LUARM – An Audit Engine for Insider Misuse Detection

G. Magklaras, S.M. Furnell and M. Papadaki

Centre for Security, Communications and Network Research, University of Plymouth, Plymouth, UK
e-mail: cscan@plymouth.ac.uk

Abstract

'Logging User Actions in Relational Mode' (LUARM) is an open source audit engine for Linux. It provides a near real-time snapshot of a number of user action data such as file access, program execution and network endpoint user activities, all organized in easily searchable relational tables. LUARM attempts to solve two fundamental problems of the insider IT misuse domain. The first concerns the lack of insider misuse case data repositories that could be used by post-case forensic examiners to aid an incident investigation. The second problem relates to how information security researchers can enhance their ability to specify accurately insider threats at system level. This paper presents LUARM's design perspectives and a 'post mortem' case study of an insider IT misuse incident. The results show that the prototype audit engine has a good potential to provide a valuable insight into the way insider IT misuse incidents manifest on IT systems and can be a valuable complement to forensic investigators of IT misuse incidents.

Keywords

Insiders, misuse, detection, auditing, logging, forensics

1. Introduction

The problem of insider IT misuse is a very real threat for the health of IT infrastructures encompassing both intentional activities (e.g. targeted information theft and accidental misuse (e.g. unintentional information leak). Numerous studies have tried to define an “insider” in the context of Information Security. A generic definition from Probst et al. (2009) is ”a person that has been legitimately empowered with the right to access, represent, or decide about one or more assets of the organization's structure”.

The most widely known insider misuse cases are usually about intellectual property theft. The arrest of Lan Lee and Yuefei Ge by FBI agents (Cha, 2008) is a classic case. The arrested men were engineers of NetLogic Microsystems (NLM) until July 2003. During the time of their employment, they were downloading trade sensitive documents from the NLM headquarters into their home computers. These documents contained detailed descriptions of the NLM microprocessor product line. Eventually, their ties to the Chinese government and military were discovered by investigators. However, both mass media case descriptions and relevant security surveys do not provide the tools or the methodology to systemically study and mitigate the problem. Insider IT misuse is a multi-faceted problem and one of the things insider misuse

researchers really need is a repository of more detailed case descriptions with a focus on the impact insider misuse actions have at computer system level (NSTISSAM). This is the area of Insider Threat Specification, the core concept behind the proposed logging engine which is examined in the next section.

2. Insider Threat Specification and modelling

Threat specifications follow the principles of intrusion specification, a concept which is not new in the information security world. Techniques to describe threats exist for an entire range of information security products, from anti-virus software to several intrusion detection/prevention systems (IDS/IPS) (Bace, 2000), where threats are specified by anomaly detection, pattern matching (also known as misuse detection) mechanisms or a heuristic-based combination of the two. Insider Threat Specification is the process of using a standardized vocabulary to describe in an abstract way how the aspects and behaviour of an insider relate to a security policy defined misuse scenario. Figure 1 shows the information flow of a typical IT misuse detection system. The security specialist translates the Security (and resulting monitoring policy) into a set of misuse scenario signatures, standard descriptions of IT misuse acts that describe the behaviour of a user at process execution, filesystem and network endpoint level (Magklaras et al, 2006). The misuse scenario signatures and collected audit data (Bace, 2000) from the IT infrastructure are fed into a misuse detection engine.

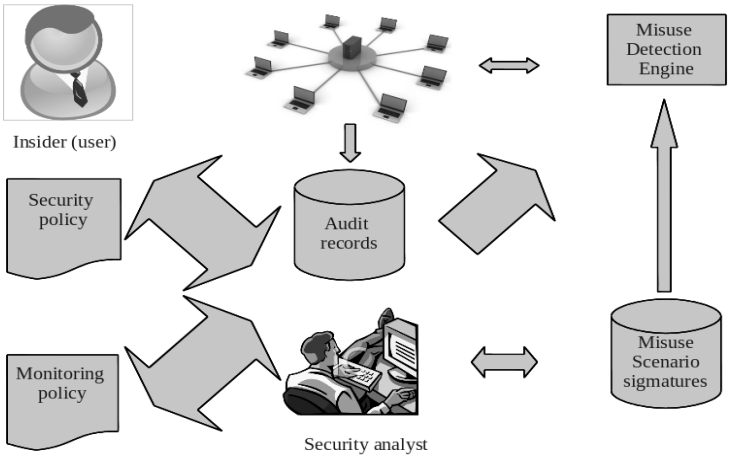

Figure 1: Information flow in an insider misuse detection system

Vital to insider threat specification is the structure and content of the audit record, at the center of Figure 1. If the audit record is incomplete, in terms of the type of information we need to log or unavailable, because the data are vanished due to bad system design or intentional data corruption, the specification of insider threats is useless. This is one of the primary objectives that LUARM tries to address by providing an evidence rich and reliable audit record format.

3. Insider misuse detection auditing requirements

Bace (Bace, 2000) discusses intrusion detection (and hence misuse detection) as an audit reduction problem. Audit reduction is the process of filtering the relevant information out of the audit records, in order to infer a partially or fully realized threat and excluding information that is irrelevant or redundant. The structure of an audit record is important for a misuse detection system. A good structure has well defined fields that can be easily parsed. Moreover, the structure of the audit record should easily facilitate relational type queries. It is necessary for the information to be applied on the disjunction (OR), conjunction (AND), and negation (NOT) operators, in order to increase the query versatility and speed of response.

A desired aspect of a suitable crafted audit record format for insider misuse detection is clear user accountability. This means that the audit record should be able to reliably and easily associate user entities to recorded actions. The wealth and replication of vital information in various types of audit records is a requirement for proper event correlation and step instance selection (Meier, 2004).

Another important issue of audit record engines is that of referencing time. In large IT infrastructures that span several networks and time zones, audited systems might report in different time formats. They can also experience 'clock skew', a difference in time recorded amongst computer systems due to computer clock hardware inaccuracies, especially when an NTP (Mills et al, 2010) server is not available to provide a reliable time source.

One of the most recent and commonly referenced works that concern the format of audit records is the Common Criteria for Information Technology Security Evaluation (Common Criteria Portal, 2009) standards. The Common Criteria (CC) effort does not fully address the previously mentioned audit record requirement omissions of its predecessor, the Orange Book (DOD 5200.28-std, 1985). However, some of its high level functional audit requirements are interesting. In particular, CC requirement 88 of section 8.2 states that: “At FAU_GEN.2 User identity association, the TSF shall associate auditable events to individual user identities.” In CC terminology TSF stands for Target of evaluation Security Functionality, meaning essentially the software and hardware under evaluation. In addition, CC mentions a set of requirements that concern various aspects of the audit record storage. Once again, the requirements are given in high-level terms, specifying that:

- unauthorized deletion and/or modification of audit records
- any other condition that could cause storage failure.

should be mitigated.

The next section discusses whether today's audit engines satisfy these requirements.

4. Existing audit record engines

Audit record engines have existed since the very early days of operating systems. However, not all of them fit the requirements of misuse detection engines, as discussed in the previous section.

The most common variety of audit record engines uses information that comes directly from the Operating System. Characteristic examples of this category of engines are Oracle's Basic Security Module (BSM) auditing system (Oracle Corporation, 2010) and its open source implementation OpenBSM (Trusted BSD Project portal, 2009), the psacct audit package (psacct utilities, 2003), as well as the syslogd (Gerhards, 2009) and WinSyslogd (Monitorware, 2010) applications.

After examining these engines, serious deficiencies can be located in terms of use for insider threat prediction. Firstly, many engines consolidate information from various different devices and operating system vendors, but they are far from describing sufficiently issues in an operating system agnostic way. In addition, process accounting tools might not cover sufficiently the variety of different system level information (file, process execution and network level). In fact, some of them might miss data as described in (HP Portal, 2003). A logging engine that cannot facilitate the description of both static and live forensic insider misuse system data at the network, process and filesystem layer could hinder a forensic examination of an IT misuse incident. Static digital forensic analysis is employed by most forensic tools and cannot portray accurately the non-quiescent (dynamic) state of the system under investigation. Information such as active network endpoints, running processes, user interaction data (number of open applications per user, exact commands), as well as the content of memory resident processes may not be recorded accurately on non-volatile media. (Hay et al, 2009) discuss the shortcomings of static digital forensics analysis in detail. In order to overcome the barriers of static analysis, Adelstein et al. (2006) discuss the virtues of non-quiescent or live analysis, which essentially gathers data while the system under-investigation is operational.

Several audit record systems do not report consistently the timing of audit record generation. For instance, many implementations of the syslog audit standard and psacct tools generate the audit record by entering the time stamp of the client system. If the client system does not have a reliable time source, this generates inaccurate information and could seriously hinder event correlation.

Finally, one of the most serious drawbacks of existing audit approaches is the inability to store the audit information in a form that can utilize relational queries. Section 3 discussed the reasoning behind this requirement. In one sense, some people might argue that this is an audit management feature rather than an audit log design issue. However, as section 3 discussed the advantages of using a relational schema to form audit queries in a structured log record, the author's view is that everything that increases the expressive power of an audit log query should be incorporated in the structure of the audit log, rather than being left as an 'add-on' feature.

5. The LUARM audit engine

LUARM is a prototype Open Source audit record engine (LUARM portal, 2010) that uses a Relational Database Management System (RDBMS) for the storage and organization of audit record data. The employment of an RDBMS is a core design choice for the LUARM engine. Beyond the relational type query support discussed in Section 3, an RDBMS offers the necessary data availability, integrity and scalability features, because most RDBMS tools are explicitly designed to organize and store large amounts of data, as dictated by many CC requirements. The Structured Query Language (SQL) facilitates instance selection and completion, as well as data correlation can be performed by using clauses such as 'FROM' and 'WHERE'.

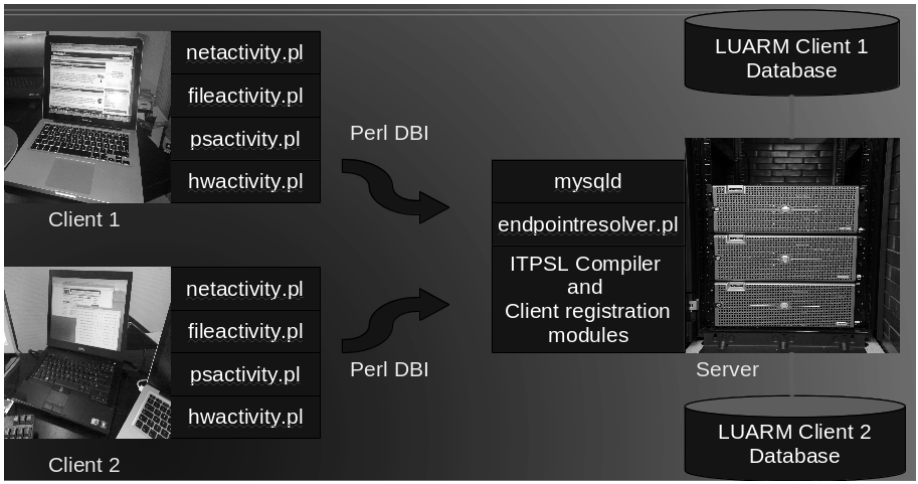

Figure 2: The LUARM architecture

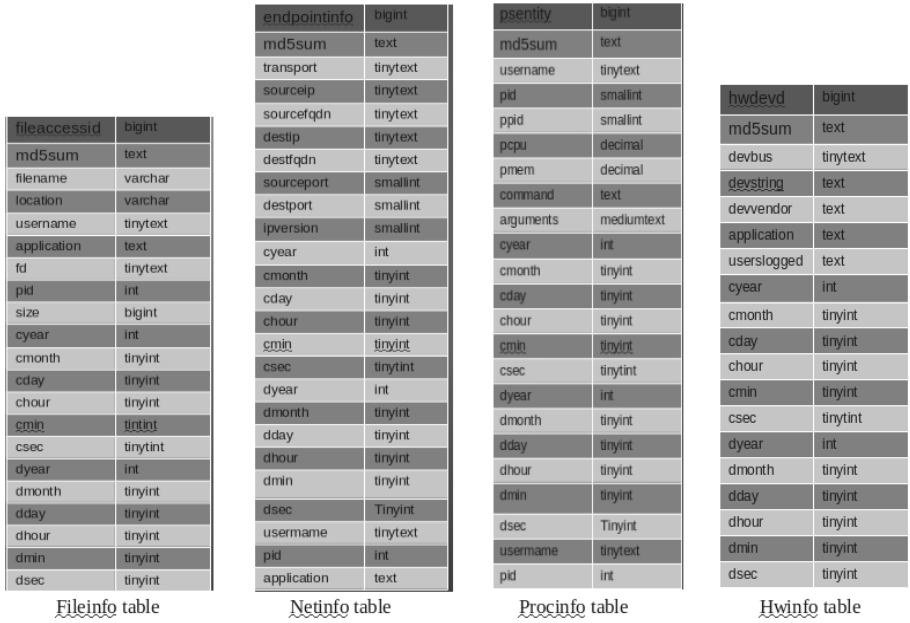

Figure 3: LUARM relational table structure

Figure 2 depicts the module client-server architecture of the LUARM audit engine. On the left of the figure, we can see a set of audited computer clients. Every client is running a unique instance of a set of monitoring scripts. Each of the client scripts audits a particular system level aspect of the operating system: 'netactivity.pl' audits the addition and creation of endpoints, 'fileactivity.pl' records various file operations, 'psactivity' provides process execution audit records and 'hwactivity.pl' keeps a log of hardware devices that are connected or disconnected from the system. The right hand side contains the centralized server part of the architecture where audit data are stored, maintained and queried in a MySQL (Oracle MySQL portal, 2010) based RDBMS (other RDBMS systems could be used as well). The Perl programming language is used to implement the modules and the communication between client and server is performed via a Perl DBI (CPAN-DBI, 2010) interface.

The client-server architecture avoids leaving the data in vulnerable clients. The central host MySQL server has its own authentication system responsible for controlling who has access to the audit data. By authenticating audit reviewers against the RDBMS authentication system, we de-couple the users being audited from the auditors, a desirable property that ensures that audited insiders cannot easily manipulate audit data. Furthermore, by assigning a separate database instance per audited client, we reduce the likelihood of compromising the data for all clients. If the database access credentials of one client are compromised, the damage is limited to the audit data for that client only.

Figure 3 displays the relational table format for the four main types of recorded audit data in LUARM: fileaccess, process execution, network endpoint and hardware device information. Temporal information is provided by event creation time stamps (cyear, cmonth, cday,chour,cmin,csec) and respective event destruction time stamps (dyear,dmonth,dday,dhour,dmin,dsec). The combination of the two types of timestamps can pinpoint exact time intervals for events in a consistent format for all recorded event types. In contrast, most audit systems may provide only event creation time references without hinting for the duration of an event.

The sampling of events is done at 100ms intervals and is adjustable by means of modifying certain variables on each monitoring module. At first, this might seem problematic as many attack steps can occur much faster than that amount of time. However, in an event sampling loop, one has to account for the time delay to update the database, which can vary from 10ms to 60-70 ms intervals on heavily loaded clients and servers. In addition, time resolution varies amongst operating systems. For these reasons, LUARM relies on the Perl Time::HiRes module (CPAN-HiRes, 2010) to bridge the gap between the different operating system timer implementations. A time granularity of 100 ms is also a good compromise between accuracy and scalability. The more granular the time resolution, the greater the computational load for both the client and the server LUARM parts.

Another important design decision that concerns the format of the audit table was to include common attributes amongst different event tables for the purposes of increasing the ability to correlate events and provide user entity accountability. For instance, fields such as 'username' (user entity), pid (numeric process ID of the program responsible for the event creation) and application (string that represents the name of the application that matches the pid) can be found in most of the event tables. This enables the audit reviewer to use SQL and relate events, so he can form queries of the type “Find the network endpoint created by program x of user y” in an easy manner.

The 'fileinfo' table stores file access related events. The filename specification consists of two parts. The 'filename' field which holds the filename with the file extension (i.e. data.txt) and the 'location' field which contains the absolute path of the file. The fact that the two are divided in separate fields makes it easier to search by location or by field name only, increasing the versatility of mining file data. In order to populate the data on this table, LUARM relies on the 'lsof' utility (Pogue et al, 2008). The utility is versatile and can record a variety of events including file and network endpoints in real time. It exists for an entire range of UNIX/Linux and MACOSX operating systems, covering a large spectrum of computing devices.

The 'netinfo' table logs the creation and destruction of network endpoints. In the context of LUARM, the term 'network endpoint' refers to the operating system data structures employed to facilitate network connectivity via the TCP/IP protocol suite. Network endpoint activity is considered as live forensic data. A series of table fields are used to record endpoint details ('sourceip', 'destip', 'sourceport' , 'destport' and 'transport' record source and destination IP addresses, source and destination port and

transport protocol respectively). The fields 'sourcefqdn' and 'destfqdn' hold the DNS (Mockapetris, 1987) resolved Fully Qualified Domain Name (FQDN) for the source and destination hosts.

The 'sourcefqdn' and 'destfqdn' fields are not populated by the client LUARM routines. In contrast, they are populated on the LUARM server side. Due to the criticality of correct DNS data for the audit records, the frequent DNS configuration errors (Barr, 1996), aspects of DNS operational security (Bauer, 2003) and client performance, the endpoint name resolution is left on the server side. This provides a greater control on DNS derived data and does not rely on vulnerable clients (malicious insiders or software vulnerabilities) for auditing network connections.

Process execution activity is recorded in the 'psinfo' table (Figure 3). This table records 'live' forensic data. The table includes both the proces ID ('pid') and parent process id ('ppid'), so that process execution flow can be traced back to the original process. In order to speed up process execution searches, the LUARM engine also separates the executed command ('command') from its arguments ('arguments'). One might like to search them separately in the process of mining process execution data. The 'ps' UNIX/Linux utility (Pogue et al, 2008) is used to collect process information. For all active processes (whose d* temporal fields are NULL), LUARM updates in near real time these two fields.

The 'hwinfo' table logs 'live' device connection and disconnection events. All events generated by devices that connect to the Peripheral Component Interconnect (PCI and PCI-Express) and Universal Serial (USB) buses. These two buses are commonly found on a large array of computing devices. For instance, an audit reviewer or forensics analyst might correlate file activity to a portable storage medium connection, as part of an intellectual property theft scenario. In that case, the 'hwinfo' table logs information in various fields that help identify the attached device ('devstring', 'devvendor'), the bus the device was connected to ('bus') and correlate the device attachment event against a number of users that are logged into the system at the time of the device attachment ('userslogged').

6. LUARM in action

Having a proposed structure and content for the various categories of audit events as described in the previous section, we can now issue sample SQL statements to illustrate how audit data mining is performed. Figure 4 displays sample queries that demonstrate the expressiveness of LUARM's audit record content and structure.

There are a few important observations to make about the example LUARM SQL queries. The first one concerns the embedding of system specific knowledge inside the statement. In essence, the third example of Figure 4 defines a step of an insider trying to transfer a sensitive file to a portable medium. One has to know the name of the sensitive file 'prototype.ppt' and also the fact that '/media' is used as a mount point for portable media for that host. Additional possible destination locations could be specified by means of OR operators. The use of the 'RLIKE' operator (RLIKE

RegExp, 2008), always in relation to the second and third examples of Figure 4. The operator implements a regular expression type of match. Apart from the conjunction operator (OR), regular expressions give the specification polymorphic properties (one specification string, many matching results), a desirable property for compact misuse detection language statements.

Find all accesses of the file 'prototype.ppt' by users 'toms' OR 'georgem' between 9:00 and 14:00 hours on 23/10/2009.

```
SELECT * FROM fileinfo WHERE filename='prototype.ppt' AND ((username='toms') OR
(username='georgem')) AND cyear='2009' AND cmonth='10' AND cday='23' AND chour >= '9' AND
chour <= '13' AND cmin >= '0' AND cmin >= '59';
```

Find all USB devices that were physically connected to the system when users 'toms' OR 'georgem' were logged on 23/10/2009.

```
SELECT * from hwinfo WHERE devbus='usb' AND ((userslogged RLIKE 'toms') OR (userslogged
RLIKE 'georgem')) AND cyear='2009' AND cmonth='10' AND cday='23' AND chour >= '9' AND chour
<= '13' AND cmin >= '0' AND cmin >= '59';
```

Find whether users 'georgem' or 'toma' have tried to move or copy a file called 'prototype.ppt' (irrespective of location) under the directory '/media' between 9:00 and 13:00 hours on the 23rd of October 2009.

```
select * FROM psinfo WHERE ((command='cp') OR (command='mv')) AND (arguments RLIKE
'prototype.ppt' AND arguments RLIKE '/media') AND ((username='georgem') OR (username='toms'))
AND cyear='2009' AND cmonth='10' AND cday='23' AND chour >= '9' AND chour <= '13' AND cmin >=
'0' AND cmin >= '59';
```

Figure 4: Using SQL to mine data in LUARM

LUARM was tested on a variety of simulated insider misuse scenarios. The scenarios were derived by real world LUARM captured data. However, permission to publish the original audit data was not obtained by the organizations in question. Thus, we had to reconstruct the misuse incidents by means of writing down a text based description of each incident and ask a team of users to re-enact it under a controlled IT infrastructure. The following paragraphs will present one of these incidents and demonstrate how the correlation versatility of the LUARM relational audit log structure can shed forensic light into the actions of a malicious insider. The scenario is provided below:

> 'Autobrake' Corp is a company designing car braking systems. Their engineering department is the most information sensitive work area. The braking system design process takes place in high performance Linux workstations, one for each design engineer. The engineers have normal user rights to the workstations. Superuser rights (root) is given only to the IT admin. The designs reside on the local hard drives of the workstations and the company's IT policy forbids any transfer of sensitive data to portable media. Autobrake's system administrator has requested a salary raise various times. This has been denied by management. The system administrator is lured by a competing company that asked him to deliver schematics of the new and revolutionary Autobrake's RGX9 SUV braking system in return for a large amount of money. Enjoying the trust of everyone and having full control of the engineering CAD workstations, the system administrator decides to take the offer of the competing company. He performs the intellectual property theft by following a well designed approach which is summarized below:

- He carefully chooses the user account of a mechanical engineer (username 'engineer3') that had some disputes over work issues with management. He aims to avoid detection by means of masquerading as the engineer in question.
- After successfully masquerading as the engineer in the IT system he uses a portable USB key to obtain the commercially sensitive RGX9 schematic, leaving only the traces of the engineer "actions".

Assuming that a third party auditor manages the audit process and monitors the logging (ensuring that the logging infrastructure works) and that all Engineering workstations are monitored by LUARM, we are now tasked to find the offender and clear the name of 'engineer3'. The reader should consult the LUARM relational table structure (Figure 3), in order to follow the SQL queries presented below.

The investigation begins from the most important file, that of RGX9, and the people that work on it. From the audit record of the workstations with name 'proteas', we utilize LUARM to find out who has been using the file:

mysql> select username,pid,cday,chour,cmin,location,filename from fileinfo where filename RLIKE 'RGX9' OR location RLIKE 'RGX9' \G

From the many hits we get from the data base, we focus our attention on the following ones:

**************************** 111. row ****************************
username: engineer3
pid : 8301
cday: 4
chour: 15
cmin: 30
location: /storage/users/engineer3/work/designs
filename:RGX9.jpg
...
**************************** 118. row ****************************
username: engineer3
pid: 28538
cday: 4
chour: 15
cmin: 32
location: /media/U3SAN03-12
filename: RGX9.jpg

The reason these file access patterns looked suspicious is that they were different than the normal pattern of accessing the file by the staff engineer. Normally, user 'engineer3' would access the file by means of certain design and image editing applications, under its usual directory (/storage/users/engineer3/work/designs). This time, however, things look a bit different, if one follows the association of file access

to process execution, in order to confirm which programs performed the file transaction. The following SQL queries achieve the desired association:

mysql>select username,pid,command,arguments,cyear,cday,chour,cmin from psinfo where username='engineer3' AND pid='8031' AND cyear='2011' AND cday='4' AND chour='15' AND cmin='30;

**************************** 1. row ****************************
username: engineer3
pid: 8031
command: /bin/cp
arguments: work/designs/RGX9.jpg /tmp/
cyear: 2011
cday: 4
chour: 15
cmin: 30**

mysql>select username,pid,command,arguments,cyear,cday,chour,cmin from psinfo where username='engineer3' AND pid='8031' AND cyear='2011' AND cday='4' AND chour='15' AND cmin='30;

**************************** 1. row ****************************
username: root
pid: 28538
command: mv
arguments: RGX9.jpg /media/U3SAN03-12
cyear: 2011
cday: 4
chour: 15
cmin: 32**

Essentially, the previous results verify that the file was first copied from the normal directory to /tmp and then was moved to the /mnt/usb. At this point, a little bit of system specific knowledge comes into light, as /mnt/usb is the usual mount point where Linux links portable storage media to the filesystem. Hence, the question to raise is whether a portal storage medium was connected to the workstation, prior to the 'mv' file transaction. The query result yields a positive answer:

mysql> select * from hwinfo where cyear='2011' AND cmonth='01' AND cday='04' AND chour='15'\G
**************************** 1. row ****************************
hwdevid: 71
md5sum: a16e7386f14de769a7a9491da2071f5b
cyear: 2010
cmonth: 12
cday: 4
chour: 15**

cmin: 30
csec: 28
devbus: USB
devstring: Cruzer Micro U3
devvendor: SanDisk Corp.
userslogged: engineer3,root
dyear: 2010
dmonth: 1
dday: 4
dhour: 15
dmin: 33
dsec: 38

This database hit seems to be in line with the actions of engineer3, as it indicates a device connection before the execution of the 'mv' command and a disconnection well after the mv command. Thus, everything seems to point out that 'engineer3' violated the company policy and transferred a sensitive file to a USB medium, against the company IT regulations. However, this had been categorically denied by the actual person. A good but non IT based alibi for the staff engineer was that he exited the building with his security card token around 14:50, returning back to his desk at 15:50, a wide gap for him. Clearly, something else was going on and the clue was the 'userslogged' field of the last LUARM result. This 'hwinfo' LUARM table field contains the usernames for accounts that are logged into the workstation at the time of the device connection. Apart from 'engineer3' we note the root account being active, which is clearly the only other choice that, under the circumstances, could have performed the mount procedure.

Based on the time stamp of the mv operation, a careful investigation of the root account actions reveals a key command execution, derived from the 'psinfo' table:

mysql> select * from psinfo where pid='27865' AND cyear='2011' AND cday='4' AND cmonth='1' AND chour='15' AND cmin >= '20' AND cmin <='33' \G
**************************** 1. row ****************************
psentity: 97654
md5sum: 7067284f2e1aefc430339ef091b4e41b
username: root
pid: 27865
ppid: 26407
pcpu: 0.0
pmem: 0.0
command: su
arguments: - engineer3
cyear: 2011
cmonth: 1
cday: 4
cmin: 28
chour: 15

csec: 36
dyear: 2011
dmonth: 1
dday: 4
dhour: 15
dmin: 28
dsec: 39

The 'su' command is used routinely by administrators to switch user credentials, in order to test environment settings and perform system tasks (Garfinkel et al, 1996). However, it can be easily used as a masquerading tool to covertly perform actions using the credentials of somebody else. A further investigation also found the USB key on the desk of the IT administrator with the RGX9.jpg file. The hwinfo table device identifier data ('devstring', 'devvendor') as well as the mount point identifier (/media/U3SAN03-12) from the psinfo commands contributed towards strengthening the final piece of the puzzle.

This case shows the versatility of the relational structure of the LUARM record that showed the way from simple file operation to related program execution and other events that can provide strong evidence and lead to the misuser. In addition, LUARM has also been used successfully to provide evidence about security incidents of external origin (Magklaras, 2011). Thus, it offers a valuable complement of existing logging mechanisms.

7. Conclusions

A very important tool to mitigate Insider IT misuse is an audit record which is specifically designed to address its various needs, as well as complement existing forensic tools when security specialists perform a post-mortem incident examination. LUARM is an audit engine that provides a detailed log of user actions at file, process execution and network endpoint level stored in a Relational Database Management System. Its file, process and network endpoint data provide a dynamic forensic view of the system, a useful complement to existing forensic tools that offer only static data in their majority. The relational storage layer increases the correlation versatility amongst the different types of audit data, as it is vital to be able to perform various associations during the investigation of an incident (process to file, process to network activity) and reliably relate actions to user entities.

The results are promising, showing a much better way to examine a system than looking at static text files which are difficult to parse and even more difficult to correlate. However, LUARM is a work in progress. It has its deficiencies and needs many improvements, in order to become a production real-world audit engine for insider misuse.

The first issue that was identified relates to the sampling frequency of user processes execution. After examining carefully the consistency of audit logs, it became evident that LUARM was losing process execution data. A fault was located at the process

execution monitoring module. Due to the way the sampling loop was written in that module, the effective sampling frequency could exceed by far the desired 100 millisecond sampling frequency. As a result, LUARM would miss processes that executed by various users in the system. The module was re-written using an entirely different process execution sampling philosophy. A Linux kernel technique called 'execve wrapping' was employed by adopting the Snoopy logger open source software (Snoopylogger portal, 2000). A modified 'execve wrapper' logger like 'Snoopy' logger provides a way to log the process execution and its arguments without relying on a sampling loop and is thus a more efficient interface to capture live process execution data. This solved the problem of losing process execution data due to a slow sampling rate and thus corrected an important deficiency of LUARM.

Addressing the issue of user privacy is not so straightforward. There is always a tension between insider IT misuse monitoring and privacy. LUARM needs to retain and collect data about a user's behavior, in order to help the analyst infer IT misuse. In direct contrast, privacy dictates the right of individuals to define whether somebody will collect data about their online actions and the extent or way the data can be used. The best compromise between these two opposing needs is to control the amount and type of logged data. This can be achieved by pseudo-anonymizing certain parts of the audit record, in order to protect certain aspects of the user privacy but still be able to infer IT misuse reliably. The term 'Privacy-Respecting Intrusion Detection' (Flegel, 2007), encompasses all the efforts of achieving a good compromise between the need to monitor and the need to respect user privacy.

The achievement of the LUARM prototype has been to demonstrate that structured evidence based logging for IT misuse is feasible. The authors welcome feedback and participation to the development of its code base. The prototype is not yet ready for production deployment, but it should be suitable for experimentation and has already proved its value on a number of insider IT misuse incidents.

8. Acknowledgements

The authors wish to thanks the University of Oslo IT engineers Harald Dahle, Jean Lorentzen and Melaku Tadesse for helping with the simulation of various misuse scenarios.

9. References

Adelstein F. (2006), “Live Forensics: Diagnosing Your System without Killing it First”, Comm. ACM, vol.49, no.2, 2006, pp. 63-66.

Bace R. (2000), “Intrusion Detection”, Macmillan Technical Publishing, Indianapolis, USA, ISBN: 1-578701856, pp. 38-39 discuss the terms 'misuse detection' and 'anomaly detection' in an intrusion specification context, pp. 47-66 discuss various audit record issues.

Barr D. (1996), “Common DNS Operational and Configuration Errors”, Internet Engineering Task Force (IETF) Request For Comment (RFC) 1537, February 1996.

Bauer M. (2003), "Building secure servers with Linux", O'Reilly & Associates, ISBN: 0-596-00217-3: Chapter 6, pp. 154-196.

Cha A.E. (2008), "Even spies embrace China's free market.", WashingtonPost,February15,2008,www.washingtonpost.com/wpdyn/content/article/2008/02/14/AR2008021403550.html (Accessed 03 March 2011)

Common Criterial Portal (2009), "The Common Criteria for Information Technology Security Evaluation", Version 3.1, Revision 3, July 2009. Part 2: Functional security components,www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R3.pdf (Accessed 03 March 2011)

CPAN-DBI (2010), "The Perl Database Interface (DBI) module" at the Comprehensive Perl Archive Network (CPAN),search.cpan.org/~timb/DBI-1.615/DBI.pm (Accessed 03 March 2011)

CPAN-HiRes (2010), "The Perl High Resolution Timer module" at the Comprehensive Perl Archive Network (CPAN),search.cpan.org/~jhi/Time-HiRes-1.9721/HiRes.pm (Accessed 03 March 2011)

DOD 5200.28-std (1985), "Department of Defense Trusted Computer System Evaluation Criteria", National Computer Security Center: Orange Book, DOD 5200.28-std, December 1985.

Flegel U. (2007), "Privacy-Respecting Intrusion Detection", Advances in Information Security, Springer, ISBN: 978-0-0387-34346-4 .

Furnell S. (2004), "Enemies within: the problem of insider attacks", Computer Fraud and Security, Volume 2004 Issue 7, pp. 6-11.

Garfinkel S, Spafford G. (1996), "Practical UNIX and Internet Security", Second Edition, O'Reilly and Associates, Sebastopol, CA, ISBN: 1-56592-148-1

Gerhards R. (2009), "The Syslog Protocol", Internet Engineering Task Force (IETF), Request for Comment (RFC) 5424, March 2009.

Hay B., Nance K., Bishop M. (2009), "Live Analysis Progress and Challenges", IEEE Security & Privacy, Volume 7, Number 2, pp. 30-37.

HP Portal (2003), "psacct process accounting misses some commands", HP IT ,forums11.itrc.hp.com/service/forums/questionanswer.doadmit=109447626+1286381845785+28353475&threadId=1413576 (Accessed 02 February 2011)

LUARM portal (2010),luarm.sourceforge.net/ (Accessed 03 March 2011)

Snoopylogger (2000), http://sourceforge.net/projects/snoopylogger/ (Accessed 04 May 2011)

Magklaras G., Furnell S., Brooke P. (2006), "Towards an Insider Threat Prediction Specification Language", Information Management & Computer Security, (2006) vol. 14, no. 4, pp. 361-381.

Magklaras G. (2011), "Catching an undesired guest in the penguin /tmp room", Epistolatory Blogspot, epistolatory.blogspot.com/2011/02/catching-undesired-guest-in-penguin-tmp.html (Accessed 03 March 2011)

Meier M. (2004), "A Model for the Semantics of Attack Signatures in Misuse Detection Systems", K. Zhang and Y. Zheng (Eds.): ISC 2004, Springer-Verlag Berlin,Heidelberg , LNCS 3225, pp. 158–169.

Mills D., Delaware U., Martin J., Burbank J., Kasch W. (2010), "Network Time Protocol Version 4: Protocol and Algorithms Specification", Internet Engineering Task Force (IETF) Request For Comment (RFC) 5905, June 2010.

Mockapetris P. (1987), "Domain Names – Implementation and Specification", Internet Engineering Task Force (IETF) RFC 1035, November 1987.

Monitorware (2009), www.winsyslog.com/en/product/ (Accessed 03 March 2011)

NSTISSAM (1999), "The Insider Threat To US Government Information Systems", U.S. National Security Telecommunications And Information Systems Security Committee, NSTISSAM INFOSEC /1-99.

Oracle Corporation (2010), "System Administration Guide:Security Services", Solaris 10 Operating System, Part No: 816–4557–19 , September 2010, pp. 559-672,dlc.sun.com/pdf/816-4557/816-4557.pdf, (Accessed 03 March 2011)

Oracle MySQL portal (2010), www.mysql.com (Accessed 03 March 2011)

Pogue C., Altheide C., Haverkos T. (2008), "Unix and Linux Forensic Analysis DVD Toolkit", Syngress, 2008, ISBN: 978-1-59749-269-0.

Probst C., Hunker J., Bishop M., Gollman D. (2009), "Countering Insider Threats", ENISA Quarterly Review Vol. 5, No. 2, June 2009, pp. 13-14.

Psacct utilities (2003), Utilities for process activity monitoring, linux.maruhn.com/sec/psacct.html (Accessed 03 March 2011)

Rivest R. (1992), "The MD5 Message-Digest algorithm", Internet Engineering Task Force (IETF) Request For Comment (RFC) 1321, April 1992.

RLIKE RegExp (2008), "String Regular Expression Operator", MySQL 5.1 Manual, Oracle Corporation,dev.mysql.com/doc/refman/5.1/en/regexp.html (Accessed 03 March /2011)

Trusted BSD Project portal (2009), "OpenBSM: Open Source Basic Security Module (BSM) Audit Implementation",www.trustedbsd.org/openbsm.html (Accessed 03 Match 2011)