networks example: Snort. With large scales and heavy usage of virtualization technologies in Cloud computing, such tools are insufficient to provide an over-arching view for monitoring files across both virtual machines (VMs) and physical machines (PMs). Moreover, these applications are usually housed within the user space, leaving them vulnerable to user space attacks.

As cloud computing and virtualization technologies become mainstream, the need to be able to track data has grown in importance. Having the ability to track data from its creation to its current state or its end state will enable the full transparency and accountability in cloud computing environments. The monitoring of virtual machines has many applications in areas such as security and systems management.The monitoring software detects the activities done on guest vm's (DomU), on basis of both file centric and system centric. Thus the events are been collected and stored. The storing of such events trough day by data activities forms a large collection of history of data in cloud thus forming a data provenance record. Thus forming a log details of various actions been performed under by number of virtual machines through number of days.

Intrusion detection systems rely on a wide variety of observable data to distinguish between legitimate and illegitimate activities one such observable sequence of system calls into the kernel of an operating system. We identified empirically the correct value by comparing the one in the guest and the ones within the hypervisor.

2 2 Related work

It address the problem on collecting provenance as first class cloud and constraints and challenges in doing so. Thus he proposed four properties that make provenance system truly useful, and implemented three protocols for maintaining provenance incurrent cloud stores. They also attempted to collect provenance using Xen hypervisor which precedes one step ahead in incorporating provenance to cloud computing.

The Provenance-Aware Storage System (PASS) is a pioneer system that treats provenance as first-class object, via automatic provenance collection and maintenance. Improved from operating at only one level of abstraction, the second PASS prototype presented in facilitates the integration of provenance across multiple levels of abstraction to better support users in analyzing their data and processes using an integrated view of the provenance.

The prototype consists of a central Data Provenance API (DPAPI) and even main components, namely libpass, interceptor, observer, er, distributor, Lasagna, and Waldo. The DPAPI allows transfer of provenance both among system components and across layers, which is exported to user level by library libpass. Provenance-aware applications can be developed by augmenting code to collect application specific provenance,

————————————————

- *Ashwini.S is currently pursuing masters degree program in Computer Science and engineering in Anna University,,Jansons Enginnering College Coimbatore, E-mail: ashwinicse1392@gmail.com.*
- Moorthi.K and Sakthi Vel *is faculty in Computer Science and engineering in Anna University,,Jansons Enginnering College Coimbatore.* *E-mail:* Sakthi Vel : Sakthi Vel

and issuing DPAPI calls to libpass. The user-level daemon Waldo reads provenance records from the log, stores them in a database, indexes them, and accesses the database for querying.

It has proposed the second PASS prototype and modified the Xen hypervisor to collect provenance from running guest kernels extended the second PASS prototype and modified the Xen hypervisor to collect provenance from running guest kernels. The approach places an interceptor on a DomU (i.e., a guest virtual machine) to intercept system calls in Xens syscall enter mechanism, and stores provenance records in a ring buffer. Concurrently, a user space daemon runs in a privileged domain (i.e., Dom0 or a special "provenance processing domain") to periodically consume records from the ring buffer, and perform other tasks related to er and Waldo similarly as in the second PASS prototype. Provenance-aware applications can be developed by augmenting code to collect application specific provenance, and issuing DPAPI calls to libpass.

It has over problem on collecting provenence via Xen hypervisor The Provenance Aware Storage Systems project (PASS) currently collects system-level provenance by intercepting system calls in the Linux kernel and storing the provenance in a stack able file system. While this approach is reasonably efficient, it suffers from two significant drawbacks: each new revision of the kernel requires reintegration of PASS changes, the stability of which must be continually tested.

the use of a stack able file system makes it difficult to collect provenance on root volumes, especially during early boot. Hence an approach to collecting system-level provenance from virtual guest machines running under the Xen hypervisor has been brought out. This approach alleviates the aforementioned difficulties and promotes adoption of provenance collection within cloud computing platforms.
This design starts to collect provenance immediately after the guest is powered on. This provides us with provenance from the root file system and from the early stages of the boot process, which is currently difficult to do with our in-kernel PASS approach.

It has proposed Cloud adoption and tried to increase transparency and accountability of data in the Cloud for both enterprises and end-users. Hence he introduced flogger to track the file activity. Flogger is a novel file-centric logger suitable for both private and public cloud environments. Flogger records file centric access and transfer information from within the kernel spaces of both virtual machines (VMs) and physical machines (PMs) in the Cloud, thus giving full transparency of the entire data landscape in the Cloud.

This mechanism provides initial experiments focusing on deploying Floggers to capture flogs across VMs and PMs for a Cloud, and also to demonstrate that we are able to join the information for VMs and their underlying host PMs. This gives a comprehensive overview of the file-centric accesses and transfers within a typical Cloud. Its drawback is that it only focuses on file basis actions. Flogger (Linux) – A Linux Loadable Kernel Module (LKM) running on VM which intercepts file and network operations and writes the events as VM flogs.Flogger (Windows) – A Windows Device Driver running on PM which intercepts file operations and writes the events as PM flogs. The activities of file with in the directory is monitored and made tracking the flow from guest to machine.

It has proposed Network intrusion detection systems (NIDS) are an important part of any network security architecture. They provide a layer of defense which monitors network traffic for predefined suspicious activity or patterns, and alert system administrators when potential hostile traffic is detected. Commercial NIDS have many differences, but Information Systems departments must face the commonalities that they share such as significant system footprint, complex deployment and high monetary cost. Snort was designed to address these issues.

It has proposed key barrier to wide spread uptake of cloud computing is lack of trust in cloud by potential customers. While preventive controls for security and privacy measures are actively being researched, there is still little focus on detective controls related to cloud accountability and audit ability .The complexity resulting from sheer amount of virtualization and data distribution carried out in current clouds has also revealed an urgent need for research in cloud accountability, has shift in focus of customer concerns from server health and utilization to integrity and safety of end-user‟s data. The key challenges in achieving a trusted cloud through the use of defective controls, and presents the Trust Cloud framework, which address accountability in cloud computing via technical and policy-based approaches.

3 Detection of Malicious Activities by Tracking Data to Ensure Security and Provide Transparency in Cloud

Cloud environment is usually supported with number of virtual machines which are used to imply the transparency and data flow activities under Xen hypervisor .The events and activities of each guest virtual machines are monitored and information are collected at host virtual machine to bring out transparency in data flow and data activities. This technique also help to identify if any intruder among these guest domain and send an alert message to the admin domain (Dom0) so as to detect and provide security to other guest domains(Dom U).

Event logger is a mechanism which captures events at file level and also captures data logs from kernel spaces of both virtual and host machines. This mechanism is usually proved by two loggers' available flogger and s2logger.under event logger there are various methodologies and prototype which is deployed and tested on the Linux operating system environment in an Open Stack-based cloud environment figure 1.

1) Information Capture - system events and block events are captured from each individual machine's kernel space.

2) Log Transfer and Storage - data logs are transmitted on the management network, maintaining isolation from the virtual machines. For virtual machines, data logs are transmit-

ted via the hypervisor's secure communication channels into host machines.

Hence, there is no network-dependent communication involved, increasing the stealthiest of the transmission of data logs from each virtual machine to their host physical machine (PM). As such, data logs accumulated from both the PMs themselves and all their VMs will have to be transmitted in batches to a data log database.

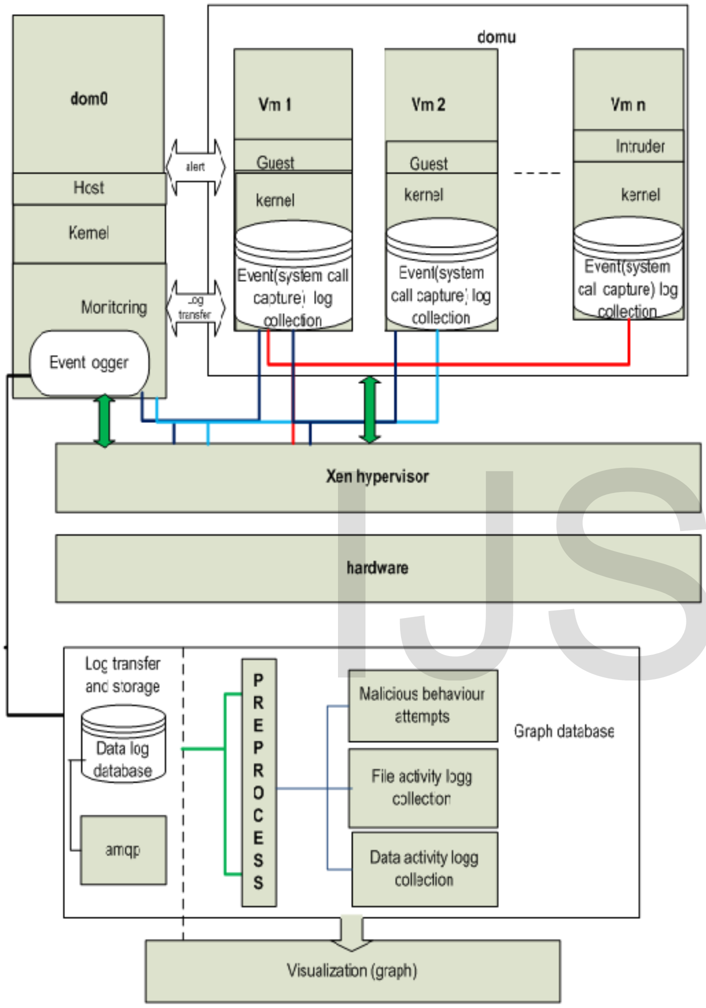

Fig 1. Architecture for data tracking mechanism via

Xen hypervisor

3.1 *Capture System Events*

Capturing of system events involve method types System event capturing is one of the two core task in the Information Capture phase. In solution, system call table events are captured by two possible methods:

1) Kernel system calls table function hooking method:

2) Linux security method.

Kernel system call hooking method is the first of two techniques to capture system events. When a Linux user-space application is running, the Linux operating system translates every user-space operation (e.g. opening a file) to equivalent kernel system call (e.g. SYS OPEN). By capturing Linux kernel system calls, we can observe, record and even modify the default Linux system call behavior. The Linux operating system stores the kernel system calls and their corresponding handlers inside the system call table. Each system call table entry is indexed by a unique identifier (e.g. SYS OPEN entry is indexed by NR OPEN identifier).

A second method to hook onto the necessary kernel syscalls is to use the Linux Security Module (LSM) API. This API is used by security-related modules in the kernel to enforce security policies on the kernel functions. Some examples of important security systems based on LSM are SELinux and apparmor. SELinux uses the filtering of critical. Kernel functions to implement a robust mandatory access control (MAC) security scheme by labeling all resources (at the inode level) and processes with the appropriate security class.

A second method to hook onto the necessary kernel syscalls is to use the Linux Security Module (LSM) API. This API is used by security-related modules in the kernel to enforce security policies on the kernel functions. Some examples of important security systems based on LSM are SELinux and apparmor. SELinux uses the filtering of critical. Kernel functions to implement a robust mandatory access control (MAC) security scheme by labeling all resources (at the inode level) and processes with the appropriate security class.

3.3 LINUX SECURITY MODULE (LSM)-BASED METHOD

A second method to hook onto the necessary kernel syscalls is to use the Linux Security Module (LSM) API. This API is used by security-related modules in the kernel to enforce security policies on the kernel functions. Some examples of important security systems based on LSM are SELinux and apparmor. SELinux uses the filtering of critical. Kernel functions to implement a robust mandatory access control (MAC) security scheme by labeling all resources (at the inode level) and processes with the appropriate security class.

For the purpose of tracking file provenance, rather than implementing a security policy, the LSM interface is used to keep a log of the entire file and network related system calls. The tracking module defines a custom function for selected hooking functions in the structure. This custom function receives the system call parameters whenever they are called from the user-space, and gives a return value to authorize the operation, while logging the call itself through the logging layer the LSM functions that are being targeted, and the information that is captured from each of the system call figure 2

.

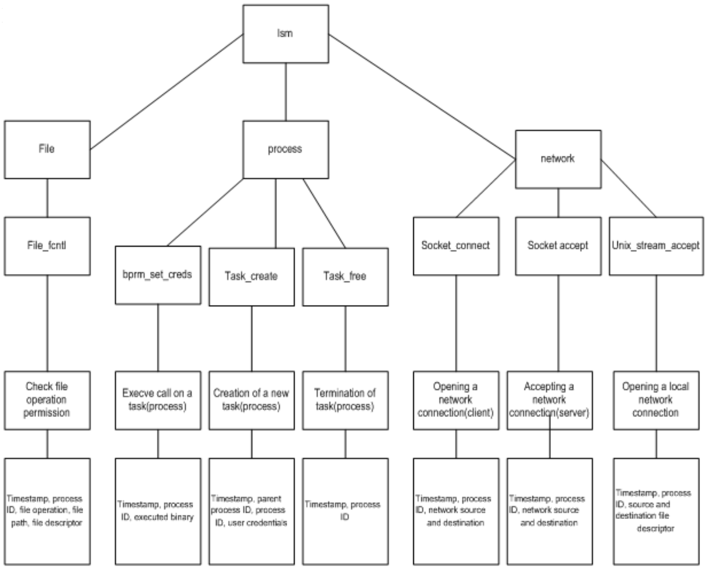
lsm
File
process
network
File_fcntl
bprm_set_creds
Task_create
Task_free
Socket_connect
Socket accept
Unix_stream_accept
Check file operation permission
Execve call on a task(process)
Creation of a new task(process)
Termination of task(process)
Opening a network connection(client)
Accepting a network connection(server)
Opening a local network connection
Timestamp, process ID, file operation, file path, file descriptor
Timestamp, process ID, executed binary
Timestamp, parent process ID, process ID, user credentials
Timestamp, process ID
Timestamp, process ID, network source and destination
Timestamp, process ID, network source and destination
Timestamp, process ID, source and destination file descriptor

Fig .4 Linux security module function

3.2 *Data Provenance For Collecting Data Activities*

Data provenance tracing algorithms designed for local data malicious action (leakage)problem aims to detect those file operations that can grant control to or leak content of confidential files, thus may result in data leakage implicitly. Specifically, these algorithms detect file copying, file renaming, and file movement in Linux. The Provenance Aware Storage Systems (PASS) currently collects system-level provenance by intercepting system calls in the Linux kernel and storing the Provenance in a stack able file system.

3.3Detecting file movement provenance algorithm

```
Step 1: for each process y in the OS does
Step 2: if y executed both Rename (Old File) and Rename(New File)
And Rename (Old File) was executed before Rename
(New File)
And the file name was unchanged before and after
Rename
And the file path was changed before and after Rename
Then
Step 3: return "A file has been moved!"
Step 4: end if
Step 5: end for
```

3.4Provenance Algorithm Detecting File Sending

```
Step 1: for each process tree t in the OS does
    Step 2: if two processes p1 and p2 belong to the process tree t
And p1 executed a connect to INET syscall issued
By Ssh
And p2 executed an open syscall issued by SCP then
Step 3: return "A file has been sent using SCP!"
Step 4: end if
Step 5: end for
```

3.5Algorithm Detecting File Receiving

```
Step 1: for each process tree t in the OS does
    Step 2: if two processes p1 and p2 belong to the process tree t
And p1 executed a connect from INET syscall issued
By sshd
And p2 executed an open new syscall issued by SCP
Then
Step3: return "A file has been received using SCP!"
Step 4: end if
Step 5: end for
```

3.6.5 Provenance algorithm at cloud activity phase

```
Step 1: procedure PROVCAPSULE (a; b)
Step 2: for a unique ID   key and original FORG Data file X and
Step 3: each file contains provenance metafile File Provenance and stored in Profile Y array
Step 4: provenance PSB   Loc Z
Step 5: while object ID 6= 0 do
Step 6: Read inputs in FORG
Step 7: Record for every object an in Data File X
Step 8: while a = Data File X do
Step 9: object ID = a: object ID and
Step 10: original file with comparison identity FORG=FORG+1
Step 11: queue object stored Data FileX+1 =FORG
Step 12: continue
Step 13: end while
Step 14: Record for every object c in Provenance File Y array
Step 15: If object ID: File Provenance =c. File Provenance
Step 16: object Defile provenance   object ID: FORG
Step 17: object ID   + 1
Step 18: end while
Step 19: Provenance FileY = (Loc [PSB], PSB: Execute, object ID)
Step 20: return object ID: File Provenance & length.DataFile X
Step 21: end procedure
```

Provenance-aware file system that stores provenance records along with data these records describe; provenance records are written internally to a log whose format ensures.onsistency between the provenance and data. The user-level daemon Waldo reads provenance records from the log, stores them in a database, indexes them, and accesses the database for querying.

The Provenance Aware Storage Systems project (PASS) currently collects system-level provenance by intercepting system calls in the Linux kernel and storing the provenance in a stackable file system.

CONCLUSION

A breakthrough in cloud data provenance achieved by proposing an end-to-end data-centric mechanism which detects data activities across the cloud, for example, data leakages, data stealing, intrusions at control points event Logger directly addresses data traceability concerns in cloud computing infrastructures. .Two different methods were used to capture system calls at file and system level. Granular data events recorded the kernel module are then transmitted to the physical host via an efficient local channel across the hypervisor. Analysis of data logs are performed based on graph representations. By traversing between critical resources or network endpoints, the end-to-end data leakage and data hacking path can be observed. Simplification techniques can be applied to prune non-critical paths in the data movement graph. Enforcement of data leakage and data stolen policies are performed at control points, by receiving data event notification from the respective hosts. With event Logger, cloud computing users are now empowered with the ability to trace their data within and across machines in clouds. A graph is set for the activities performed by guest with can detect the malicious activities.

REFERENCES

[1] www.usenix.org "event trace" 2009.

[2] "System calls" www.protecting xen hypercall. Year 2009.

[3] Kiran-Kumar Muniswamy-Reddy, Peter Macko, and Margo Seltzer. "Provenance for the cloud". File and storage technologies, FAST, Berkeley, CA, USA, 2010. USENIX Association.

[4] Peter Macko, Diana Maclean, Daniel Margo, Margo Seltzer, and Robin Smogor. "Layering in provenance systems". 2009 conference on USENIX Annual technical conference, USENIX, Berkeley.

[5] Peter Macko, Marc Chiarini, and Margo Seltzer." Collecting provenance via the Xen hypervisor." Workshop on the Theory and Practice of Provenance, 2011.

[6] Ryan K L Ko, Peter Jagadpramana, and Bu Sung Lee. "Flogger: A file-centric logger for monitoring file access and transfers within cloud computing environments." IEEE TrustCom 2011, Changsha, China, 2011. IEEE.

[7] Martin Roesch. Snort: Lightweight intrusion detection for networks. In Proceedings of the 13th Conference on Systems Administration 1999 USENIX Association.

[8] Ryan K L Ko, Peter Jagadpramana, Markus Kirchberg, Qianhui Liang, Miranda Mowbray, Siani Pearson, and Bu Sung Lee. "Trustcloud – a framework for accountability and trust in cloud computing." July 2011. IEEE Computer Society.

[9] Olive Qing Zhang, Markus Kirchberg, Ryan K L Ko, and Bu Sung Lee. "How to track your data: The case for cloud computing provenance." In Cloud Computing Technology and Science (Cloud Com), 2011 IEEE Third International .

[10] Kiran-Kumar Muniswamy-Reddy, Uri Braun, David A. Holland, Peter Macko, Diana Maclean, Daniel Margo, Margo Seltzer, and Robin Smogor. Layering in provenance systems. In Proceedings of the 2009..

[11] Olive Qing Zhang, Ryan K L Ko, Markus Kirchberg, Chun Hui Suen, Peter Jagadprsamana, and Bu Sung Lee. How to track your data: Rule-based data provenance tracing algorithms. In Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference.

[12] Ryan K L Ko, Markus Kirchberg, and Bu Sung Lee. From "systemcentric to data-centric logging-accountability, trust & security in cloud computing" IEEE, 2011..

[13] Kiran-Kumar Muniswamy-Reddy, David A. Holland, Uri Braun, and Margo Seltzer. "Provenance-aware storage systems". In Proceedings of the Annual Technical Conference,, Berkeley, CA, USA, 2006.

[14] Fredic beck and Oliver Festor,Theme com-System communication project "Syscall Interception in xen hypervisor" MADYNES Raport technique n*9999-November 2009.