DEALING WITH ROGUE VIRTUAL MACHINES IN A CLOUD SERVICES ENVIRONMENT

B. Colbert and L. M. Batten
Deakin University, 221 Burwood Highway, Burwood, Victoria 3125, Australia

Keywords: Virtual machine, Rogue machine, Attack, Security.

Abstract: In current cloud services hosting solutions, various mechanisms have been developed to minimize the possibility of hosting staff from breaching security. However, while functions such as replicating and moving machines are legitimate actions in clouds, we show that there are risks in administrators being able to perform them. We describe three threat scenarios related to hosting staff on the cloud architecture and indicate how an appropriate accountability architecture can mitigate these risks in the sense that the attacks can be detected and the perpetrators identified. We identify requirements and future research and development needed to protect cloud service environments from these attacks.

1 INTRODUCTION

Over the last few years, cloud computing has grown from being a promising business to one of the fastest growing segments of the IT industry (Gartner, 2009) because businesses can now gain access to the latest concept and best business applications while, in most cases, minimising the cost of hardware/infrastructure. This is achieved by a client of a cloud services provider being able to share computer platforms, networking and storage infrastructure with the provider's other clients.

However, the downside of this arrangement is that the onus is on the cloud service provider, not the client, to ensure that the particular client's confidential company information is kept appropriately separated and secure, and to take measures to protect that client's company information from malicious software and processes that may be initiated on any of the cloud's virtual machines. Consequently, there is a significant, and growing, level of concern about the cloud and its security (Ristenpart et al. 2009).

One of the security issues which became evident in outsourcing business models, and which is now seen in the cloud services environment, is the trust of staff (Cattedu et al., 2009). In current hosting solutions, various mechanisms have been developed to minimize the possibility of hosting staff from breaching security such as physical separation and logging and audit controls. However, while functions such as replicating and moving machines (often done to optimize operations and efficiencies) are legitimate actions in clouds, we shall show that there are risks in administrators being able to perform them, including the potential to:

- *impersonate clients,*
- *expose confidential data,*
- *corrupt data, and*
- *make fraudulent transactions.*

In this paper, we describe three attacks on the virtual machine layer of a typical cloud services architecture which can be perpetrated by hosting staff, resulting in all of the above risks being effected.

We describe requirements on the cloud architecture which allow partial mitigation of the above risks in the sense that the attacks can be detected and the perpetrators identified. These requirements include logging of access events, staff profiling and machine profiling. While some of these requirements are not yet in use in cloud service environments, we believe that research and development over the near future will enable the gaps to be filled.

In the next section we discuss current work in this area. In Section 3, we describe our attacks scenarios. In Section 4 we analyze these scenarios and describe how they can be mitigated. In Section 5, we conclude and propose further work.

2 RELATED WORK

Cloud service systems are dynamic and require organizational collaboration across many sectors, including business services, administration and technical services. Thus, predefined business logic and service level agreements are normally in place to enable smooth business processing. This, in itself, is not sufficient to ensure that the environment can be completely trusted. Violations can occur. As pointed out in (Yao et al. 2010), the ability to detect violations is a critical component of developing a trusted computing environment. They argue that this ability requires a separation of activities and evidence logging into different domains which they call the 'business service domains' and the 'accountability domains' respectively. This is now a commonly accepted approach to cloud architecture (Yao et al. 2010), (Popa et al. 2010).

Yao et al. argue that the purpose of accountability is 'binding each activity to the identity of its actor' and thus demonstrate a method which is shown to do this for a loan-service scenario implemented on the Amazon Elastic Compute cloud (http://aws.amazon.com/ec2/).

Dalton and colleagues in (Dalton et al., 2009) argue that reducing the size and complexity of the code which needs to be trusted is the best approach to providing security and trustworthiness. Figure 1 provides a standard description of the use of a hypervisor in a virtual machine environment. The hypervisor is a layer of software that sits between hardware and higher-level software structures such as applications, and decides which of them gets how much access to memory, storage and CPU time. In the top of the figure, several virtual machines, software versions of PCs, are available for use by customers, as in for instance an online shop. The chassis supplies the power to the blade server which has its own CPU, memory and hard disk. In (Dalton et al., 2009), the authors disaggregate the hypervisor, identifying 'the security critical features of the platform and moving them into separate and smaller restricted privilege domains' and develop in the paper an architecture around an interface with the disaggregated hypervisor (see Figure 3 of their paper). The authors claim that this new architecture, along with a trusted platform module allows for better integrity measurement and reporting, though they do not report on any experimental work.

In work of Corney, Mohay, Clark and Lopes (Corney et al, 2011) prototype software was developed which identifies anomalous events in a computing environment based on usage patterns and user profiles. The authors are able to detect unauthorized use of software by users in an organization. They do this by building a user profile for each user and detecting departures from the routine. At this point, their work is based on the presence or absence of a particular application in a user's file, but they plan to extend the work to build probabilistic models. In Section 5, we indicate how this type of profiling can be a solution to stopping the attacks with which we deal in this paper.

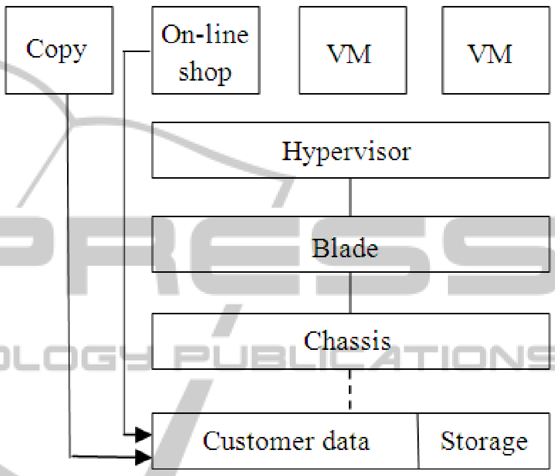

Figure 1: An on-line shop is copied.

3 THE ATTACK SCENARIOS

We consider the hardware set-up of the virtual machine environment. Figure 1 presents this in detail. A number of virtual machines run off a hypervisor supported by the blade and chassis. Data gathered from the virtual machine is sent to the storage area once it has been processed.

We assume, as in (Yao et al. 2010), that the Business Services Domain is separated logically from the Accountability Domain and that no staff in either domain are able to perform the functions of the staff in the other domain.

The following three sub-sections describe attacks on this set-up.

3.1 Unauthorized Activities

A cloud staff member may abuse the privilege of accessing the storage area and gain access to information which is an abuse of their privilege status. This individual may also run unauthorized software or applications on machines in the Business Services Domain. We describe in Section 5 how the cloud architecture can detect these abuses and identify the individual.

3.2 VM Machine Copying

In order to handle peak demands on the system, it is standard practice to make copies of virtual machines in order to deal with the excessive traffic. For example, an on-line shop may experience a peak in business during lunch-time hours from Monday to Friday. A copy of the shop can be made to deal with the peak hours and then closed down when no longer needed.

A business service staff member therefore, legitimately, has both the ability and right to establish a copy of a virtual machine, as shown in Figure 1. The copied virtual machine has the same capability and data as the original and can be used to impersonate clients, fraudulently purchase goods using customer credit card information, corrupt credit card information with the aim of denying a customer the opportunity to purchase, for example, or gather credit card information for the purposes of selling it.

3.3 A Rogue VMware Machine

A cloud staff member may be authorized to replace a mal-functioning machine, but may introduce a rogue machine to the stack, substituting it for a legitimate machine. This machine may then access the storage and steal or corrupt customer data. It may carry malware which can infiltrate the hypervisor and so the other virtual machines in the stack, stealing or corrupting data, playing havoc with the network and connections and business processes.

We argue that the Accountability Domain must be able to deal with these situations in a number of ways.

1. It must be able to detect these violations.
2. It must be able to determine the exact time at which the violation occurred.
3. It must be able to determine which staff were responsible.
4. It must be able to differentiate between the scenarios 3.2 and 3.3.

In the next section, we analyze these attack scenarios and determine the information needed by the accountability domain in order to provide answers to the above questions.

4 PROVIDING ACCOUNTABILITY

In this section we develop the framework for an Accountability Architecture which will, amongst other things, be able to successfully complete the four tasks delineated in Section 3.2. In doing so, we draw on work of other authors as well as introducing some novel components.
We first define some of the terms we shall use in this section.

Definition. An ‘event’ is an action initiated either by the user or the computer. An example of a user event is a keystroke. An example of a computer generated event is a notification based on the time of day

Definition. An ‘enactor’ is the entity generating an event. This might be a computer or a user.

4.1 Binding Events to Enactors

Following Yao et al., we stipulate that each event is bound to its enactor. This is done in the Business Services Domain by administrative means. There is a set list of events (with associated times) available for execution. Staff members are assigned to enact these events at specific times.

When staff member S enacts event E, for example instantiating or copying a virtual machine, she signs a set of input parameters to the event and a set of output parameters from the event with her private key from the company’s public key scheme. These are then lodged with the Accountability Domain which can verify the enactor by means of her public key which produces information connected to the event.

For actions, such as loading software onto virtual machines, or reconfiguring the environment, change management processes and approval structures should be in place to provide further accountability for these actions.

4.2 Logging of Events

Each event is automatically logged by the Accountability Division along with the identity of the enactor in the form of the two signed data-strings, and the time the signing occurred. At the time of logging, the identity of the enactor is confirmed and also logged in the Accountability Domain.

We note that ensuring the integrity of the logs themselves has been considered by several people. For example, (Schneier et al., 1999) use hash chains to ensure that logs cannot be tampered with.

4.3 Staff Profiling

Here, we introduce a methodology based on ideas in (Corney et al. 2010). The Accountability Domain starts to build a profile of each staff member in the Business Services Domain as of their date of appointment. All events enacted by each staff member are logged in the profile. Each profile is then regularly checked for anomalies and this process should be automated in some reliable fashion. Nevertheless, the amount of data needing to be checked can grow quickly into quantities that distract from the core business. This can be managed very well by means of a sliding window approach, as demonstrated in (Corney et al. 2010). In this method, while all data in the profile is stored, only the last n hours are checked on a regular basis. The number, n, of hours chosen may be a function of level of access assigned to the staff member, level of confidentiality of data accessed during the event, or any subset of a number of additional items. We assume that for each staff member S, the number n_S of hours checked on a regular basis is pre-determined jointly by the administration of the Business Services Division and the Accountability Division. The number n_S may change from time to time relative to the same staff member S and depending on changes in the role and responsibilities of S.

4.4 An Accountability Architecture

We now combine the components discussed in Sections 4.1, 4.2 and 4.3 into an accountability architecture which is exhibited in Figure 2.

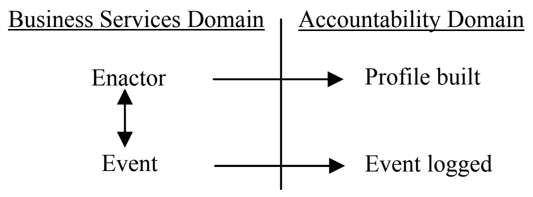

Figure 2: Accountability architecture.

In Figure 2, the enactor and event are represented as being bound by means of the private key of the enactor which signs the event. The event is logged and, at the same time, the fact that the enactor performed the event is recorded in her profile.

5 DEALING WITH THE ATTACK SCENARIOS

We now explain how the Services Accountability Architecture, introduced in the previous section, can deal with the situations posed in Section 3.

5.1 Detecting Unauthorized Activities

As discussed in Section 4, the practice of logging the actions of privileged users is necessary to provide accountability for staff. While standard logging procedures may be in place, the diagnostics of such logs in this environment need development in order to deal with mass production and to deal with dynamic features of virtual machines. To our knowledge, there is no current research on the most appropriate or the most accurate diagnostic methods in such an environment; nor is there any indication that diagnostic methods used in past service provider environments are appropriate for cloud services.

However, once such research has been accomplished, the Accountability Domain will be able to examine the activities of all virtual machines and correlate them to determine which were unauthorized and when they occurred, as well as which staff were involved. On the other hand, we note in the next two sub-sections some problems which need to be understood when developing log diagnostics to determine the time and the staff member responsible.

5.2 Determining the Exact Time of a Violation

Consider the situation where a virtual machine is shut down under certain conditions; this may include the case where the number of requests falls below a certain threshold for example. Precise conditions such as these would be established in the working environment. However, if a staff member uses a machine as a false copy of a legitimate machine, request rates may remain high enough to maintain machine operation. In such a situation, technically, a violation occurred at the time at which the flow of legitimate requests fell below the pre-determined threshold. But the system cannot distinguish between legitimate and illegitimate requests; it is only at the time that the machine is discovered to be a copy that the system logs will be invoked to determine when the copy was made.

5.3 Determining the Staff Member Responsible

Once the time of a violation has been determined, the profiles of all staff on duty at that time can be examined. Events logged at that time identify the staff member or members associated with the event by means of the private key used. The events related to the violation then indicate which staff were involved.

5.4 Differentiating between Violations 3.2 and 3.3

Unauthorized actions, such as accessing storage, will either come from a virtual machine in the environment or from a management console. Access from management consoles and virtual machines is logged and so detectable by examination of the logs.

Other types of attacks may come from virtual machines however, and their detection relies on suitable logging actions. Machine copying, for example, can be detected if the machine acts in an unusual way. A machine profile is an appropriate method of detecting a copy, since a machine which has been functioning in the cloud service environment performs particular, specified tasks (such as those relevant to an online shop) and its operation follows a regular pattern over a particular block of time (one day or one week for example). A copy of such a machine is expected to follow the same pattern. Any departure from such a pattern would be suspicious.

A rogue machine introduced as a new piece of hardware as described in Section 3.3, would not have established a usable profile with which to compare behaviour. However, unlike the situation of a copied machine, which would try to remain undetected, a rogue machine would have negative impact on other machines in its environment. In this case, it is the profiles of the affected machines that would detect anomalies, and the collection of these anomalies would point to a rogue machine.

In summary, appropriate logging methods in conjunction with staff and machine profiles enable the cloud service provider to deal with any of the attacks discussed in this paper, either individually or combined. Methods of ensuring tamper-proof logs as in (Kelsey et al. 1999) can be extended to ensuring that profiles are also tamper-proof.

6 FUTURE WORK

Simulation of authorized and unauthorized activities in large production domains will provide direction for the development of efficient and accurate log-based diagnostics in cloud environments.
The establishment of profiles for individual staff members is extremely useful in identifying anomalies in the individual's behaviour which may be indicative of suspicious behaviour. Such profiling can be extended to sections of the cloud operation such as a particular business process or a particular machine or cluster of machines in order to detect a situation which may indicate malicious behaviour. Future work in this area might include the use of techniques currently used to detect a denial-of-service attack on a server from outside the firewall.

The gathering of data in an on-going automated way for potential forensic use is a second area of research which is worthwhile pursuing. This would have to be done in such a way that the rights and privacy of staff members were not violated (with respect to all relevant laws) and also such that data was gathered efficiently and in a manner that was easy to access and use when needed.

The ability of a cloud customer to detect the failure of conditions in the service level agreement would be a useful addition to assurance of data integrity. This idea has been proposed in several publications, for instance, in (Simmhan, 2010) and it is an area of interest for further work by the present authors.

REFERENCES

Cattedu, D. and Hogben, G., editors. 'Cloud computing security benefits, risks and recommendations', Nov. 2009 Report by the *European Network and Information Security Agency*.

Corney, M., Mohay, G., Clark, A., R., Lopes, J. 2011. Detection of anomalies from user profiles generated from system logs. In *Proceedings of AISC; to appear*.

Dalton, C., Plaquin, D., Weidner, W., Kuhlmann, D., Balacheff, B. and Brown, R. 2009. Trusted virtual platforms: A key enabler for converged client devices. In *ACM SIGOPS Operating Systems Review*, vol.43, 36-43.

Gartner Inc. 'How cloud computing will change business' in www.businessweek.com/print/magazine/content/09_24/b4135042942270.htm

Popa, R., Lorch, J., Molnar, D., Wang, H., Zhuang, L. 2010. Enabling security in cloud storage SLAs with CloudProof. Microsoft report available at http://research.microsoft.com/pubs/131137/cloudproof-msr-tr.pdf

Ristenpart, T., Tromer, E., Shacham, H., & Savage, S. (2009). Hey, You, Get off of my Cloud: Exploring Information Leakage in Third

Schneier, B., Kelsey, J. 1999. Secure audit logs to support computer forensics. In A*CM Transactions on Information and System Security*, 2, 159-176.

Simmhan, Y. and Gomadam, K. 2010. Social web-scale provenence in the cloud. In *Proceedings of IPAW 2010, LNCS* vol. 6378, pp 298-300.

Yao, J., Chen, S., Wang, C., Levy, D. Zic, J. 2010. Accountability as a service for the cloud. Proceedings of *IEEE International Conference on Services Computing, IEEE Computer Society*, 81-88.