COMPUTING SCIENCE

A Generic Logging Template for Infrastructure as a Service Cloud

Winai Wongthai, Francisco Liberal Rocha and Aad van Moorsel

TECHNICAL REPORT SERIES

No. CS-TR-1367

January 2013

A Generic Logging Template for Infrastructure as a Service Cloud

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

Abstract

Infrastructure as a Service (IaaS) consists of a cloud-based infrastructure to offer consumers raw computation resources such as storage and networking. These resources are billed using a pay-per-use cost model. However, this type of infrastructure is far from being a security haven as the seven main threats defined by the Cloud Security Alliance (CSA) indicate. Using logging systems can provide evidence to support accountability for an IaaS cloud, which helps us mitigating known threats. In this paper, we research to which extent such logging systems help mitigate risks associated with the threats identified by the CSA. A generic architecture 'template' for logging systems is proposed. This template encompasses all possible instantiations of logging solutions for IaaS cloud. We map existing logging systems to our generic template, and identify a logging solution to mitigate the risks associated with CSA threat number one (related to spam activities). We then argue that the template we suggest can be used to perform a systematic analysis of logging systems in terms of security before deploying them in production systems.

Bibliographical details

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

A Generic Logging Template for Infrastructure as a Service Cloud
[By] W. Wongthai, F. Liberal Rocha, A. van Moorsel

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

(Newcastle University, Computing Science, Technical Report Series, No. CS-TR-1367)

Added entries

NEWCASTLE UNIVERSITY

Computing Science. Technical Report Series. CS-TR-1367

Abstract

Infrastructure as a Service (IaaS) consists of a cloud-based infrastructure to offer consumers raw computation resources such as storage and networking. These resources are billed using a pay-per-use cost model. However, this type of infrastructure is far from being a security haven as the seven main threats defined by the Cloud Security Alliance (CSA) indicate. Using logging systems can provide evidence to support accountability for an IaaS cloud, which helps us mitigating known threats. In this paper, we research to which extent such logging systems help mitigate risks associated with the threats identified by the CSA. A generic architecture 'template' for logging systems is proposed. This template encompasses all possible instantiations of logging solutions for IaaS cloud. We map existing logging systems to our generic template, and identify a logging solution to mitigate the risks associated with CSA threat number one (related to spam activities). We then argue that the template we suggest can be used to perform a systematic analysis of logging systems in terms of security before deploying them in production systems.

About the authors

Winai Wongthai 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 is supervised by Professor Aad van Moorsel and Dr Feng Hao.

Francisco Liberal Rocha is a second year PhD student under the supervision of Professor Aad van Moorsel with the School of Computing Science at Newcastle University. His research interests include systems security, software security and security architectures. His latest work focus on developing prevention techniques that guarantee memory confidentiality and integrity for cloud consumers, in the presence of malicious insiders in cloud computing.

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

IAAS

CLOUD MONITORING

LOGGING SYSTEM

A Generic Logging Template for Infrastructure as a Service Cloud

Winai Wongthai*, Francisco Liberal 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—Infrastructure as a Service (IaaS) consists of a cloud-based infrastructure to offer consumers raw computation resources such as storage and networking. These resources are billed using a pay-per-use cost model. However, this type of infrastructure is far from being a security haven as the seven main threats defined by the Cloud Security Alliance (CSA) indicate. Using logging systems can provide evidence to support accountability for an IaaS cloud, which helps us mitigating known threats. In this paper, we research to which extent such logging systems help mitigate risks associated with the threats identified by the CSA. A generic architecture ‘template’ for logging systems is proposed. This template encompasses all possible instantiations of logging solutions for IaaS cloud. We map existing logging systems to our generic template, and identify a logging solution to mitigate the risks associated with CSA threat number one (related to spam activities). We then argue that the template we suggest can be used to perform a systematic analysis of logging systems in terms of security before deploying them in production systems.

Keywords—IaaS; cloud monitoring; logging system;

I. INTRODUCTION

Many people argue that the cloud is the future of computing, and has the potential to transform the IT industry in a wide variety of application areas [1]. This includes infrastructure as a service (IaaS) which is used by enterprise, government, and academia, such as in medical experimentation [2], [3]. IaaS provides a base on which to build Platform as a Service (PaaS) or Software as a Service (SaaS) offerings, as pointed out in [4], [5]. However, trust of consumers in a cloud is extremely important for its continued proliferation. Customers want to know how and where their data and related logs are stored and who has access to them. The Cloud Security Alliance (CSA) has extensively published on this topic, see for instance Security Guidance for Critical Areas of Focus in Cloud Computing report [4] and Top Threats to Cloud Computing report [6]. The latter document is used in this study, providing the basic threats for which we identify logging-based accountability solutions.

Although by itself not sufficient to mitigate the risks associated with the threats, a logging system is an important aspect of any accountability solution in the cloud, as argued by many researchers [1], [7], [8], [9]. In this paper, a logging system composes of logging processes and log files. A logging process performs logging-related tasks, whereas log files are used for storing contents produced by the logging processes. Each logging process has its own

task(s). For example, a main task of a logging process called Flogger [10] is intercepting file operations of a virtual machine (VM) for the purposes of accountability in the cloud. It manages to store such intercepted data as log files. We argue that logging systems are a core component for cloud monitoring/accountability, which can assist to mitigate risks associated with CSA threats, as also argued by [6], [9].

Recent works [7], [8] give an outline of accountability/monitoring-related solutions for the cloud. These works involve the logging processes and the log files that will be in the cloud machines. However, they do not discuss where exactly they could be in the machines (which involve visualization infrastructure and more than one party), and who will manage them and why.

Other works that involve design and implementation of the logging solutions which could be applied in the cloud include [11], [12], [9], [13]. These works focus on monitoring and logging tasks. Ideally, a logging system can be analyzed with respect to its features and achievable goals before deployment. However, these authors do not give completed concerns about the security analysis of their logging systems themselves (which include the logging processes and the log files) before deploying them to the real world IaaS. We believe it is important to perform a security analysis of the monitoring/logging systems themselves as will be fully discussed in Section II-D. In order to assist satisfying this concern, we propose a systematic approach to constructing and analyzing the security of a logging system itself before deployment. We achieve our objectives by proposing a generic template that encompasses all possible logging solutions in IaaS to mitigate the risks associated with all seven CSA threats.

We use the term template to indicate that we consider the union of all possible logging architectures that could be instantiated. In that sense, the template by itself is not meant to be implemented, but offers the building blocks from which one can choose to create an actual logging solution. As a result, the template is considered as generic. Moreover, with respect to faster system development, the template is flexible and implementation independent, and obeys reuse-ability. It could be a starting point of building accountability systems to address lack of trust in an IaaS.

Summary of Contributions: This paper has the following contributions. First, we propose a generic logging template for IaaS and discuss how it can be used to mitigate the

Figure 1. The IaaS architecture.

risks associated with the CSA threats. We then identify how existing logging solutions (in particular HP Flogger [10]) can be mapped on the template for Flogger’s security analysis purposes. Finally, in order to demonstrate how the template can be used to systematically instantiate logging systems to deal with the CSA threats, we design and implement a logging system to mitigate risks associated with CSA threat 1, which refers to the use of cloud computing for e-mail spamming and similar activities.

II. BACKGROUND

A. IaaS Architecture

This paper focuses on IaaS cloud. Its architecture is composed of a provider and a customer side, see Fig. 1. The provider side can be an organisation that offers consumer side services as rent-able VMs/domUs (e.g., Amazon EC2). The customer side can be a person or an organization that can remotely access these domUs via Internet. Our version of IaaS architecture is based on the Xen architecture. The main components of a IaaS architecture are hw, hypervisor, dom0, and a number of domUs (domU1,..., domUn). We use ‘hw’ as a short for hardware, which works as a host of a hypervisor, and all guest OSes (dom0 and domUs). It is managed, maintained and owned by the provider who uses it to store and process customers’ data. A hypervisor is a layer of software running directly on hw and allows the hw to run multiple guest OSes at the same time. It is an interface for all hardware requests from the guest OSes.

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, but has no direct access to the hw. It is a VM, a product that customers can rent from a provider. Thus, it is virtually owned by a customer, and operates independently in the system. However, a domU is actually launched and controlled by dom0. The other IaaS architecture can be based on the layers of a cloud as discussed in [4], [14], [5]. [14] argues that a cloud composes of seven layers: facility, network, hardware, OS, middleware, application, and the user. It also states that IaaS composes of facility, network, hardware, and OS. However, PaaS can always be built up by adding extra layers on top of IaaS layers.

B. The Seven CSA Top Threats to Cloud Computing

The first is *abuse and nefarious use of cloud computing*. For example, people with bad intentions can register legitimately to rent domUs, and later they engage in malicious activities (e.g., spamming). The second is *insecure application programming interfaces (APIs)* that are offered by providers. Customers have to use the APIs to interact with their rented domUs. If the APIs are insecure by design, they can be the cause of security issues (e.g., confidentiality) of customers’ data in such domUs. The next threat is *malicious insiders* who usually have full privilege over dom0 in IaaS environment. Thus, they may exploit customers’ confidential data in domUs. The forth is *shared technology vulnerabilities*. For example, an owner of domU may gain unauthorized access to the data of other domUs. The fifth threat is *data loss or leakage*. Customers’ data loss may occurred due to deletion of records without a backup of the original data in domUs. The following one is *account, service and traffic hijacking*. Attackers reuse customers’ credentials and passwords, which can be stolen through phishing and fraud. Thus, they can spy on customers’ activities in domUs. The last threat is *unknown risk profile*. It is a fact that a customer does not know about important information (such as who is sharing her infrastructure) that can be used to predict possible security risks of her domU. As a result, these threats can be the cause of a lack of trust from the customers’ points of view. Hence, we intend to study how accountability, monitoring, and logging approaches could assist to address these threats as discussed in the next section.

C. Accountability, Monitoring, and Logging in the IaaS

Enhancing accountability is necessary in order to mitigate the risks associated with all CSA threats, and thus enable customers to securely and confidently adopt IaaS. One important mechanism for accountability is monitoring, which deploys a logging system as its core. Accountability in IaaS roots on a similar concept with Accountable Cloud (AC) in [8]. AC is the cloud that (i) the consumers can investigate whether the provider is processing their data according to the agreement, (ii) if errors are reported, the provider can provide the evidence to verify who is responsible for the problem, and (iii) if there is an argument, the provider can present a proof to the third party. Therefore, AC could be a very good concept to deal with the CSA threats in IaaS.

1) *Monitoring as a Solution to the CSA Threats*: In [6], CSA states that a monitoring approach is one of the remediations in its list of solutions for each threat. It also argues that monitoring can mitigate threats 1, 3, 4, 6, and 7. In [9], HP also summarizes that some sort of log files (such as histories of file access on the provider side) can aid the providers and the consumers to mitigate CSA threats 1, 2, 3, 5 and 7.

2) *Monitoring and its Components in IaaS*: This paper considers a monitoring system to be a system that can monitor which activities take place inside dom0 and/or

domUs, and its main subsystem is a logging system. The basic idea of a logging system is to use logging processes to record particular information, such as in [12] the logging processes record incoming and outgoing network packets of domUs. Then they store such information as log files to be used on particular purposes such as evidence for identifying malicious network traffic in the domUs. This paper did not discuss real-time elements of monitoring such as triggers. It is because these elements should be fundamental for monitoring system, and should always be considered and implemented before deploying the system. We rather focus on the security analysis of logging processes and log files, which are critical components in a logging system.

There is some work outlining the components of monitoring mechanisms. First, Chow et al. [7] introduced a trusted monitor, which is installed in a cloud server, and produces a proof of compliance for the data owner. This compliance can be used to indicate that the data has not been manipulated against the access policies. Second, Haeberlen [8] introduces an accountable cloud (AC) to mitigate losing control of the consumer data in the cloud. This work outlines a set of building blocks of AC. One of the essential blocks is tamper-evident logs which are all inputs and outputs of cloud machines in the provider side. Both works have the same two important components which are logging processes and log files. For example: in [7], the trusted monitor is indeed as logging processes that produce the compliance (as log files); and in [8], the tamper-evidence logs is log files that could be eventually produced by a logger (as logging processes).

These works only give the outline of the logging processes and the log files that will be in the cloud machines. They did not discuss a set of security relevant questions such as where the log files and logging processes could be in a IaaS architecture, should they run or be stored in the hw, hypervisor, dom0, or domU, and who will manage them and why. Hence, this paper considers the logging processes and the log files as critical components in our proposed template, in Section III. It also focuses on where precisely the logging processes and log files can be inside the IaaS architecture. It is because the locations of these components in the IaaS can answer such questions above, which can directly and significantly affect security concerns of these components as will be fully discussed in Section II-D. This is our motivation to study and to propose *a generic logging template* to enhance the security of accountability/monitoring in IaaS.

D. Security Analysis of Logging Systems in an IaaS

The security analysis of logging systems themselves is very important because the logging processes and log files are critical components and so they need to satisfy a set of security properties (e.g., integrity and privacy). Hence, without the security analysis of logging systems, it could be difficult to effectively and efficiently build and deploy logging systems that can satisfy those security properties. A

Figure 2. The overall view of a generic logging template and its components (logging process:Px, and log files:Fy).

IaaS environment, which involves virtualisation and more than one party, makes the security analysis of logging systems more complicated than ones within a private organization. Haeberlen [8] pointed out that one of the research challenges of the Accountable Cloud is that it needs to have mechanisms to allow legacy users to access the logging machine (which deploys logging processes) but not to maintain the log files. Crosby and Wallach [15] agreed that ensuring the integrity of the log files is a critical part of a larger system. Therefore, it can be difficult to avoid the security analysis of logging systems themselves before deployment. Hence, this paper considers the security analysis of logging systems as a significant criterion to establish the goals of the template as will be discussed in the next section.

III. A GENERIC LOGGING TEMPLATE

This generic logging template is a generalisation of previous research work concerned with logging systems (in the related work section) and how they fit together with the components of a IaaS architecture. We investigate logging process and log file components of each work. Then we locate these components into the IaaS architecture. To deal with the CSA threats as discussed in Section II-B, **the first goal of the template** is that it can be used to instantiate new logging system architecture to mitigate the risks associated with all CSA threats in IaaS environment. Moreover, it is as stated earlier in Section II-D that the security analysis of logging systems is critical before they are deployed into the real world IaaS. Hence, **the second goal** is that the security of the newly built logging system architecture, that is based on the template, can be systematically analyzed before deployment.

A. The Details of the Template and its Components

The template and its components are presented in Fig. 2 and Table I-III. All defined components can be divided into three sets. The first one is the IaaS set of components (all white boxes) which are derived from the IaaS architecture (Fig. 1). This set of components includes hw0, hypervisor,

Table I
THE IAAS SET OF COMPONENTS (ALL THE WHITE BOXES)

Component	Description/Functions	Examples	Locations in IaaS environment
hw0	the hw described in Section II-A, it is managed and owned by a provider/dom0, and consists of a disk0 and mem0	a PC/server	in the provider side infrastructure
disk0	a physical disk of the hw0, it is managed by dom0	server's disks	physically in hw0
mem0	a main memory of the hw0	server's main memory	physically in hw0
hypervisor	described in Section II-A, it concurrently runs a dom0 and domUs	Xen	on top of hw0
dom0	described in Section II-A, it consists of app0	a Fedora 16 dom0	on top of the hypervisor
app0	an application that runs inside dom0 by a provider	a logging-related application	in dom0 user level
domU	described in Section II-A, it consists of appU, hwU	a Fedora 16 domU	on top of the hypervisor
appU	an application that runs inside domU by an owner of domU	mail application (a <i>mail</i> command in Linux system)	in domU user level
hwU	virtual hardware that consists of diskU and memU	-	virtually in domU (hwU is physically in hw0; however, we virtually locate it in the domU kernel level)
diskU	a virtual disk of domU	-	virtually in hwU, and physically in disk0/hw0
memU	a virtual main memory of domU	-	virtually in hwU, and physically in mem0/hw0

Table II
THE LOGGING PROCESS SET OF COMPONENTS (SHADED BOXES)

Component	Description	Examples/Functions	Locations in IaaS environment
P1	1) a dom0 user level process or 2) special dom0 libraries for logging purposes	in PASSXen [11]: <i>Waldo</i> (to read provenance record from the log, store them in a database, index them, and access the database for querying); or <i>Coordinator-like</i> (to periodically consume temporary logs from mem0, and perform other tasks by communicating with other dom0 kernel level Px or dom0 user level Px) <i>libVMI</i> [16] (to be called, for example, by app0 to read memU for introspection purposes)	in dom0 user level
P2	a dom0 kernel level process	in HP Floggers System [10]: <i>Flogger</i> (to intercept dom0 file and network operations, then store such intercepted data into disk0); or in PASSXen [11]: <i>Analyzer</i> (to process the provenance records)	in dom0 kernel level
P3	a domU kernel level process	in HP Floggers System [10]: <i>Flogger</i> (to intercept domU file and network operations, then it could temporarily store such intercepted data into diskU in case the data are too big to be stored in memU); or in PASSXen [11]: <i>Interceptor</i> in domU (to intercept domU system calls, and temporarily store such intercepted records in a mem0)	in domU kernel level
P4	a process inside a hypervisor	in AVMs system [12]: <i>Logger</i> (to record information about incoming and outgoing network packets of domU)	in a hypervisor

Table III
THE LOG FILE SET OF COMPONENTS (SHADED BOXES)

Component	Description	Examples/Functions	Locations in IaaS environment
F1	temporary log data in diskU	in Floggers System [10]: a domU HP Flogger (interceptor/P3) could temporarily stores the domU intercepted data as <i>F1</i> in diskU (in case the data are too big to be stored in memU) for further processing	in diskU
F2	temporary log data in memU	in HP Floggers System [10]: as same as the example of F1, but Flogger/P3 could eventually and temporarily store the log data as <i>F2</i> in memU for further processing	in memU
F3	permanent log data in disk0	in HP Floggers System [10]: Flogger/P2 can permanently store the intercepted data (as <i>F3</i>) in disk0; or in AVMs system [12]: a Logger (P4) permanently stores the domU intercepted network data (as <i>F3</i>) in disk0	in disk0
F4	temporary log data in mem0	in PASSXen [11]: domU Interceptor (P3) stores intercepted domU system calls data (as <i>F4</i>) in mem0	in mem0

hwU, dom0, domU, app0, appU, disk0, diskU, mem0, and memU. Table I explains description, functions, examples, and locations in the IaaS environment of each component. The next ones are the logging process set and the log file set of components (shaded boxes) which can be created and then located into the first set. They are the keys of the template. In Table II, the logging process set composes of logging processes (Px, x=1,...,4). We place only one Px in particular locations as a representative of multiple Px in each location. Hence, it is possible to have more than one Px in a particular location such as it could be P2-4 in a dom0 kernel level, depending on design and implementation of the actual logging systems. Any Px can collaborate with another or others to achieve logging tasks. The log file set, in Table III, composes of Fy (y=1,...,4). The representative approach of Px can apply to Fy as well. Thus, it is possible to have F1-F3 in disk0. Table II and III present description, examples, functions, and, locations in the IaaS environment of each component in both sets.

B. Discussion of Locations of Px and Fy in the Template

Px can have different functions and can be deployed in different locations in the IaaS layers when dealing with different CSA threats. Fig. 2 and Table I - III show that there are four locations for Px. It can be in dom0 user level (P1), dom0 kernel level (P2), domU kernel level (P3), or hypervisor (P4). There are also four locations for Fy. It can be a temporary Fy, which is temporarily stored in either diskU (F1), memU (F2), or mem0 (F4) and can be a permanent Fy, which is stored in diskU (F3). Locations of Px and Fy can be used as their security indicators (e.g., privacy of Fy) as stated in Section II-C2 and II-D, and as will be demonstrated and discussed in the case studies, in Section III-B. We consider mem0 and memU as a component of the template because log data will eventually and temporarily be in these memory, which affects the security analysis of such log data. There are no Px and Fy located in domU user (without root privileges) level, and Fy could not be permanently stored in diskU. It is because they could be easily disabled by the owner of domU which will undesirably stop logging tasks as also argued by [10].

C. A Case Study of Mapping HP Flogger on the Template

As stated earlier in Section III-A, a logging system has more than one Px in a particular location, depending on its design and implementation. The Flogger system [10] actually consists of more than one Px in its dom0 and domU kernels. The template is generalized from related work in Section V. Therefore, one can map any logging systems in the related work section on the template. It is because that HP Flogger is one of the related works; therefore, it is easily mapped back on the template as shown in Fig. 3. We assume this mapping is for the purposes of security analysis of the logging system itself. However, the figure

Figure 3. Mapping some HP Flogger Px and Fx on the template.

Figure 4. The logging system architecture to mitigate the risks associated with threat 1.

shows only mapping of some main components (P2-3 and F3), which their examples/functions are already discussed in Table II and III. After mapping the Flogger, one can analyze security of this system based on the provided mapping, such as how to ensure the integrity of F3 that is stored in disk0 which is fully owned and controlled by a provider. Hence, all the related work in Section V can be mapped back on the template; and their security can be analyzed.

D. A Case Study of an Identification of the Appropriate Logging System Based on the Template to Mitigate the Risks Associated with CSA Threat 1

In order to satisfy the goals of the template, this case study demonstrates how to use the template to instantiate the logging system architecture, and how to analyze the security of this new built architecture. This case study, see Fig. 4, simulates spam activities by assuming that spammers rent a Linux VM/domU from an IaaS provider. They then use appU (the *mail* command) to send a spam email to a victim (see Fig. 5). The command (*c1*) used as spam activities is: *mail -s spamSubj winai.wongthai@ncl.ac.uk*. The *mail* command in *c1* has three arguments. They are *ag1* (*-s*), *ag2* (spamSubj), and *ag3* (winai.wongthai@ncl.ac.uk). Thus, it sends the string 'spamSubj' as an e-mail subject to the email address 'winai.wongthai@ncl.ac.uk'. **The case**

```
File Edit View Search Terminal Help
[root@localhost /]# mail -s spamSubj winai.wongthai@ncl.ac.uk
```

Figure 5. Spam activities performed by appU (*mail* command) in domU.


```
[root@callerton src]# ./examples/process-list
.....Waiting...for...mail...command.....
Found mail command, its detail below:
command name = mail
ag1 = -s
ag2 = spamSubject
ag3 = winai.wongthai@ncl.ac.uk
Finished writing ag2 and ag3 to the DB.
```

Figure 6. LA in dom0 extracts *c1*'s *ag2-3* as evidence for identifying spam activities.

study's goal is to capture *c1*'s *ag2-3*, and then write them to a database as the log files. These log files can be used by an auditor, a Trusted Third Party, or a provider as the evidence to identify spam activities in this rented domU later. We assume that *ag2-3* is sufficient to identify these spam activities.

1) Design and Implementation of the Logging System:

This logging system architecture is designed and based on the template. Thus, we can choose some components from the template (Fig. 2 and Table I-III) to build up the logging system. At the time of the experiment, we found libVMI [16] as the only available tool that can achieve the goal of the case study. It is an introspection library to read memory from VMs/domUs. We deployed libVMI as P1 (in dom0 user level) to read memU of this spamming domU. Therefore, we reuse libVMI as P1, then we choose F3 as log files to build the logging system architecture. As a result, Fig. 4 shows the overview of the newly designed logging system architecture according to this case study's goal. It is quite clear that the new architecture composes of only P1 (libVMI), F3 (log files), and app0 (the Logging Application: LA).

2) *The Result of This Case Study:* We run the *process-list* command (the LA) inside dom0 (the first line in Fig. 6). It keeps checking memU until *c1* is performed. When *c1* is performed as shown in Fig. 5, then LA extracts *c1*'s *ag2-3* (Fig. 6, the second and third lines from the bottom) then writes them to F3. The result in Fig. 5 and 6 shows that we can use an application (e.g., LA), in dom0 to capture information in memU or in the memory space of any process, or commands of appU (such as *c1*) in domU. We now have an appropriate logging system to mitigate the risks associated with CSA threat 1. However, this system is in the IaaS which involves at least two parties (provider and customer). Hence, the system's security needs to be analyzed before deployment as described in the next section.

3) *Analyzing the Security of our Logging System:* Ideally, these analyses needs to be completed before deploying our logging system. We can directly use the logging system architecture, Fig. 4, as a tool for our analyses. According to Fig. 4, to achieve the security analysis of the logging system this security relevant question need to be answered. The question is: where the F3 will be located. From the architecture, it will be located in disk0. Our discussion is limited to only this question as an example of this analyzing. The security analysis of the logging system directly involves

the answer above. One can analyze the privacy of F3 because it is in disk0, which is a part of hw0/dom0 (as described in Table I) and is physically owned by a provider. Therefore, the provider may maliciously learn about, or alter F3, which is a privacy violation to a customer. Rocha and Correia [17] demonstrate how this kind of attack can obtain confidential data from a customer domU. This issue needs to be addressed before deploying F3. However, the solution for this issue is out of the scope of this paper.

E. To Detect Other Forms of Mis-behaviours of Threat 1

It is possible to detect some other forms of attacks enclosed in threat 1 such as domUs that host downloads for illegal software as stated in the CSA report [6]. This can be achieved by detecting the process that downloads an unacceptable certain amount of incoming network package. It is possible to detect the downloading process's mis-behaviours as we do with the spam process. Although dealing with all possible forms of threat 1 needs more research, this case study shows that it is possible to detect process's mis-behaviours and then record them as log files to be used as evidence to identify malicious use of domUs.

IV. EVALUATION AND DISCUSSION

A. Evaluation of the Template

The template's goals were already discussed in the beginning of Section III. We evaluate the template against these goals, and also provides many discussions of the template (in Section IV-B, IV-C, IV-D), which affect many aspects of logging systems in IaaS, as well as PaaS. **To satisfy the first goal**, we demonstrate how to instantiate a logging system to mitigate the risks associated with CSA threat 1 in Section III-D1, III-D2, and III-E. This is an example of how the template can be used to instantiate logging systems that are capable of mitigating risks associated with all CSA threats. **To satisfy the second goal**, we discussed how to analyze the security of the new designed logging system architecture in the same case study in Section III-D3. Then we found out this logging system's privacy issues that need to be addressed before deployment.

We also consider the the template as 'generic' because of these considerations. 1) Ones can choose the components (especially Px and Fy) from the template to instantiate their new logging systems architecture. 2) Then, the security of the new logging systems can be analyzed before deployment using the systems' architecture (from 1) as a tool. 3) Thus, any logging system that deals with any CSA threat will follow step 1-2 as same as we did in the spamming case study. 4) The security of any existing logging system in IaaS can be mapped on the template, then the system's security can be analyzed before deployment as discussed in Section III-C.

B. Discussion of the Template

1) Any Other Locations of Px and Fy in the Template?:

We identify and describe all possible and important locations of Px and Fy of the template in the IaaS as mainly discussed in Section III-B. However, basic understanding of logging systems and the template in the Xen-based IaaS layers could apply to more complicated-based layers such as the layers discussed in [4], [14]. These complicated-based layers can indicate new locations of Px and Fy apart from the locations which are presented in our template.

2) *Traditional Log Files Versus Fy in the Template:* This template shows the possible logical and physical locations of Fy. These locations can affect many security concerns of logging systems as discussed in Section III-B and III-D3. We did not propose a new approach to enhance fundamental properties of log files (e.g., non-repudiation or tamper evident). There is research such as [15] that outline or propose these properties (which can be applied to implementation of Fy). For example, to deal with the tamper evident property, Fy can be implemented by reusing existing logging file systems such as the research done by Crosby and Wallach, which introduces the semantics of tamper-evident logs using a tree-based data structure [15]. Therefore, it could be possible to efficiently reuse existing Fy when building a new logging system, instead of building a whole new log file system. This should make a logging system reliable, its security analyzable, and its development and deployment faster, flexible, rapidly adaptable in the IaaS real world.

C. How Does the Template Also Facilitates Analyzing of the Security of Logging Processes (Px)

We already discussed how the template facilitates the security analysis of Fy in Section III-D3. We also argue that the security of a logging process itself (such as P1 in the spamming case study, Fig. 4) has to be systematically analyzed before deployment as well. For example from Fig. 4, the simple security relevant question is how can customers ensure the integrity of P1, which is run by the provider in dom0's user level. Locating P1 in dom0's user level is a security risk because the providers/insiders may maliciously modify P1's code to produce contents of log files (F3) which benefit themselves.

D. The Other Possible Advantages of the Template

1) *Performance Analysis of Logging Systems Which are Based on the Template:* This paper presents only example of how to analyze the security of a new logging architecture itself before deployment, in Section III-D3. However, in the complex environment IaaS, performance of the new built logging system could also be a critical factor that needs to be thoroughly analyzed before deployment. For example, from the template (Fig. 2), the performance of a logging system that deploys P2 in dom0 kernel, and of one that deploys P3 in domU kernel, should be different and needs to be

considered before deployment. With the locations of Px in the newly built logging system architecture, made clear by using the template, the system's performance analysis could be feasible. We believe that the template may be seen as a preliminary study to really achieve a complete analyzing of the newly built logging systems, based on all possible aspects, such as security and performance.

2) *Flexibility and Reuse-ability of Development of the Log Files (Fy) and the Logging Processes (Px):* We argue that Fy can be implemented by reusing existing log files as partly discussed in Section IV-B2. Fy can also be implemented and based on provenance concepts which become increasingly important as discussed in [18]. We already demonstrated reusing available logging processes (libVMI) as P1 that assists building the logging system in the case study, Section III-D1. This is possible because the template is implementation independent. Moreover, it presents the logical and physical locations of Px and Fy in the IaaS structure which one can choose these components to build up their own logging system architecture. These flexibility and reuse-ability enable the development of logging systems faster, and more adjustable as will be fully discussed in the next section. Thus, we are always ready to deal with new emerging threats in the real world IaaS.

3) *The Template Facilitates Faster Logging System Development With Security Concerns:* There is some work focusing on to implement Fy such as [15] as discussed in Section IV-B2. Moreover, other works also focus on implementation of Px such as all that are discussed in the related work section. This paper did not focus on creating a new Px and/or Fy. We encourage ones to build a logging system by following these two steps. 1) The design phase, they can instantiate a logging system architecture by choosing the appropriate Px and Fy (from the template), which fit their needs. 2) The implementation phase, they can reuse the existing Px and/or Fy systems (if any) that fit the designed architecture in 1). Thus, we argue that one can build her own logging systems based on the template with the security concerns of the systems themselves and reuse-ability. However, it is also possible to create her own versions of logging systems without reusing, but the systems can be still based on the template as well. As result, all participating parties can analyze how critical components (Px and Fy), of a newly built system, are distributed in the IaaS visualization environment, which affects the security of this system.

4) *Applying the Template to Mitigate Risks Associated with CSA Threats for PaaS:* Although this paper focuses on dealing with the CSA threats to IaaS, these threats are for the security of PaaS and SaaS as well [6]. PaaS can always be built up by adding extra layers on top of IaaS layers as discussed in the end of Section II-A. Therefore, comprehension of our proposed template and of logging systems in IaaS could also assist to mitigate risks associated with the CSA threats applicable to the security of PaaS. For

example, it is possible that ones can add the extra layers (e.g., in [4], Integration&Middleware layer can be added on top of IaaS to build up PaaS) on top of our proposed template to build up the template for PaaS. Then they can consider the possible and appropriate locations of Px and Fy in this new PaaS template. These new locations of Px and Fy in PaaS architecture could affect their security concerns as same as we discussed our template for IaaS in this paper.

V. RELATED WORK

TrustCloud [9] is a framework proposed by HP to address the lack of trust in the cloud. Its most important layer is the system layer. It is a foundation to build other layers, and deploys Flogger [10] as its core. Flogger is an interceptor that can be placed in dom0 kernel or domU kernel to intercept file and network operations of that domain. Haeberlen et al. proposed Accountable Virtual Machines (AVMs) [12] that detect mis-behaviors of online gaming servers by using a modified hypervisor (VMware) to record all messages sent and received by an untrusted server. The main process of this work is a logger that is in the hypervisor to record incoming and outgoing network packets of domUs. PASSXen [11] is an approach to collect system-level provenance of domUs that run under Xen. PASSXen's interceptor in domU's kernel tracks and collects the creation, access, and destruction of processes and files this domU. [19] proposes a network monitoring application which identifies which process inside a Windows domU is responsible for malicious network traffic leaving this domU. The last work ([13]) is a demo monitoring program in dom0 that outputs all file/directory creation/removals happening in domU's /root directory.

VI. CONCLUSIONS

Despite its lack of trust, IaaS is a base for other types of cloud, and is increasingly used by individuals and organizations. Accountability/monitoring, which composes of logging systems as a core, is necessary to build trust. However, there is not much research that discusses how to analyze the security of such logging systems themselves before deployment in the IaaS environment. This paper investigates all CSA threats. It points out the importance of accountability/monitoring, its critical logging components, and security analysis of these components before deployment in the IaaS. We propose a generic logging template, and describe its goals and components for the purposes of mitigating risks associated with all CSA threats. Finally, we provide two case studies of how to use the template for security analysis of logging systems. The template and the case studies can be a starting point to build and deploy systematic logging systems in the IaaS with security or performance concerns of the systems to truly mitigate the risks associated with all CSA threats. This allows us to deal with the trust issues for IaaS, which will benefit both the customers and the providers.

REFERENCES

- [1] 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*, vol. 53, no. 4, pp. 50–58, Apr. 2010.
- [2] R. De Paris, "Remi-a middleware to handle molecular docking simulations of fully-flexible receptor model in hpc environment," Master's thesis, 2012.
- [3] NHS, "Genetics 'cloud' to create new opportunities for researchers and clinicians," 2011. [Online]. Available: <http://www.manchester.ac.uk/aboutus/news/display/?id=7371>
- [4] CSA, "Security guidance for critical areas of focus in cloud computing version 3," Tech. Rep., November 2011. [Online]. Available: <https://cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf>
- [5] W. Dawoud, I. Takouna, and C. Meinel, "Infrastructure as a service security: Challenges and solutions," in *Informatics and Systems*, 2010, pp. 1–8.
- [6] CSA, "Top threats to cloud computing, version 1.0," Tech. Rep., 2010. [Online]. Available: <https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf>
- [7] 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 *ACM workshop on Cloud computing security*, 2009, pp. 85–90.
- [8] A. Haeberlen, "A case for the accountable cloud," *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 2, pp. 52–57, 2010.
- [9] R. Ko, P. Jagadpramana, M. Mowbray, S. Pearson, M. Kirshberg, Q. Liang, and B. S. Lee, "Trustcloud: A framework for accountability and trust in cloud computing," in *Services (SERVICES), IEEE World Congress*, 2011, pp. 584–588.
- [10] R. Ko, P. Jagadpramana, and B. S. Lee, "Flogger: A file-centric logger for monitoring file access and transfers within cloud computing environments," in *International Conference on Trust, Security and Privacy in Computing and Communications*, 2011, pp. 765–771.
- [11] M. C. Peter Macko and M. Seltzer, "Collecting provenance via the Xen hypervisor," in *3rd Workshop on the Theory and Practice of Provenance*, 2011.
- [12] A. Haeberlen, P. Aditya, R. Rodrigues, and P. Druschel, "Accountable virtual machines," in *USENIX conference on Operating systems design and implementation*. USENIX Association, 2010, pp. 1–16.
- [13] B. Payne, M. de Carbone, and W. Lee, "Secure and flexible monitoring of virtual machines," in *Computer Security Applications Conference*, 2007, pp. 385–397.
- [14] J. Spring, "Monitoring cloud computing by layer, part 1," *IEEE Security&Privacy*, vol. 9, no. 2, pp. 66–68, 2011.
- [15] S. A. Crosby and D. S. Wallach, "Efficient data structures for tamper-evident logging," in *Proceedings of the 18th USENIX Security Symposium*, 2009.
- [16] B. Payne, "VMIttools, virtual machine introspection tools." [Online]. Available: <http://code.google.com/p/vmitools/>
- [17] F. Rocha and M. Correia, "Lucy in the sky without diamonds: Stealing confidential data in the cloud," in *IEEE/IFIP on Dependable Systems and Networks Workshops*. IEEE Computer Society, 2011, pp. 129–134.
- [18] J. Lyle and A. Martin, "Trusted computing and provenance: Better together," in *Proceedings of the 2nd Workshop on the Theory and Practice of Provenance*. Usenix, 2010.
- [19] B. Dolan-Gavitt, B. Payne, and W. Lee, "Leveraging forensic tools for virtual machine introspection," School of Computer Science, Georgia Institute of Technology, Tech. Rep. GT-CS-11-05, 2011.