Logging System for Cloud Computing Forensic Environments

Alecsandru Pătrașcu, Victor-Valeriu Patriciu

Military Technical Academy, Computer Science Department
Bucharest, Romania (e-mail:alecsandru.patrascu@gmail.com, victorpatriciu@yahoo.com).

Abstract: Cloud computing represents a rather new technology and a different paradigm in the field of distributed computing that involve more and more researchers. We can see in this context the need to know exactly where, when and how a piece of data is processed or stored. Compared with classic digital forensic, the field of cloud forensic poses a lot of difficulties since data is not stored on a single storage unit and furthermore it involves the use of virtualization technologies.
In this paper we will present in detail a new and novel way of monitoring activity in cloud environments and datacenters using a secure cloud forensic framework. We talk about the architecture of such framework and how can it be applied on top of new or existing cloud computing deployments. Also, for testing and results collecting we have implemented this solution to our previous developed cloud computing system.

Keywords: cloud computing; data forensics; logging framework; distributed computing; binary diff.

1. INTRODUCTION

Since its creation, the cloud computing technology presented itself to the users as a way in which they could rent various amounts of computing power under the form of virtual machines, intermediate platform targeted to developers or ready to use applications for mass usage. The technologies surrounding it have evolved with great pace, but nevertheless, we can find a single concern in all of them - cloud computing security. Also, the need of knowing how the information is delivered from and to the clients and under what condition is it processed is emerging alongside with the security issues.

In this context, cloud computing has become in the last years a paradigm that attracts more and more researchers. One of the main research areas in this field is the way in which common data and processing power can be shared and distributed across single or multiple datacenters that are spread across a specific geographical area or even the entire globe. A new need for IT experts is increasing: the need to know exactly how, where and in what condition is the data from the cloud stored, processed and delivered to the clients. We can say with great confidence that cloud computing forensics has become more and more a need in todays distributed digital world.

In case of classic computer forensics, the purpose is to search, preserve and analyze information on computer systems to find potential evidence for a trial. In cloud environments the entire paradigm changes because we don't have access to a physical computer, and even if we have access, it is a great chance that the data stored on it is encrypted or split across multiple other computer systems.

In order to resolve these fundamental problems, researchers have developed over the years various technologies. Systems such as *Host-based Intrusion Protection Systems* (HIPS) or *Network-based Intrusion Protection Systems* (NIPS) have a single point of view at their core: making network penetration hard.

But this is not enough in our current digital world, as we store more and more data remotely, in cloud systems. Hackers, malware and all other Internet threats are real menaces to our data. Thus, legal investigators must have a way in which they can monitor the activity of a certain virtual machine. The problem that they face in this case is mainly concerning jurisdiction, as the cloud data is often split across multiple datacenters, over multiple countries or even continents. On second case, current cloud infrastructures tend to leave this sensible part away and only monitor virtual machines for performance rather than what is happening inside them.

Taking in account all the variables that have appeared in cloud computing technologies, modern hypervisors and virtualization technologies implement more or less simple mechanisms for data monitoring across datacenters. Starting from the basic building blocks composed by simple logs that are gathered from the entire cloud infrastructure, every monitoring module must have a precise target and must not affect the proper function of the systems from the datacenter. All virtualization technologies have a difficulty in this area.

Because of the reasons explained so far, a fair observation can be made: Cloud Computing is a new, raw and tough research domain because it involves knowledge from many other domains like Distributed Computing, algorithms and data structures, networking protocols and many other, in order for it to function properly. The main concerns that are rising are influenced by the transition from the regular basic server infrastructure held by users to a centralized datacenter maintained by a third-party provider.

In this paper we are going to present a new and novel way in which we can integrate a full forensics framework on top of a new or existing cloud infrastructure. We will talk about the architecture that stands at it ground and we will present its advantages for the entire cloud computing community. We will present also the impact that our technology proposal will have on existing cloud infrastructures and as a proof of concept we will present some particular implementation details over our own cloud computing framework that we have already developed in (Patrascu *et al*., 2012). Of course we are not neglecting the security part of our proposal and we will present briefly a mechanism that helps us secure the transmissions across our cloud infrastructure modules.

The rest of the document is structured as follows. In section 2 we present some of the related work in this field, that is linked with our topic and in section 3 we present in detail our proposed cloud forensics logging framework. Section 4 presents how our framework can be installed inside a datacenter and what modifications can be made in order to improve the overall performance without interrupting the normal activity of the datacenter. Section 5 is dedicated to presenting our results from our implementation made so far, and in section 6 we conclude our document.

2. RELATED WORK

In the field of classic incident response there are is a lot of active research and many books, guides and papers. Nevertheless, in the field of cloud computing incident response the papers are mostly theoretical and present only an ideal model for it.

In the direction of classic incident response, one of the most interesting guides is the one from NIST. In it we can find a summary containing a short description and recommendations for the field of computer forensics, along with the basic steps that must be made when conducting a forensic analysis: collection, examination, analysis and reporting. A great deal of attention is paid to the problem on incident response and how should an incident be detected, isolated and analyzed.

Bernd Grobauer and Thomas Schreck talk in (Grobauer *et al*., 2010) about the challenges imposed by cloud computing incident handling and response. This problem is also analyzed in (Chen *et al.,* 2012), where they consider that incident handling should be considered a well-defined part of the security process. Also it is presented a description for current processes and methods used in incident handling and what changes can be made when migrating towards a cloud environment from the point of view of a customer or an security manager.

Furthermore, the integration of cloud incident handling and cybersecurity is presented in two papers, one written by (Takahashi *et al.,* 2010) and the other written by (Simmons *et al*., 2012). They talk about how Internet development leads to a widespread deployment of various IT technologies and security advances. In their paper they also propose an ontological approach for integration of cybersecurity in the context of cloud computing and present how information should be used and analyzed in such environments.

The field of cloud logging, as a support for forensics, is also starting to emerge along with the ones presented before. In this directions, we find thesis, such as the one of Zawoad et al which present in (Zawoad, 2013) an architecture for a secure cloud logging service. They talk about the need of log gathering from various sources around the datacenter or hypervisors in order to create a permanent image of the operations done in a datacenter and they present an architecture that can be used to help in this direction.

The same challenges are evidenced by (Marty, 2011; Sibiya *et al*., 2012). The paper discusses a logging framework and presents a series of guidelines that can provide assurance to forensics investigators that the data has been reliably generated and collected and propose a standardized way to do logging, in order to have a single and centralized logging collector and processor, thus saving time and money for both businesses and users.

Also, as can be seen in papers such as (Amarilli *et al*., 2011; Atanasiu *et al*., 2012), the variety of applications that include logging and that can be used in case of a network is large and include also tools for reverse engineering and obfuscation detection and prevention. This is the main reason that forensic investigators must follow a standard set of procedures: after physically isolating the targeted computer, so that its data cannot be accidentally altered, they make a digital copy of the hard drive. Once the drive has been copied, it is put in a secure storage facility to maintain it in proper conditions. All of our investigation is done on a digital copy of the original data.

During our research we have focused on choosing an intermediate representation of data that is sent between the local and central forensic modules. We have analyzed different existing metalanguages for logging.

The first one is the "Management metalanguage Specification" proposed by the UnixWare community. Its advantage is that it can be used as a transparent API in the kernel modules as it provides an interface for an external host. The downside is that it needs a lot of auxiliary binary data to be sent in order to re-create the entire picture at the other end, and using it we get quickly traffic larger than the one that can be obtained by sending only the basic snapshots. This is due to the fact that this metalanguage is designed to be used only locally over a system.

On the other side, the CEE (Common Event Expression) organization proposes a set of specifications using the JSON and XML markup languages for event logging on disk or in transit over a network. These requirements are designed for maximum interoperability with existing event and interchange standards to minimize adoption costs. The advantage of this approach is that CEE expresses its interfaces and does not promote an actual implementation.

Using the information gathered at this step and taking in consideration that our project is novel in this field, we will provide our own logging metalanguage, based on the CEE

specifications for compatibility. Due to advantages it offers, we are going to use the JSON markup language as data envelope because it is a simple, clean, concise and human-readable format. It is also good for decoupling our cloud forensic modules because we can implement each module using its own programming language and having only a common JSON interface for using it. Along with this format we intend to use it along with a storage module that is fit for our needs. We have chosen this approach in order to have the collected data from a host in a single database and only provide management messages to the above central forensic modules.

3. LOGGING FRAMEWORK ARCHITECTURE

In this section we will present the top view architecture of a cloud enabled forensics system. We will start with the general concepts and aspects that our system will implement and then focus on the logging part. We will also talk about the system perspective from the forensic investigator part.

3.1 General forensics architecture

The system presented in this paper has a modular architecture and each of the modules is presented in detail. It is easy to see that the entire framework can be extended with other modules or plugins. In order to have a solid working platform, we must first introduce the concept of a cloud computing framework. As can be seen in Figure 1 the top view of a cloud computing framework contains two main layers: the virtualization layer and the management layer.

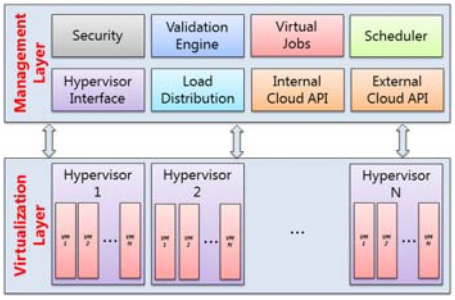

Fig. 1. Basic cloud computing architecture.

In the virtualization layer we find the actual workstations that host the virtual machines and have virtualization enabled hardware. In the management layer we find the modules responsible with enabling the entire operations specific to the cloud, as presented in the previous sections. These modules are:

- *Security.* This module is responsible with all security concerns related to the cloud system. For simplicity we can consider it as an intrusion detection and alarming module.
- *Validation engine.* This module receives requests to add new jobs to be processed. Every new request is checked for consistency and it is validated and if it is legit, the new lease is transformed in a job for our system and it is properly inserted in the job queue.
- *Virtual jobs.* This module creates an abstraction between the data requested by the user and the payload that must be delivered to the cloud system.
- *Scheduler.* This is one of the most important modules in a cloud framework. Its main purpose is to efficiently schedule the jobs to the virtualization layer. It also must communicate with the other modules in order to find new instances, new services, virtual machine managers, load balancers in the system.
- *Hypervisor interface.* This module acts like a translation layer that is specific to a virtualization software vendor. It must implement each vendor API specifications.
- *Load distribution.* This module is responsible with horizontal and vertical scaling of the requests received from the scheduler. It must run a distinct application framework in order to decouple the code from the existing underneath runtime. The algorithm must be applied automatically and in the process of this analysis, the number of workstations must be taken in account.
- *Internal cloud API.* This module is intended as a link between the virtualization layer and the cloud system. In order to be more scalable and also maintain a high degree of abstraction, a common interface must be provided and every implementation of the specific API must implement this.
- *External cloud API.* This module offers a way to the user to interact with the system. It must provide means to add new jobs in the cloud system. The requests are registered and sent to the validation engine module. This API must be flexible enough to permit adding details to the jobs, like the hardware specifications of the virtual machine, operating system to be used, packages to be installed.

Now that the notion of a cloud computing framework was presented, we will talk about the modifications that must be made to it in order to create a forensic enabled cloud computing architecture. As can be seen in Figure 2 the modification affects all the existing modules.

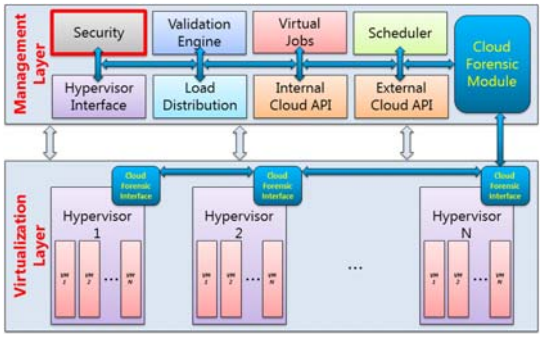

Fig. 2. Forensic enabled cloud computing architecture.

More exactly, we see a new module, the *Cloud Forensic Module*. Its main goal is to gather all forensic and log data from the virtual machines that are running inside the virtualization layer. Furthermore, we must attribute to the security module greater responsibilities and permit it to communicate with all the other modules in the management layer.

Of course, in order to gather data reliably from the virtual machines we must interact with the hypervisors existing in the workstations kernel. In our paper we present only what modifications must be made to a Linux kernel. We have chosen this alternative because in a Linux kernel we can find at least two distinct, free and open source virtualization techniques: KVM and XEN. An image of a forensic enabled kernel can be seen in Figure 3.

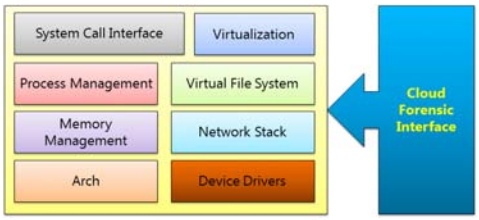

Fig. 3. Forensic enabled kernel.

On our research we will focus on the KVM technology. KVM (Kernel-based Virtual Machine) is a full virtualization solution for Linux on x86 hardware containing virtualization extensions on Intel or AMD processors. It consists of a loadable kernel module called *kvm.ko*, that provides the core virtualization infrastructure and a processor specific module called *kvm-intel.ko* or *kvm-amd.ko*. Using KVM an user can run multiple virtual machines running unmodified Linux or Windows images. Each virtual machine has private virtualized hardware such as a network card, disk and a graphic adapter.

The cloud forensic interface is implemented as a series of stand-alone kernel modules and user-space applications that can be activated or disabled at runtime. Our goal is to provide the users a way to manage it from the kernel building menu. We need this segregation because we want to have access to all the modules from the kernel that helps in the entire process of virtualization. Parts like system calls, process management, virtual file system, memory management, networking management are extremely important to our research because they represent the basic building blocks between a virtual machine running on a host and the operating system. We will detail furthermore the main interest directions for our research.

The first step toward full kernel integration is to have a proper API both to the kernel and to the external system. We are interested mostly about KVM internal API. This API is a set of ioctl instructions that are issued in order to control different aspects of a running virtual machine. In computing, *ioctl* (short for "input-output control") represents a system call made to a specific device which cannot be done using regular system calls. Logically, this API is split across three different main parts: *main system ioctls*, meant to set proper KVM internal variables and used when creating a new virtual machine, *virtual machine ioctls*, meant to set different attributes of a virtual machine, like the memory size or layout and *virtual CPU ioctls*, meant to be used to control the operation of a virtual machine virtual CPU.

In order to be fully compatible with the Linux API, the entire KVM API is centered around the concept of *file descriptors*. This means that once activated, KVM creates a new device called "*/dev/kvm*". On initial open of the device we get a handle of the internal KVM system that we can use to issue proper ioctl commands. For example, sending a *KVM_CREATE_VM* command to the kernel, we get a response containing a virtual machine file descriptor that we can further use to set different values.

It is easy to see that we can get all details concerning about the status of the "virtual hardware" that is used for a certain virtual machine. This is useful because, for example, we can get details such as:

- the memory pages that are dirtied since a last call, using the *KVM_GET_DIRTY_LOG* ioctl and the following data structure:

```
struct kvm_dirty_log {
        {...}
        union {
                void __user *dirty_bitmap;
                __u64 padding;
        };
        {...}
```

- getting processor registry values, using the *KVM_GET_REGS* ioctl and the following data structure:

```
struct kvm_regs {
        {...}
        __u64 rsi, rdi, rsp, rbp;
        __u64 r8,  r9,  r10, r11;
        {...}
```

- setting processor registry values - *KVM_SET_REGS* ioctl;

- translation of a memory virtual address according to virtual CPU own translation mode, using the *KVM_TRANSLATE* ioctl and the following data structure:

```
struct kvm_translation {
        __u64 linear_address;
        __u64 physical_address;
        {...}
```

From the point of view of live cloud computing forensics, a great impact is given by the Memory Management Unit (MMU). For virtualization software it is very important to have a proper MMU module. In our case, KVM uses its own MMU modules with the purpose of translation from guest physical address to host physical address. This gives us a real advantage because interacting with these modules gives us a

full map of what is going on inside a virtual machine memory space.

In order to be considered reliable, a virtual machine MMU must respect a set of particular requirements, like correctness (the virtual machine must not be able to determine that it is running using an emulated MMU) and security (the virtual machine must not allocate memory beyond the limits imposed by the MMU). These requirements are going to be monitored by our forensic module and at any time we will have a full memory footprint and the whole previous states.

It is important for a forensic investigator to have access to the network communication devices and to the storage devices. In case of virtual machines running under KVM it is done using the *virtio* interface. Virtio is a virtualization standard for network and disk device drivers where just the guest's device driver is aware of the fact that it is running in a virtual environment, and cooperates with the hypervisor and this enables guests to achieve high performance network and disk operations. Its design allows the hypervisor to export a common set of emulated devices and make them available through a common API. Using this interface we gain full access to everything related to the disk and/or network devices than an investigator can use.

3.2 Cloud logging architecture

In this section we present how our framework is working and how it is created in order to run on top of new or existing cloud computing infrastructures. As example for it we will present the integration with our previously implemented Cloud Computing framework in (Patrascu *et al.*, 2012).

The architecture is a layered one, containing five layers, each with its own purpose. We will present each of them in detail in the following paragraphs. We also represented it in a graphical way, in Figure 4 the whole layers and the relationship between them. The layers are all implemented using the distributed computing paradigm. Actually, they represent jobs in our cloud computing environment. We preferred this architecture because it is natively scalable on top of existing datacenters or computer networks and can deal with large amount of data and connected clients.

Furthermore, in order to make sure that the data is kept safely and no one will tamper it, each operation made by the system will go through a hashing algorithm, both original, resulting and diff files. These hashes will be stored encrypted using a passphrase provided by the investigator.

The cloud architecture presented in our previous work makes use of the concept of leases, in which we can specify the amount of time the job must run, or specify between what hours in a day it is running. To achieve our goal, we created a lease that during the day, when the datacenter is mostly occupied by the users, uses a minimum number of nodes and during the night, when the datacenter is almost entirely free, it automatically scales up to use a maximum number of nodes.

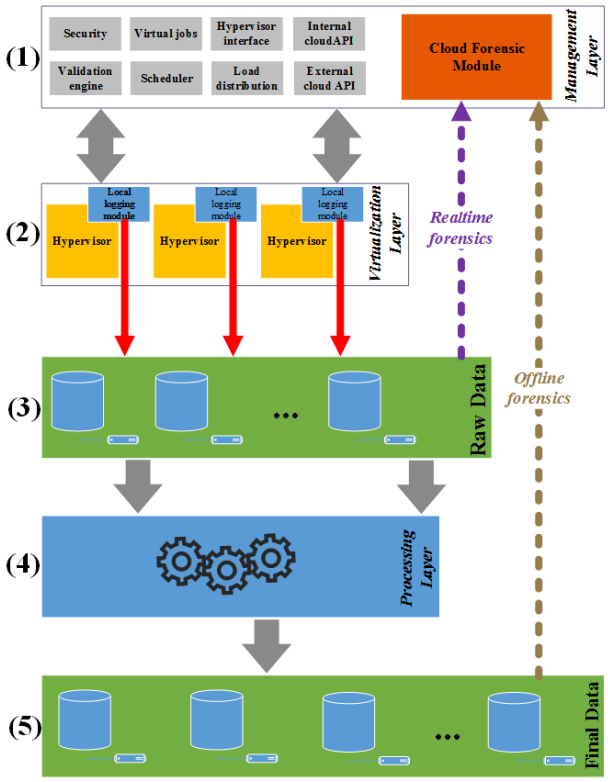

Fig. 4. Cloud Forensics Logging Framework.

3.3 Top view architecture

The first layer, as presented in our previous work (Patrascu and Patriciu, 2013), represents the management layer in a cloud computing deployment. It contains the modules responsible with enabling the entire operations specific to the cloud. We can also see the previously mentioned Cloud Forensic Module.

The second layer represents the virtualization layer in a cloud computing deployment. Its purpose is to contain the actual workstations that host the virtual machines and have virtualization enabled hardware. A dedicated "Local logging module" must be installed into the existing physical machine. It is responsible with the RAW data gathering from the monitored virtual machines. The data quantity can be adjusted by the investigator and he can choose to monitor a particular virtual machine or monitor the entire activity existing inside that machine.

In order to gather data reliably from the virtual machines the local logging module must be integrated fully with the running hypervisor inside the physical machine. In this paper we focus on the integration with the "KVM" virtualization technology that exists in modern Linux kernel releases. We have chosen it because it is a full open-source virtualization solution, integrated with the Linux kernel since 2007 and it is actively used by many companies across the world.

An important thing that must be taken in consideration is what data are we intercepting from the virtual machine and send it to further processing. Since all the activity can be intercepted, there is the risk of severe time penalties and processing speed. In order to solve this problem, at this point we will offer the possibility for an investigator to choose the logging level for a certain virtual machine. This is helpful considering that, for example, an investigator only wants to analyze the virtual memory for its contents, and it is not interested in virtual disk images or virtual network activity. Also at this step we must consider the problem of network transmission overhead.

The third layer represents a storage layer for the RAW data sent from the local logging modules existing in the virtualization layer. The logging modules will send RAW data, in the form they are gathered from the hypervisor. Thus, this layer has the function of a distributed storage and it contains a series of nodes, each running a database. We have chosen this approach in order to create a flexible and scalable layer architecture that can face the data traffic coming from the upper layer.

Since the data that is going to be sent from the physical virtualization host to the central forensic management unit can reach important size, we will implement a mechanism of "diff" between two pieces of data. For example, if an investigator will want to analyze a virtual machine memory over a period, the local forensic module will sent only one initial memory snapshot and after that only what has been changed will be sent. Of course we can use the full potential of the host and provide a local aggregation module that will pre-process the data collected before sending it to the central forensic module. This approach is new to the field of cloud computing forensics and we consider it a great way to reduce the impact over the network.

The process will run in the following manner. Initially the logging modules will send a reference file and then, at a user defined time period, the modules will send a delta file, that represents the difference between the previous reference file and the current state. Thus, it will implement a snapshot mechanism at the hypervisor level. We have chosen this approach because we want to offer to the forensic investigator the possibility to have an image of what is happening inside a virtual machine between two snapshots. This feature is currently not available in other hypervisors, such as VMware's; in their case we can have a snapshot at time t_0 and one at time t_i, but we cannot know the state of the virtual machine between the 0 and i step.

This layer has also another purpose. In case of extreme emergency, the forensic investigator can "see" a real-time evolution of the monitored virtual machine by issuing a direct connection to this layer. This feature is made available through the Cloud Forensic Module, which has the ability to by-pass normal RAW data processing.

The fourth layer has the purpose of analyzing, ordering, processing and aggregating the data stored in the previous layer. Since all these steps are computing intensive, the entire analysis process will be made in an offline manner and will be available to the investigators as soon as the job is ready. After this entire process the investigator will have a full image of what happened over the monitored remote virtual machine in a manner such as the one encountered in software source code version tools, thus permitting him to navigate back and forth into the history of the virtual machine.

This layer is implemented also as a distributed computing application. We have chosen this approach due to the processing power needs that our framework demands; more exactly it needs to do correlations between different snapshots in a fair amount of time.

The fifth layer represents the storage of the results published by the previous layer. A forensic investigator will interact with the monitored virtual machine snapshots at this layer, by using the Cloud Forensic Module from the Management layer.

4. DATACENTER ENABLED ARCHITECTURE

In order to have a proper working forensic and logging system we must pay a great deal of attention to its performance. Since all the activity can be intercepted, there is the risk of severe time penalties and processing speed. In order to solve this problem, we offer the possibility for an investigator to choose the logging level for a certain virtual machine. This is helpful considering that, for example, an investigator only wants to analyze the virtual memory for its contents, and it is not interested in virtual disk images or virtual network activity.

The situation is different in case of implementing our solution in a real datacenter where large-scale parallel computers are the base of high performance computing and it relies on a Ethernet network interconnection that is fast and efficient.

To better understand the issues imposed by our solution we will first present a computer architecture existing in today's datacenters. In Figure 5 you can see a *Fat Tree* topology, as presented in Al-Fares et al (2008), composed from multiple building blocks. The basic building block is called a *cluster*. A cluster is composed from multiple *racks*, each rack having multiple servers and one switch called *top of the rack switch* (ToR). This allows communications between adjacent servers to be made really fast. Using the same principle of data locality, each ToR is linked by a *level 2 switch* (L2S) and each L2S is linked to an *aggregation switch*. Each cluster is linked to a *cluster router* (CR) and finally, each CR to a *border router*.

To better handle data that is going to be processed and transferred, we propose a slight modification of this topology. In the first place, the modifications will start at the physical servers that are stored in each rack. For this, we need a dedicated forensics network port, just like a management port, as can be seen in Figure 6. Also, this port must have a correspondent in the ToR switch. This port will be used by our Cloud infrastructure for collecting and processing data and also it will be used by the authenticated forensic investigators. Using this approach we will make sure that the

network connections between the users and their virtual machines remain untouched and the time penalties will be reduced to a minimum. In this case the only lag the hypervisor will have is the moment in which the forensic system will try to take a snapshot of a virtual machine.

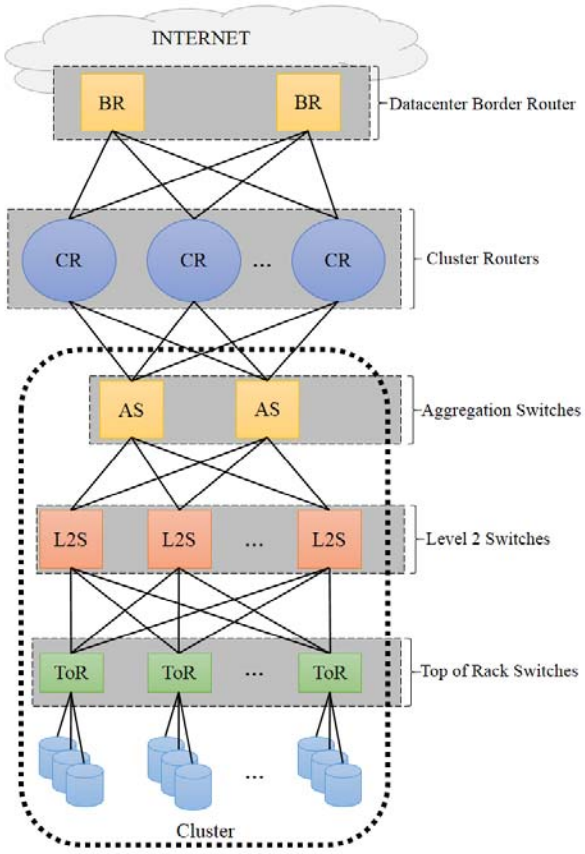

Fig. 5. Datacenter topology.

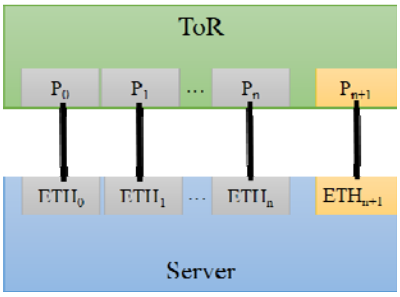

Fig. 6. Dedicated forensics port on a server.

Cascading the modification, the new datacenter topology will be like the one presented in Figure 7. We marked with a red line the alteration that must be made in the form of an extra Ethernet used port. This topology is just like a secondary backbone, which is used only by the forensic system and the investigators.

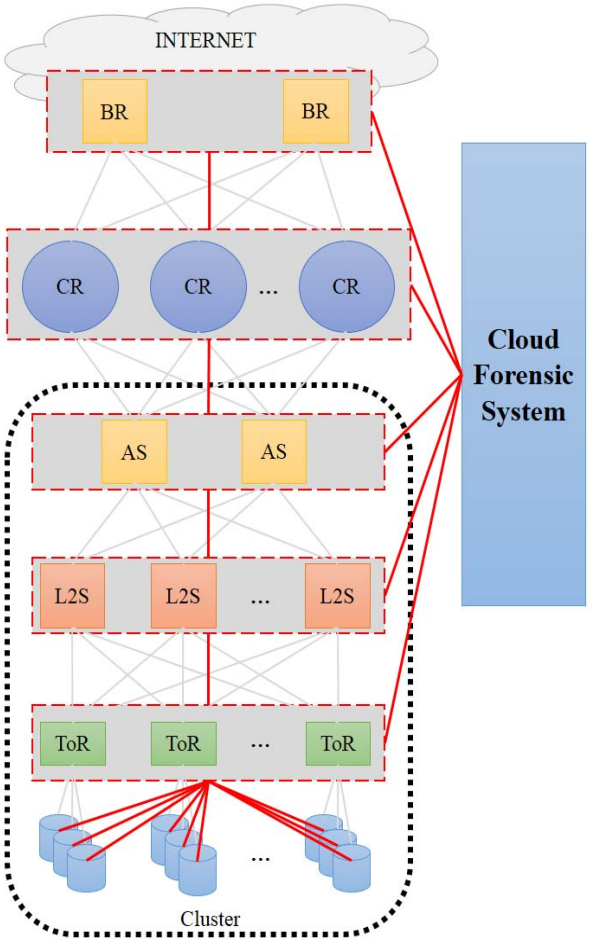

Fig. 7. Datacenter topology for Cloud forensics.

5. RESULTS

In this section we are going to present details regarding the results collected after the implementation of our Cloud Logging modules.

5.1 Testbed configuration

For testing, the modules have been implemented and split across multiple workstations, as can be seen in Figure 8. They are represented as a cluster of servers, each having the functionality presented in detail in the architecture section. As it can be seen, the entire modules found in the dotted perimeter, called ``Cloud Computing Forensic System'', can also be ran all on one workstation. Elements like network switches are not represented in order not to burden the graphic, but the IP address of the hosts are kept. In our configuration we have used three distinct workstations, each having the functionalities and network addresses presented in the figure.

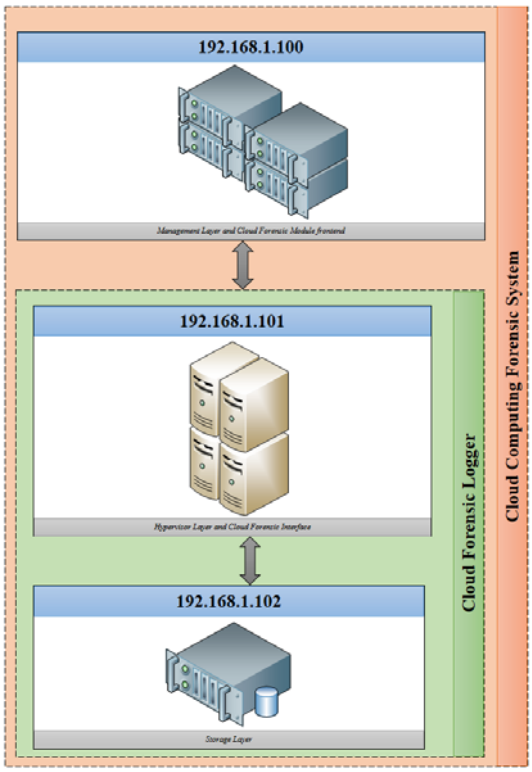

Fig. 8. Mapping modules to workstations.

The hardware platform used was composed from an AMD Phenom II X6, 6 cores, 8GB RAM, RAID0 configured hard-disks running KVM as hypervisor and QEMU as a hypervisor interface, an Intel DualCore, 4GB RAM as the storage layer and an AMD C-60 DualCore, 4GB RAM as the management layer. The network used is 10/100 MB.

5.2 Experimental results

The experiments were made using KVM as a hypervisor and QEMU and libvirt as drivers for the hypervisor. The tests that were made had the target set on the virtual machine used memory (RAM snapshot) and the virtual machine storage (DISK snapshot). The network communication is still work in progress and no actual results can be published.

The actual snapshots were taken using the following commands for RAM snapshot: *virsh snapshot-create-as VM_NAME SNAPSHOT_NAME –atomic* and for the DISK snapshot: *virsh snapshot-create-as VM_NAME SNAPSHOT_NAME --disk-only --atomic*

The process of recording the virtual machine activity was made over a period of several hours, at a time step of 10 minutes. The CPU load when conducting records using all the 6 cores was about 20%.

The results are interesting, if we take in consideration the technologies used internally by KVM. For example, RAM snapshots are made entirely from host machine RAM and do not contain necessarily consecutive RAM location. Nevertheless, in our experiments the RAM snapshots were the largest, reaching even gigabytes in size. On the other hand, the DISK snapshot is made efficiently by KVM each snapshot having a couple of tens or hundreds of megabytes in size.

Bellow you can see the actual tests that were made. We have split the tests in two distinct zones, one up to 100 MB and one after this barrier. Table 1 and Figure 9 presents the data collected from our modules and the time needed to process the entire data. The transfer time between the Cloud Forensic Interface module and the Storage module is not taken in consideration, as being a constant time, of about 82 seconds for a 800 MB file. Table 2 and Figure 10 presents the data collected from our modules and the time needed to process the entire data.

Table 1. Tests up to 100 MB in size.

Size (KB)	Time (ms)
4	296
454	517
1227	1136
5505	4929
10813	8000

Table 2. Tests over 100 MB in size.

Size (KB)	Time (ms)
108036	58982
740032	401156
4251346	2277855

Fig. 9. Tests up to 100 MB in size.

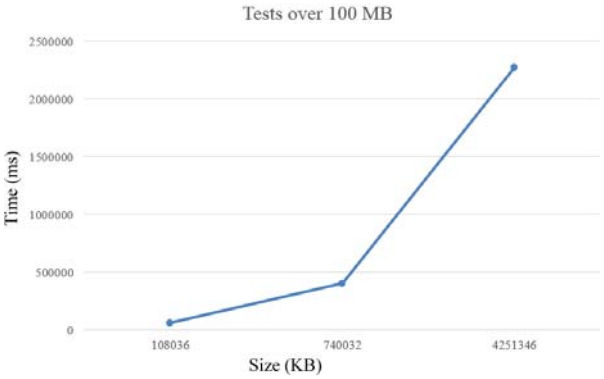

Fig. 10. Tests over 100 MB in size.

5. CONCLUSION

In this paper we presented a novel solution that provides to the digital forensic investigators a reliable and secure method in which they can monitor user activity over a Cloud infrastructure. Our approach takes the form of a complete framework on top of an existing Cloud infrastructure and we have described each of its layers and characteristics. Furthermore, the experimental results prove its efficiency and performance.

As we have seen, the field of cloud computing forensics and incident response is a new field for research that attracts more and more scientists. It poses a lot of challenges due to the distributed nature of the cloud but steps are starting to be made in this direction. In our paper we have presented a novel and new way in which user actions can be monitored and reproduced inside a cloud environment, even if it spreads over multiple datacenters.

Our work is focused on increasing reliability, safety, security and availability of Cloud Computing systems. The characteristics of such systems present problems when tackling with secure resource management due to its heterogeneity and geographical distribution. We presented the design of a hierarchical architectural model that allows investigators to seamlessly analyze workloads and virtual machines, while preserving scalability of large scale distributed systems.

As future work we intend to continue in this research direction in order to further optimize the snapshot and transfer algorithms, as well as the modules of the framework. Of course, further testing using more complex scenarios and a thin integration with other existing Cloud infrastructures would also help us to further improve our solution.

REFERENCES

A. Amarilli, D. Naccache, P. Rauzy and E. Simion, “Can a program reverse-engineer itself?”, in *Proceedings of the Thirteenth IMA International Conference on Cryptography and Coding*, 2011.

A. Atanasiu, R.F. Olimid and E. Simion, “On the Security of Black-Box Implementation of Visual Secret Sharing Schemes”, in *Journal of Mobile, Embedded and Distributed Systems*, 2012.

A. Pătraşcu and V. Patriciu, “Beyond Digital Forensics. A Cloud Computing Perspective Over Incident Response and Reporting”, in *IEEE 8th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, 2013

A. Pătraşcu, C. Leordeanu, C. Dobre and V. Cristea, “ReC2S: Reliable Cloud Computing System”, in *European Concurrent Engineering Conference*, Bucharest, 2012.

B. Grobauer and T. Schreck, “Towards incident handling in the cloud: challenges and approaches”, in *Proceedings of the 2010 ACM workshop on Cloud computing security workshop*, New York, 2010

G. Chen, “Suggestions to digital forensics in Cloud computing ERA”, in *Third IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC)*, 2012

G. Sibiya, H. Venter and T. Fogwill, “Digital forensic framework for a cloud environment”, *Proceedings of the 2012 Africa Conference*, 2012

http://cee.mitre.org/language/1.0-beta1/cls.html

http://uw714doc.sco.com/en/UDI_spec/m_mgmt.html

M. Al-Fares, A. Loukissas and A. Vahdat, “A Scalable, Commodity Data Center Network Architecture”, in *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, 2008

M. Simmons and H. Chi, “Designing and implementing cloud-based digital forensics”, in *Proceedings of the 2012 Information Security Curriculum Development Conference*, pages 69-74, 2012

NIST SP800-86 Notes, “Guide to Integrating Forensic Techniques into Incident Response”, http://cybersd.com/sec2/800-86Summary.pdf

R. Marty, “Cloud Application Logging for Forensics”, in *Proceedings of the 2011 ACM Symposium on Applied Computing*, 2011

S. Zawoad, A.K. Dutta and R. Hasan, “SecLaaS: Secure Logging-as-a-Service for Cloud Forensics”, in *8th ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, 2013

T. Takahashi, Y. Kadobayashi and H. Fujiwara, “Ontological Approach toward Cybersecurity in Cloud Computing”, 2010