COMPUTING SCIENCE

Logging Solutions to Mitigate Risks Associated With Threats in
Infrastructure as a Service Cloud

Winai Wongthai, Francisco Rocha and Aad van Moorsel

TECHNICAL REPORT SERIES

Logging Solutions to Mitigate Risks Associated With Threats in Infrastructure as a Service Cloud

W. Wongthai, F. Rocha and A. van Moorsel

Abstract

Cloud computing offers computational resources such as processing, networking, and storage to customers. However, the cloud also brings with it security concerns which affect both cloud consumers and providers. The Cloud Security Alliance (CSA) defines the security concerns as the seven main threats. This paper investigates how threat number one (malicious activities performed in consumers' virtual machines/VMs) can affect the security of both consumers and providers. It proposes logging solutions to mitigate risks associated with this threat. We systematically design and implement a prototype of the proposed logging solutions in an IaaS to record the history of customer VM's files. The proposed system can be modified in order to record VMs' process behaviour log files. These log files can assist in identifying malicious activities (spamming) performed in the VMs as an example of how the proposed solutions benefits the provider side. The proposed system can record the log files while having a smaller trusted computing base compared to previous work. Thus, the logging solutions in this paper can assist in mitigating risks associated with the CSA threats to benefit consumers and providers.

Bibliographical details

WONGTHAI, W., ROCHA, F., VAN MOORSEL, A.

Logging Solutions to Mitigate Risks Associated With Threats in Infrastructure as a Service Cloud
[By] W. Wongthai, F. Rocha, A. van Moorsel

Newcastle upon Tyne: Newcastle University: Computing Science, 2013.

(Newcastle University, Computing Science, Technical Report Series, No. CS-TR-1403)

Added entries

NEWCASTLE UNIVERSITY

Computing Science. Technical Report Series. CS-TR-1403

Abstract

Cloud computing offers computational resources such as processing, networking, and storage to customers. However, the cloud also brings with it security concerns which affect both cloud consumers and providers. The Cloud Security Alliance (CSA) defines the security concerns as the seven main threats. This paper investigates how threat number one (malicious activities performed in consumers' virtual machines/VMs) can affect the security of both consumers and providers. It proposes logging solutions to mitigate risks associated with this threat. We systematically design and implement a prototype of the proposed logging solutions in an IaaS to record the history of customer VM's files. The proposed system can be modified in order to record VMs' process behaviour log files. These log files can assist in identifying malicious activities (spamming) performed in the VMs as an example of how the proposed solutions benefits the provider side. The proposed system can record the log files while having a smaller trusted computing base compared to previous work. Thus, the logging solutions in this paper can assist in mitigating risks associated with the CSA threats to benefit consumers and providers.

About the authors

Winai received his BSc and MSc degree in Computing Science from Thailand. He received his MSc in Systems Design for Internet Applications from school of Computing Science at the University of Newcastle upon Tyne in 2009. Winai is currently a PhD student in the school of Computing Science at the University of Newcastle upon Tyne. His research topic is Systematic Support for Accountability in the Cloud which supervised by Professor Aad van Moorsel and Dr Feng Hao.

Francisco Rocha is a third-year PhD student with the School of Computing Science at Newcastle University. His research interests include systems security, software security, and security architectures. His latest work focuses on developing prevention techniques that guarantee memory confidentiality and integrity for cloud consumers in the presence of malicious insiders in cloud computing. Francisco also maintains a LinkedIn profile.

Aad van Moorsel is a Professor in Distributed Systems and Head of School at the School of Computing Science in Newcastle University. His group conducts research in security, privacy and trust. Almost all of the group's research contains elements of quantification, be it through system measurement, predictive modelling or on-line adaptation. Aad worked in industry from 1996 until 2003, first as a researcher at Bell Labs/Lucent Technologies in Murray Hill and then as a research manager at Hewlett-Packard Labs in Palo Alto, both in the United States. He got his PhD in computer science from Universiteit Twente in The Netherlands (1993) and has a Masters in mathematics from Universiteit Leiden, also in The Netherlands. After finishing his PhD he was a postdoc at the University of Illinois at Urbana-Champaign, Illinois, USA, for two years. Aad became the Head of the School of Computing Science in 2012.

Suggested keywords

CLOUD MONITORING
LOGGING SYSTEM
ACCOUNTABILITY

Logging Solutions to Mitigate Risks Associated With Threats in Infrastructure as a Service Cloud

Winai Wongthai*, Francisco Rocha*, Aad van Moorsel*

*School of Computing Science, Newcastle University, UK

Email: winai.wongthai, f.e.liberal-rocha, and aad.vanmoorsel@ncl.ac.uk

Abstract—Cloud computing offers computational resources such as processing, networking, and storage to customers. However, the cloud also brings with it security concerns which affect both cloud consumers and providers. The Cloud Security Alliance (CSA) define the security concerns as the seven main threats. This paper investigates how threat number one (malicious activities performed in consumers' virtual machines/VMs) can affect the security of both consumers and providers. It proposes logging solutions to mitigate risks associated with this threat. We systematically design and implement a prototype of the proposed logging solutions in an IaaS to record the history of customer VM's files. The proposed system can be modified in order to record VMs' process behaviour log files. These log files can assist in identifying malicious activities (spamming) performed in the VMs as an example of how the proposed solutions benefits the provider side. The proposed system can record the log files while having a smaller trusted computing base compared to previous work. Thus, the logging solutions in this paper can assist in mitigating risks associated with the CSA threats to benefit consumers and providers.

Keywords—cloud monitoring; logging system; accountability;

I. INTRODUCTION

The cloud is attractive to many organizations because of its flexibility and benefit of reducing IT costs [1]. It can potentially transform the IT industry in a wide variety of application areas and is the future of computing as argued by many people [2]. [3], [4] point out that Infrastructure as a service (IaaS) provides a base on which to build Platform as a Service (PaaS) or Software as a Service (SaaS) offerings. It is increasingly used in many areas (e.g., in medical experiments [5], [6], [7]). However, trust of consumers in a cloud is one of the main barriers for its continued proliferation. Customers want to know how the cloud operate their data (e.g., who has access to it, and when it was accessed) as argued by [8], [9], [10], [11]). The Cloud Security Alliance (CSA) has published top threats to cloud computing report [12] that relates to the security issues. This document is used in this paper to provide the basic threats which this paper aims to mitigate risks associated with these threats.

Research usually discusses the security in the cloud from customer perspectives. However, the CSA publish Top Threats Cloud Computing Survey 2012 [13] which includes both cloud consumers (53%) and providers (30%) as the respondents. The survey is a good example that research should consider the threats to cloud from both customer and provider perspectives. Thus, our study discusses mitigating the CSA threats for both parties. Many researchers [2], [1], [14], [9], [8] argue that a logging system is a significant aspect in any accountability

solutions to mitigate risks associated with the threats in the cloud. A logging system composes of logging processes and log files [8]. A logging process focuses on logging-related tasks, and log files store contents produced by these processes. However, works that propose log files for accountability ([2], [1], [14], [9], [8]) do not clearly focus on the security of both customers and providers. Additionally, they do not clearly discuss how the log files' contents could be, and how the contents can mitigate the risks associated with threats to benefit both customers and providers in detail.

This paper concerns the importance of customers' critical files, which is files in customers' virtual machines/VMs disks, and is valuable assets for their businesses such as databases files, full detail in Section II-C. It also aims to provide logging solutions to benefit both customers and providers. In our previous work [8], we already summarized all seven threats from CSA report [12]. Thus, this paper focuses on and describes only threat 1 (abuse and nefarious use of cloud computing), as an example of the cloud issues we tend to provide the logging solutions to deal with. Forms of this threat can be malicious customers use their VMs to attack other customers' VM or attack the provider's hosting system/dom0 (which hosts all customer VMs), or spam activities in the VMs.

There are two types of log files in the cloud as discussed in [10], [11]: *file-centric logs* and *system-centric logs*. The first is tracing files from the time they are created to the time they are deleted as discussed in [9], [10]. The later is the hardware layer logs such as memory use, disk storage, temperature, voltage, etc, as argued by [15]. It also can be event logs, user account activity logs, processor usage, etc, as argued by [11]. However, we focus on the file-centric logs. This is because current research of logging in the cloud only focuses on the system-centric logs [11], which is usually disclosed to consumers as argued by [16]. Although [11], [17] consider file-centric logs in their work which can benefit customers, they do not make clear how the logs benefit providers. Thus, this paper proposes solution regarding the file-centric logs and to benefit both customer and provider sides.

Summary of Contributions: First, this paper provides an in-depth analysis of how CSA threat 1 affects both customer and provider security concerns simultaneously. The analysis differs from previous work (e.g., [12], [8]), which concerning security only from either customer or provider perspectives. The value of the analysis is to provide a basis for what contents that logging solutions need to collect as evidence to deal with the threat. To benefit both customers and providers, the knowledge of the log file contents facilitate provision of the appropriate logging solutions to deal with the right problems.

Thus, this paper then proposes appropriate logging solutions to produce the file-centric logs (above), which assist in mitigating risks associated with the threat to benefit both sides.

Second, regarding the file-centric logs, and to benefit both customer and provider sides, this paper designs, and implements a prototype of the proposed logging solutions. It then discusses how the result from the prototype implementation can be used to form log files to be used as evidence to mitigate risks associated with threat 1. Third, the proposed system can be an alternative approach to collect file-centric logs as evidence to enhance accountability in IaaS by customer's VM memory introspection approach from the provider's hosting system. This approach yields smaller trusted computing base compared to placing interceptors in the VMs as in previous work [11] [17]. Work by [18], [19] can yield the same TCB as our proposed system. However, they are not designed to achieve history of critical files (e.g., information of who accesses these files, full detail in Section IV-A1) and the process behaviour log files (e.g., information of which file this process reading, Section IV-B1) as demonstrated in this paper.

The design of the architecture is based on a generic logging template (our previous work [8]) which can be used to perform a systematic analysis of logging system security. [8] also achieves recording VM's malicious process behaviours. However, it does not concern the history of critical files as done by this paper. We also use the template to clarify the layout of significant related components in IaaS such as the provider's hosting system/dom0, a VM, logging processes, log files, and the critical files. The template facilitates to systematically design and implement the proposed logging system, and to compare the proposed system to previous work in term of TCB.

We use Xen to replicate IaaS cloud architecture for the prototype implementation of the proposed logging system. It is currently a virtualization layer of many cloud providers (e.g., Amazon Elastic Compute Cloud or EC2 [20]) as argued by [17], [21]. The first contribution is simultaneous analysis of how threat 1 affects both customer and provider, and the proposed logging solutions. For the analysis, we investigate how customer VMs can attack other customer VMs, and attack dom0 in detail. We observe exactly literature underestimates the risks to the provider. Thus, we carefully analyse vulnerabilities of the provider's dom0 which may lead to compromising dom0 itself, then, all the customer VMs hosted by this dom0. For the proposed solutions, we then indicate how important of customers' critical files in VMs, and discuss the concepts of history of critical files, and process behaviour log files. Then, we discuss the content of both types of log files, and discuss how these log files can assist in mitigating risks associated with threat 1 to benefit both customer and provider sides.

To achieve the second contribution which is the design, implementation, and discussion of a prototype of the proposed solutions, we describe the proposed logging system architecture to obtain the history of critical files based on the template. Then we explain the prototype implementation, and discuss how the result from the implementation can mitigate the risks associated with threat 1. To achieve the last contribution which is the proposed system can be an alternative technique that yields smaller trusted computing base compared to previous work, based on the related components of logging systems from the template, we discuss how the proposed system can

Fig. 1. The IaaS architecture, from [8]

Fig. 2. The overall view of a generic logging template from [8], its IaaS components (white boxes) and logging components (shaded colour, logging process: P1-4, and log files: F1-4)).

achieve the log files we need while yielding smaller TCB compared to previous work [11], [17], [18], [19].

In this paper, Section II provides the background. Section III discusses how threat 1 affects both customer and provider. The proposed solutions and their prototype implementation are discussed in Section IV, V respectively. The result and how to obtain it are in Section VI. Section VII discusses how the result assist in mitigating the risks associated with threat 1, and compares our proposed system with previous work. Section VIII is the conclusions.

II. BACKGROUND

A. IaaS Architecture

This IaaS architecture is based on the Xen system. The provider side can be an organization that offers VMs to the customer side. The customer side can rent the VMs and remotely access them via the Internet. In Figure 1, hw is short for hardware. It is a machine that works as a host of a hypervisor and all guest operating systems (OSes). A hypervisor is software that enables the machine to run more than one guest OSes in parallel. Dom0 is a privileged domain guest OS that is launched by the hypervisor during system boot. It directly accesses the hw and manages domUs. A domU is an unprivileged domain guest OS that runs on top of the hypervisor. It is a VM and an example of an IaaS cloud product that the providers offer to customers to purchase.

B. A Generic Logging Template for IaaS Cloud

Our previous work [8] provide this template. We use it throughout this paper to facilitate in understating the layout of logging system components in IaaS. In the template (Figure 2),

IaaS components include hw0, hypervisor, hwU, dom0, domU, app0, appU, disk0, diskU, mem0, and memU. The first four components were already discussed in Section II-A. Hw0 is the same as hw in Figure 1, 0 indicates that it is managed and owned by a provider. AppU and app0 are applications that runs inside domU and dom0 respectively. Disk0 is a physical disk of the hw0, and diskU is a virtual disk of a domU. Mem0 is a main memory of the hw0, and memU is a virtual main memory of domU. P1-P4 are logging processes that perform logging-related tasks. F1-F4 are log files that are used for storing contents produced by the logging processes.

C. Customers' Critical Files in DomUs

This paper defines **critical files as files in diskUs that are owned by customers (domU's owners). These files can be any file type, such as text, executable, or database files. They are the customers' asset and valuable for their businesses. Thus, the customers do not want anyone to access these files apart from themselves and their authenticated users, and do not want loss or leakage of the files.** It would be very serious for any company if its business adversaries can access its critical files (e.g., business database files) [22]. Figure 3 shows the location of a critical file (called f) in diskU within the template. Critical files can be created inside domUs, or may be uploaded via the Internet by customers from their local machines to diskUs in the cloud.

For EC2, customers upload files to domUs to run their systems; for example, in medical experiments [5], [6], [7]. In [5], in order to more quickly find new drugs to heal new diseases, a computer intensive scientific experiment can be conducted in EC2. After EC2 domUs launched, all essential software, the input files, and executable C program files are transferred to the diskUs. Thus, all these files in diskUs are critical from the point of view of the owner of these domUs. Figure 3 shows the location of a critical file (called f) in diskU. Thus, one can clarify all components (logging components, IaaS components, and this critical file) in one view. This view assists us to analyse and find solutions to mitigate risks associated with CSA threats regarding both providers and customers. In Figure 3, the dot-arrow line virtually shows that Alice

Fig. 3. Illustration of a location of a critical file f in diskU

(the domU owner) accesses her critical file f using the appU. However, forms of threat 1 (such as customer VMs attacker others VMs, fully discussed in Section III) may allow attackers to control this appU, and use it to access the file maliciously.

III. THREAT 1 AFFECTS A CUSTOMER AND PROVIDER

Some forms of attacks enclosed in threat 1 (e.g., criminals use domUs to attack other domUs or dom0) can be critical. This is because they can eventually cause CSA threat 5 (data loss or leakage which may occur in domUs). This threat has been ranked as top in order of severity by the recent CSA reports [13], [22]. CSA state that the severity of this threat can be a devastating impact on the damage to one's business brand and reputations, or loss of core intellectual property [12]. Thus, we provide an in-depth analysis of the impact threat 1 can have on both customers and providers simultaneously.

A. Effects on A Customer due to Threat 1

1) *DomU attacks domU*: [23] demonstrate how to use a domU to extract the private keys of another domU in an Xen-Cloud environment (e.g., EC2). CSA also argue that domUs can host malicious software that has proven especially effective in compromising critical private resources in cloud environments [24]. Although they did not mention that the critical private resource is belong to whom, it may have belonged to customers. Especially, [25] also argue that a business competitor of a victim domU can use a malicious domU to attack the victim domU. Then, the competitor may be able to read private data or compromise the victim domU. Thus, domU attacks domU can cause a serious effect on cloud customers such as malicious access to, loss or leakage of (threat 5) the customers' critical files (e.g., database files).

2) *DomU attacks dom0 and uses this dom0 to attack other domUs*: Virtualization vulnerabilities in Windows 2008 (the hypervisor) allow domU running under the hypervisor to crash the Windows 2008 host (dom0) [26]. Thus, a domU can control dom0 and exploit the other domUs hosted on the same physical machine [27]. After dom0 is compromised, attackers can get control on the entire domUs [28]. Thus, they may obtain root accounts of these domUs, log into them, access the domUs' critical files (see Figure 3, and as discussed in Section II-C above). If these files are the customers' business databases, this can be a very serious incident. [21] argue that as dom0 can transparently read and write the memory content of the domU using the management interface; thus, if dom0 is compromised by attackers, they may use this interface to steal the valuable information from any domU. It is also argued by [15] that dom0 can access all data in diskUs. This can be a serious security concern from the point of view of the customers.

B. Effects on Provider due to Threat 1

1) *Criminals can use domUs to attack provider dom0*: Again, it can be crucial when dom0 is compromised because then all domUs could be at risks [4]. [26] states that domUs can attack the dom0 that hosts them. [29] state that this is because of a difficulty in clarifying the borders between a dom0 and domUs in the same physical machine with virtualization infrastructure. Thus, these unclear borders can be one of the attack channels. Another channel can be vulnerabilities in dom0. An example can be holes in the management consoles of dom0 that allow attackers to gain the root privileged in this dom0, as argued by [21]. Moreover, after dom0 is compromised, it can be used by attackers to monitor domUs, eavesdrop of communications between domU and dom0, take control of all domUs, and inject malware into domU images [30].

2) *A number of criminals in providers' cloud infrastructure can affect providers' business reputation*: There are many forms of attacks enclosed in threat 1 such as all the incidents caused by threat 1 discussed in Section III-A. It seems that these incidents affect only customer security concerns. However, these incidents can also affect provider security concerns; for example, allowing attackers to control dom0, and use it to compromise all domUs. Then, attackers (especially the competitors of the victim domUs) may access, lose, or leak customers' critical files (threat 5). Others forms of attacks enclosed in threat 1 also can be domUs that host spamming activities, or downloads for illegal software [12]. Thus, if the customers know that a lot of criminals (or all mentioned forms of attacks enclosed in threat 1) are inside the provider's cloud infrastructure, this can impact on the providers. The impacts can be losing the provider's business reputations (which can be important for customers when deciding to buy cloud products [31]), or these attacks can be an indicator of vulnerabilities in the provider infrastructure. Then, the customers may not want to buy or rent the product from this provider.

IV. PROPOSED LOGGING SOLUTIONS TO MITIGATE RISKS ASSOCIATED WITH THREAT 1 TO BENEFIT BOTH SIDES

A. History of Critical Files to Mitigate Risks Associated with Threat 1 for Both Sides

1) *What is the history of critical files*: As discussed in the compromising of both domUs or dom0 in Section III, either domUs or dom0 compromising may have the result of undesired access to, or loss or leakage of, customers' critical files. Thus, we propose to have a history of each of these critical files. This paper applies work by PASSXen [17] and HP TrustCloud [9] to form the definition of the history of a critical file. PASSXen is a system that can collect the information on the creation, access, and destruction of a file in the domU. TrustCloud is a framework to deal with the lack of trust in the cloud. It has file-centric information on domUs that is obtained by tracing domUs' data and files since they were created until deleted. Thus, **the history of a critical file in this paper is the information on the file since it was created, until permanently deleted. Precisely, it records of three periods of a critical file's life time: created, accessed, and destroyed**. This paper discusses only some information of the periods accessed as discussed below.

If a critical file *f* is *s.txt* (in Figure 3), the content of history of *s.txt* (as a log file) can be Table I. In the table, *f_nm* is the name of critical file (e.g., *s.txt*), *p_id* (e.g., 4624) and *p_nm* (e.g., read) is the id and name of the process that accesses this file respectively, and *p_ownId* (e.g., 1002, Alice Id) is the id of the owner of this process. Sections V, and VI discuss how to obtain this information. The content of the table can be more complex to provide more precise evidence, which will be discussed in the discussion, Section VII, Discussion.

<i>f_nm</i>	<i>p_id</i>	<i>p_nm</i>	<i>p_ownId</i>
<i>s.txt</i>	4624	read	1002(alice)
<i>s.txt</i>	4800	read	1003(bob)

TABLE I. THE CONTENT OF THE HISTORY OF CRITICAL FILE F (S.TXT), ADAPTED FROM [11]

In the case of some incidents happening in domUs that have negative effects on customers or providers, the history of

critical files can be used as evidence to clarify what happened with these domUs. This evidence can be a clue to discover what is going on inside the domUs that contain the critical files. Section IV-A2, IV-A3, VII-A, VII-B discuss how to discover the causes of the incidents. Consequently, the evidence should mitigate risks associated with threat 1 (e.g., criminal domUs). As a result, this should mitigate causes of negative impact on both customer and provider companies such as brand damage, as discussed in Section III-A, III-B.

2) *The history of critical files to mitigate risks associated with threat to benefit the customers*: It would be useful if we had a history of each of domU's critical files to assist in indicating, for example who has access to these files, and which appU accesses them. The history information can enhance accountability in the cloud and customers' confidence. For example for threat 1, when Alice domU is compromised by attackers, then they may control appU to access her critical file *f* (*s.txt*), as shown in Figure 3 (discussed in Section III-A). The history of *f* (the information such as which appU accesses *s.txt*, when, and by whom this appU belonged to) can be used to clarify this undesired malicious incident by the attackers. History of critical files could assist in analysing attacker behaviours inside domUs. To analyse attacker behaviours, Alice can check row 2 column 4 in Table I, and may discover that someone else (Bob) accesses her critical file *s.txt*. The content of the table can be more complex to provide more precise evidence, which will be discussed in Section VII.

3) *How the history of critical files helps providers to deal with dom0 compromising*: In Table I, when one discovers that Bob has accessed Alice's critical file *s.txt*, this can be a trigger for the providers to be aware that their dom0 may be compromised. To identify malicious Bob accurately, they can conduct further investigation, for example, by pinpointing Bob's Id (*p_ownId*, column 4), the appU name (*p_nm*, column 1) he used to access *s.txt*, or *s.txt* file name. Then, they can gather more necessary evidence. For example, this can be achieved by recording Bob's appU behaviours, as done in a case study of [8], or monitoring this appU/domU for malicious network traffic, as done in [18]. Thus, the history of critical files can be of benefit the provider.

B. Process Behaviour Log Files to Assist in Mitigating Risks Associated with Threat 1 to Benefit the Provider

1) *What is a process behaviour log files*: This paper uses Linux processes as an example of processes in a domU, and uses a process and command interchangeably. Actually, an appU (in the template) becomes one or more processes/commands. This paper assumes that each appU becomes only one process. The provenance collection in PASSXen also has the creation, access, and destruction of processes in domUs. However, this paper briefly discusses a **process behaviour log file as a record of some of the process's activities, such as the process name and id, a file name of a file that this process has access to, and the owner id of the process**. For example, command '*cat addr.txt*' is when a *cat* (concatenate) command in Linux is accessing the text file.

Table II can be an example of a *cat* process behaviour log file. The content of the log of the *cat* can be the name of the process (*p_nm*, *cat*), the id of the process (*p_id*, 4000), the

name of the file accessed by cat process, (f_nm, addr.txt), and the id of the owner of this process (p_ownId). This log can be different, depending on who (a provider, customer, or auditor) wants it and what it is for. The table shows only the content for the purpose of identifying a spam domU below.

p_nm	p_id	f_nm	p_ownId
cat	4000	addr.txt	1002

TABLE II. A PROCESS BEHAVIOUR LOG FILE TO SHOW THE MALICIOUS CAT COMMAND READING ADDR.TXT

2) *Process behaviour log files to assist in mitigating risks associated with spam activities*: First, the providers may use a network monitoring (as used in [18], [32]). The monitoring in [18] can pinpoint which processes inside a domU are responsible for malicious or heavy network traffic leaving the domU. Second, the provider may discover the commands in this domU that send emails (e.g., mail command in Linux). To send email to a@b.c with subject as spam, the command can be 'mail -s spam a@b.c' [8]. Third, the providers record behaviours of the mail command as evidence to identify accurately this spam domU. We have already demonstrated recording this mail command behaviours as a log file (by capturing the subject and the victim's email address parameter of mail command above) in a domU in our previous work [8]. Lastly, we can take one step further from [8]. Mail command to send spam emails can be 'mail -s subject \$(cat addr.txt)'. It sends emails to all victim addresses in addr.txt. Thus, this command involves addr.txt (Table II, column 3). Hence, this file could be very important evidence to identify these spam activities. Section VII-B discusses an example of the complete process behaviour log file to show the malicious mail and cat command involving spam activities.

Thus, regarding addr.txt and when the providers have already pinpointed the mail command that sends spam emails, as demonstrated by [8], they can then combine capturing the mail command malicious behaviours (in [8]) with the cat command behaviour log file (in Table II, this command reading addr.txt) as evidence to assist in identifying this spam domU. Thus, process behaviour log files (e.g., the cat's behaviour log) can be useful to assist in accurately analysing and identifying spam domUs in the providers' IaaS cloud. Reducing a number of criminal (spam) domUs should maintain the providers' reputation. This is because customers may buy the cloud product based on providers' reputations [31].

V. THE PROTOTYPE IMPLEMENTATION

A. Aim of the Proposed Logging System

Figure 4 shows the context of a domU for this implementation. In the figure, Alice rents a Linux domU. She has a critical file (s.txt) in diskU. Read application (in the rectangle in user level) is an appU. The name 'read' is the appU's name and also the process name of the appU. Alice can run this application to read s.txt (the dot-arrow line) with root or her privilege. *read_mem* (the ellipse in memU) is the memory space of this read appU/process. This memory space holds all information we need to record. This information (as shown in row 1 of Table I) is a file name of s.txt, a process Id (e.g., 4624) and a process name of read appU (e.g., read), and an owner Id of read process (e.g., 1002 which is Alice's user Id in this domU).

Hence, **the aim of the logging system in this implementation** is to record the history of critical files (only row 1 of Table I, as discussed above), and then store them in a log file.

Fig. 4. The context of the domU in the implementation: a read appU, a critical file (s.txt) in diskU, and *read_mem* (in memU) as memory space of read appU

B. System Architecture of the Logging Solutions

Figure 5 is the system architecture of the proposed logging system which is based on the template (Figure 2). The main components are logger, P1/libVMI [33], and F3 as a log file. LibVMI is a C library that can read the memory space (*read_mem*) in memU from domU. Step 1: the logger (in dom0 user level) is an app0 that calls libVMI to access memU (step 2) to get the information in *read_mem* such as a file name of s.txt, as discussed above. Step 3: the logger writes the information (as shown in row 1 of Table I) into F3. This architecture is based on the template. Thus, it can be used to analyse the security of the proposed logging system before deployment. **How the**

Fig. 5. The system architecture of the proposed logging system

logger knows whether a file in a domU is critical or not: in this prototype implementation, this paper does not discuss how to manage the logger application. Thus, we assume that the logger knows the critical file.

C. The Security Analysis of the Proposed Logging System

DomUs cannot tamper with the logging components inside dom0: our implementation deploy the logger and libVMI in dom0, and the log files (F3) in hw0. The advantage of doing this is that domUs cannot tamper with these components. In contrast, when deploying the components inside domU, this allow the owner of this domU to tamper with these components, as argued by [11], [34], [8], [19]. However, deploying

the logging components in dom0 also need to consider the other security aspects as discussed below. The solutions for these security issues is out of the scope of this paper.

First, **the security analysis of the log files**: we use the logging system architecture (Figure 5) as a tool for this analysing. For example, the security relevant question is how to ensure the integrity of the log files that are stored in disk0 which are fully owned and controlled by a provider. The provider may maliciously learn about, or alter the log files. Second, **the security analysis of logging processes**: the next security relevant question is how can an auditor ensures the integrity of the logger or libVMI, which is run by the provider in dom0. Locating these components in this location can be a security risk. This is because the provider may maliciously modify the logger's code to produce contents of log files which benefit himself. We discuss only the questions above as an example of the analysis of the proposed logging system.

VI. THE RESULT AND HOW TO OBTAIN IT

A. The Result

In Figure 5, the logger command (app0) runs in dom0 (the 1st line in Figure 6). It keeps checking memU until read command is performed and existed in memU. When read command is performed in domU (the 1st line in Figure 7), the logger then keeps waiting until the read command reads s.txt. When the read command starts reading s.txt (the 2nd line in Figure 7), the logger immediately extracts necessary information (2nd line of Figure 6). These information are the file name (s.txt), read process name (read), process id (4624), and the id of the owner of read command (1002). After that, the logger writes this extracted information to F3.

```
①[root@perosteck examples]# ./logger
②s.txt, By_proc_Id: 4624, By_proc_name:read, By_proc_owner_Id:1002
```

Fig. 6. The logger running in dom0 to record history of critical file s.txt

```
①[alice@localhost ~]$ ./read
.....②.start reading ..... s.txt
```

Fig. 7. The read command running in domU and reading critical file s.txt

B. How to Obtain the Result

To obtain a critical file name (s.txt), Figure 8 shows the detail of *files point. Step 1 in Figure 8, *files is a pointer of read process's *task_struct* that points to *files_struct* structure. It contains an *fdt* pointer that points to the first open file or *fd[0]* inside *fdtable* structure, step 2. Each *fd[i]* points to each open file that is opened by read process, step 3-4. It is assumed that read process is reading s.txt which is pointed by *fd[3]*, step 4. Figure 9 shows how, after the logger locates *fd[3]* pointer (step 4), that logger obtains the file name (string s.txt). Step 1, the logger finds *f_path* field (inside *file* structure), which is a *path* structure. This structure contains a *dentry* pointer which points to *dentry* structure, step 2. *Dentry* contains *d_name* that is a *qstr* structure (step 3) which has the name pointer field that points to string s.txt, step 4.

Fig. 8. Detail of files pointer (of read process's *task_struct*) pointing to s.txt

Fig. 9. How the logger obtains the file name of a critical file s.txt

VII. DISCUSSION

The logger (Figure 5) can record the result as in line 2 of Figure 6. However, this section discusses how the logger can be modified to produce many types of log files, which assist in mitigating the risks associated with threat 1 to benefit both customers and providers. This section then compares the proposed system with previous work, and discusses privacy and confidentiality concerns of both sides due to the log files.

A. Forming a Complete History of Critical Files from the Results to Mitigate Risks Associated with Threat 1 to Benefit the Customers

1) *Analysing malicious incidents when Alice is a single user in a domU*: In the experiment, the domU is for a single user. However, Alice owns two accounts: the root (can run all appUs in the domU) and alice (an administrator). An example of the complete content of history of s.txt (Table III) can be constructed from the result (2nd line of Figure 6). Note that, last_acc (column 5) presents the last accessed times of the file. The asterisks (*) in the following tables indicate possible malicious events in domUs. When the history of s.txt

f_nm	p_id	p_nm	p_ownId	last_acc
s.txt	4624	read	1002	t1
s.txt	4002	read	1002	*t2
s.txt	4003	*maliciousRead	1002	t3
s.txt	4004	read	*1003	t4

TABLE III. AN EXAMPLE OF THE COMPLETE CONTENT OF THE HISTORY OF S.TXT

is available to Alice, she may audit s.txt. For example, row 1 can be a normal event when Alice logs in to her domU. Then, she runs her read appU (p_nm) to access s.txt (f_nm, the dot-arrow line in Figure 4) with her or root permission.

The history of s.txt (Table III) shows Alice's Id (1002) in column 4. In the case root's or Alice's password is compromised, which may be caused by threat 1, row 2-3 can be suspect incidents relate to her file. Row 2 can be undesired access to s.txt. The reason can be that Alice has never accessed s.txt at t2 time (column 5). Thus, she may suspect that attackers may have accessed s.txt. Another case in row 3: Alice has never used maliciousRead appU (column

3) to access s.txt. Thus, she may also suspect that attackers may be doing this. After pinpointing maliciousRead, in order to obtain more evidence for auditing, she may make further investigations using other methods (e.g., network monitoring as used in [18], [32]).

2) *Analysing malicious incidents when Alice is in a multiple users domU*: The domU in this implementation can be used by multiple users. Thus, Alice's domU have a root user, Alice (an administrator), and Bob (a standard user, who shares and can log in to Alice's domU). As discussed in Section III-A, this domU can be compromised by threat 1. Thus, attackers may obtain both Bob's and root's password. Then, they may log in to Alice's domU using Bob's account. Thus, they can run Alice's read appU (with root permission) to access s.txt. Thus, the evidence can be row 4 of Table III. This row can be a suspect incident that why and how Bob (id 1003, in column 4 row 4) accesses Alice's s.txt. Although the attacker uses the same appU (read, in column 3 row 4) which is normally used by Alice to access s.txt, the log (from our experiment) still shows Bob's Id (1003), which is not an owner of s.txt. Consequently, Alice could eventually discover this suspicion from the history of s.txt. Thus, the history information in Table III can be evidence to assist in analysing undesired incidents inside domUs. As a result, this can mitigate risks associated with threat 1 to benefit customers.

3) *Another type of log files*: To enhance flexibility of auditing in the cloud using logging systems, the logger can be modified. For example, an auditor may want to know whether a particular appU (as one that has an ability to view text files e.g., gedit in Linux) read s.txt or not. Thus, it is possible to modify the logger to record the history of this particular process instead of history of s.txt. Thus, this type of log file could be as in Table IV, which shows a list of all files that were accessed by gedit. Then, the auditor can check whether this process reads s.txt or not. In the table, the auditor can see that gedit reads s.txt (row 2 column 3). Thus, this type of log files can be useful to enable flexibility of the auditing.

p_nm	p_id	to_file_nm	p_ownId
gedit	4000	a.txt	1002
gedit	4003	*s.txt	1002

TABLE IV. A LIST OF ALL FILES ACCESSED BY GEDIT APPU

B. How the Result Assists in Mitigating Risks Associated with Threat 1 to Benefit the Providers

For dealing with the compromising of dom0, as discussed in Section IV-A3 that history of critical files (Table I) can be evidence for the providers to be aware that their cloud infrastructure may be compromised. From Table III, maliciousRead appU used by Bob (row 3, column 3) can be used as a trigger or pinpoint for the providers to make further investigations. For dealing with spamming, to record the command 'mail -s subject \$(cat addr.txt)', Table V shows that both malicious mail (column 1) and cat (column 2) command involving spam activities. One can see that this domU owner (id 1002, column 4 and 5) uses the combination of both commands to send spam emails, cat to read addr.txt (column 3) and then mail to send emails to all the victim addresses in addr.txt.

p_nm1	p_nm2	f_nm1	p_ownId1	p_ownId2
mail	cat	addr.txt	1002	1002

TABLE V. A PROCESS BEHAVIOUR LOG FILE TO SHOW THE MALICIOUS MAIL AND CAT COMMAND INVOLVING SPAM ACTIVITIES

C. Related Work and Comparisons

Our proposed logging system architecture (Figure 5) deploys libVMI (previously known as XenAccess [19]). LibVMI is based on six high level requirements of programming guidelines or good security guidelines [19]. Our proposed system architecture inherits these requirements while achieving the history of critical files. The first two requirements are: no superfluous modifications to the hypervisor, and no modifications to domUs. These requirements involve a trusted computing base (TCB) which is a significant factor when building logging systems in the cloud. This is because in order to evaluate the trustworthiness of a software system (e.g., our proposed logging system), it is necessary to identify its TCB [35]. Thus, the size of the TCB should be as small as possible [19]. However, [36] points out that an OS is difficult to analyse because of its size and complexity. He also argues that there is too much TCB when deploying an application in an OS (e.g., domUs) that is running on top of a hypervisor. [37] argue that an OS code changes rapidly over time. The changes may increase the size of the OS and its complexity, and as a result, its TCB. Thus, the TCB can be a very important aspect of security, in order to propose logging systems in the cloud. For example, the proposed systems in [36], [35], [19], [38] (that can be considered as solutions to mitigate risks associated with the cloud problem) concern reducing the TCB size along with their proposed systems. Recent work that extensively relates to a secure cloud computing environment such as that of [39] also focuses on reducing the TCB in their proposed architecture.

Thus, this section discusses related work that involves logging or detecting mechanism in an IaaS. It also compares the logging system in this paper with those in related work based on TCB and achieving a history of critical files. First, Flogger [11] and PASSXen [17] can provide a history of critical files. However, Flogger has logging processes that are distributed across domU and dom0. PASSXen also has logging processes that are distributed across the hypervisor, domU and dom0. Therefore, the proposed architecture in this paper yields less TCB than Flogger and PASSXen. This is because the TCB of our architecture includes only hypervisor and dom0, not domU, whereas their TCB includes domU. Second, [18] propose a network monitoring application that identifies which process inside a Windows domU is responsible for malicious network traffic leaving this domU. Additionally, [19] present a demo monitoring program in dom0 that outputs all file/directory creation/removals happening in domU's /root directory. Both works deploy XenAccess in dom0. Thus, these systems have small TCB (hw0, Xen, and dom0) the same as the proposed architecture in this paper. However, they are not designed to achieve a history of critical files, but could be modified to add this functionality.

D. Privacy and Confidentiality Concerns of Customers and Providers due to the Log Files

The history of critical files and process behaviours log files discussed in this paper is detailed records of activities of pro-

cesses and files in a customer VM. For auditing purposes, this history information can be investigated by a third party. Thus, this information may disclose customer business activities, or malicious activities inside provider's cloud infrastructure. This leads to privacy and confidentiality issues of both customers and providers. However, [14] argue the privacy issues is manageable. For example, it is important to consider what is being recorded, and who can access this recorded information. Thus, logging system management should return the recorded information at different levels of detail, depending on who needs it. To address the privacy and confidentiality issues, this may lead to further research such as balancing the privacy and its usage (e.g., [40]). This is out of the scope of this paper.

VIII. CONCLUSIONS

We argue that Cloud Security Alliance/CSA threat 1 (related to criminal activities in customers' VMs such as spamming) can have serious effects for both provider and customer sides. The examples are compromising provider's hosting system/dom0, consequently, customer virtual machines. The paper then proposes, implements, and discusses logging solutions to mitigate risks associated with threat 1 in infrastructure as a service (IaaS) cloud. We argue that our result can be used to form log files called history of critical files and process behaviour log files. We then demonstrate in detail how these log files can mitigate the risks to benefit both sides. To collect the log files, our proposed system yields smaller trusted computing base (TCB) compared to previous work [11], [17]. Although, work by [18], [19] can yield the same TCB as our proposed system. However, they are not designed to achieve the history of critical files as discussed in this paper. When we have the full implementation of the proposed system, it should be an appropriate point to have the evaluation of the system performance impact or the scalability. However, history of critical files and process behaviour log in this paper can be important evidence to clarify what is going on in the cloud. Thus, these log files assist in enhancing the accountability in the cloud, consequently, assist in mitigating risks associated with CSA threats to benefit both consumers and providers.

REFERENCES

- [1] R. Chow, P. Golle, M. Jakobsson, E. Shi, J. Staddon, R. Masuoka, and J. Molina, "Controlling data in the cloud: outsourcing computation without outsourcing control," in *Proceedings of the 2009 ACM workshop on Cloud computing security*, 2009.
- [2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, 2010.
- [3] CSA, "Security guidance for critical areas of focus in cloud computing version 3," Tech. Rep., 2011.
- [4] W. Dawoud, I. Takouna, and C. Meinel, "Infrastructure as a service security: Challenges and solutions," in *Informatics and Systems (INFOS), 2010 The 7th International Conference on*, 2010.
- [5] R. De Paris, "Remi-a middleware to handle molecular docking simulations of fully-flexible receptor model in hpc environment," Master's thesis, 2012.
- [6] S. Hazelhurst, "Scientific computing using virtual high-performance computing: a case study using the amazon elastic computing cloud," 2008.
- [7] P. Watson, P. Lord, F. Gibson, P. Periorellis, and G. Pitsilis, "Cloud computing for e-science with carmen," in *2nd Iberian Grid Infrastructure Conference Proceedings*, 2008.
- [8] W. Wongthai, F. Rocha, and A. van Moorsel, "A generic logging template for infrastructure as a service cloud," in *Advanced Information Networking and Applications Workshops, 27th International Conference on*, 25-28 March 2013.
- [9] R. K. Ko, P. Jagadpramana, M. Mowbray, S. Pearson, M. Kirchberg, Q. Liang, and B. S. Lee, "Trustcloud: A framework for accountability and trust in cloud computing," *Services, IEEE Congress on*, 2011.
- [10] R. Ko, M. Kirchberg, and B. S. Lee, "From system-centric to data-centric logging accountability, trust & security in cloud computing," 2011.
- [11] R. Ko, P. Jagadpramana, and B.-S. Lee, "Flogger: A file-centric logger for monitoring file access and transfers within cloud computing environments," 2011.
- [12] CSA, "Top threats to cloud computing, version 1.0," Tech. Rep., 2010.
- [13] —, (2012) Top threats to cloud computing survey results update.
- [14] A. Haeberlen, "A case for the accountable cloud," *SIGOPS Oper. Syst. Rev.*, 2010.
- [15] J. Spring, "Monitoring cloud computing by layer, part 1," *Security Privacy, IEEE*, 2011.
- [16] G. Aceto, A. Botta, W. de Donato, and A. Pescapé, "Cloud monitoring: Definitions, issues and future directions," 2012.
- [17] M. C. Peter Macko and M. Seltzer, "Collecting provenance via the xen hypervisor," 2011.
- [18] B. Dolan-Gavitt, B. Payne, and W. Lee, "Leveraging forensic tools for virtual machine introspection," Tech. Rep., 2011.
- [19] B. Payne, M. de Carbone, and W. Lee, "Secure and flexible monitoring of virtual machines," in *Computer Security Applications Conference. Twenty-Third Annual*, 2007.
- [20] Amazon. (2012) Amazon elastic compute cloud (amazon ec2).
- [21] H. Fang, Y. Zhao, H. Zang, H. Huang, Y. Song, Y. Sun, and Z. Liu, "Vmguard: An integrity monitoring system for management virtual machines," 2010.
- [22] CSA. (2013) The notorious nine: Cloud computing top threats in 2013.
- [23] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Cross-vm side channels and their use to extract private keys," in *Proceedings of the ACM conference on Computer and communications security*, 2012.
- [24] CSA. (2010) Cloud security alliance and hp identify top cloud security threats in new research report.
- [25] A. Bates, B. Mood, J. Pletcher, H. Pruse, M. Valafar, and K. Butler, "Detecting co-residency with active traffic analysis techniques," in *Proceedings of the 2012 ACM Workshop on Cloud computing security workshop*, 2012.
- [26] W. Kandek. (2010) Virtualization vulnerabilities up and coming.
- [27] K. Onoue, Y. Oyama, and A. Yonezawa, "Control of system calls from outside of virtual machines," in *Proceedings of the 2008 ACM symposium on Applied computing*, 2008.
- [28] "A survey on security issues and solutions at different layers of cloud computing," *The Journal of Supercomputing*, 2013.
- [29] D. Chisnall, *The definitive guide to the xen hypervisor*, 2007.
- [30] P. S. Wooley, (2011) Identifying cloud computing security risks.
- [31] A. Bisong and S. S. M. Rahman, "An overview of the security concerns in enterprise cloud computing," *International Journal of Network Security & Its Applications*, 2011.
- [32] A. Haeberlen, P. Aditya, R. Rodrigues, and P. Druschel, "Accountable virtual machines," in *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, 2010.
- [33] B. Payne. Vmtools Virtual machine introspection tools.
- [34] J. Schiffman, Y. Sun, H. Vijayakumar, and T. Jaeger, "Cloud verifier: Verifiable auditing service for iaas clouds," Tech. Rep., 2013.
- [35] D. G. Murray, G. Milos, and S. Hand, "Improving xen security through disaggregation," in *VEE*, 2008.
- [36] J. M. McCune, "Reducing the trusted computing base for applications on commodity systems," Ph.D. dissertation, 2009.
- [37] J. M. McCune, A. Perrig, and S. O. service), *Bootstrapping Trust in Modern Computers*, ser. SpringerBriefs in Computer Science,, 2011.
- [38] F. Rocha and M. Correia, "Lucy in the sky without diamonds: Stealing confidential data in the cloud," in *Proceedings of the 41st International Conference on Dependable Systems and Networks Workshops*, 2011.
- [39] F. Rocha, T. Gross, and A. v. Moorsel, "Defense-in-depth against malicious insiders in the cloud," in *Cloud Engineering, IEEE International Conference on*, 2013.
- [40] T. Burghardt, K. Böhm, A. Guttman, and C. Clifton, "Anonymous search histories featuring personalized advertisement-balancing privacy with economic interests," *Trans. Data Privacy*, 2011.