Dizang : A solution for collecting forensic evidences of cloud malware injection attack

Hamilton J. S. Fonte II, Marcos A. Simplicio Jr.
Escola Politécnica, Universidade de São Paulo (USP) São Paulo, SP, Brasil
Email: hamiltonii@gmail.com, mjunior@larc.usp.br

***Abstract*—Cloud architectures are increasingly more common, as is the number of security problems involving this technology. Unfortunately, due to the volatile nature of resources in the cloud, the collection of evidences for forensic analysis has faced practical and legal challenges. With a technical focus, this work analyzes proposals aimed at meeting the challenges posed by the collection of evidences in the cloud, discusses their limitations and presents a solution to overcome them. The proposal specifically focuses on the reproducibility of the collection process, without violating jurisdictions or the privacy of those not involved in the investigation.**

I. Introduction

Visualization techniques, replication of services and resource sharing among multiple users (multitenancy) provide the computational clouds with high scalability [1]. Yet these mechanisms also lead to the volatility of the virtual resources that execute the applications. After all, when submitted to a high load, an application hosted in the cloud may create clones of the virtual machines (VMs) hosting it and balance the load between them, seeking to meet the demand without harming the quality of the service offered. After this peak, the cloned VMs are usually deactivated, their resources are released and the system returns to the previous capacity, preventing unnecessary costs.

Despite interesting from the efficiency and cost viewpoints, from the forensic point of view, the volatility of the cloud causes problems in case of attacks. For example, in case one of the virtual processing instances temporarily created undergoes threats that act directly on its memory, without leaving traces in disks (e.g. log files), the evidences of this event may be completely lost after they are deactivated and their resources are released. This difficulty is further aggravated by aspects such as multitenancy and multi-jurisdiction, typical of cloud solutions [2]. Particularly, the multitenancy aspect hinders identifying the hardware executing the applications of interest; since it is shared by a number of users, removing them for analysis could lead to a violation of privacy of the users not related to the investigation. Moreover, the distributed nature of the cloud may lead to allocating information relevant to the investigation in different countries, thus hindering obtaining this information, especially when there are no cooperation agreements among the entities involved [3]. Combined, these characteristics hinder collection of evidences with the necessary credibility so that they can be accepted in legal processes, which require respect to privacy, to the jurisdiction and to the chain of custody, as well as to the reproducibility of the collection process [4].

Although the literature presents solutions aimed at collecting information in the cloud for forensic analysis, most of them approach collection, transport and storage in an isolated manner. Works such as [5] and [6] deal with factors such as multitenancy and multi-jurisdiction, discussing forms of collecting and preserving evidence outside the cloud. Works such as [7] concentrate on forensic analysis to collect evidence from VMs while they are being executed, whereas works such as [8] deal with guaranteeing the chain of custody for transporting the evidence. However, none of the proposals identified (1) describe how data are collected and stored observing the chain of custody, and (2) allow the conditions for reproducing the evidence collection process even if a virtualized resource is removed.

This work aims to overcome these limitations, presenting a proposal focusing on: (1) the reproducibility of the collection process, (2) establishing a link between the evidence collected and its origin, (3) preserving the jurisdiction and the privacy of those not involved in the investigation. In sum, the solution described provides a way of correlating evidences and their virtual origin so as to preserve their credibility and thus contribute to their acceptability in a legal process. Lastly, Incident Management, which increasingly shares processes and tools with digital forensics, may benefit from a ready-to-use solution for collecting and preserving evidences at the *preparation for the incident* stage. It should also be able to preserve evidences for the *post-incident* stage, thus making it unnecessary to be concerned with their partial or total loss at the *detection and analysis stages*. For this, the system is supposed to be monitored and executed within a cloud resource univocally identifiable. The solution specifically targets code injection attacks [9], seeing that, when these are used against the cloud architecture, they leave no traces when virtual processing resources are deactivated and their memory is released [10], [9].

This paper is organized as follows. Section II briefly discusses cloud solutions and their characteristics. Section III analyzes the works related to the forensic memory area. Section IV details the solution proposed and assesses how it deals with the target challenges of this work. Section V presents the conclusions.

II. Adoption of Cloud Architecture and Containers

A computational cloud is an infrastructure model in which resources shared at a configurable number, accessible via net, are allocated and removed by a service provider with minimum management effort [11]. There are three main models for trading use in the cloud [11]: software as a service (SaaS), in which the software to be used by the client is provided; platform as a service (PaaS), in which an environment is provided for clients to develop, test and execute their software; and the most pertinent type for this work, infrastructure as a service (IaaS), in which basic computational resources are provided, such as processing and memory, generally in a virtualized way.

The virtualization of resources in the cloud, despite traditionally conducted by means of VMs, has also increasingly been peformed in the form of containers. According to a study conducted in 2016 with 235 companies having software development as their end activity or as a support to the end activity [12], 76% of the respondents use containers to improve the efficiency of the development process and in their architecture of cloud micro-services. Differently from VMs, which involve creating a virtual hardware and also a SO on top of the native system, the virtualization with containers is conducted at the level of the native SO, which has a simpler implementation, thus eliminating layers between the application executed and the physical hardware. A technology widely used for this purpose is the Linux Containers (LXC), which takes advantage of functionalities such as cgroups and namespacing of the Linux kernel to help managing and isolating virtual resources.

III. Related works

There are a number of aspects related to forensic analysis in the cloud, going from information collection to guaranteeing the evidence chain of custody. For a better structured discussion of the works available in the literature, summarized in Table I, they are presented as follows, based on the different aspects they approach.

Table I
Comparison of solutions for collecting memory information from cloud machines for forensic analysis

	Is the collection continous?	Can it reproduce the process without the VM?	Does it guarantee the chain of custody?	Does it preserve jurisdiction and privacy?
Dizang (this proposal)	✓	✓	✓	✓
[7]	✘	✘	✘	✓
[13]	✘	✘	✘	✓
[5]	✘	✘	✘	✓
[14]	✘	✘	✘	✓
[6]	✘	✘	✓	✓
[8]	✓	✘	✓	✓
[15]	✘	✘	✘	✓
[16]	✘	✘	✘	✓
[17]	✓	✘	✘	✓
[18]	✓	✘	✓	✓

A. Accessing and collecting information about memory in the cloud

Different works on forensic analysis in the cloud concentrate on collecting data "after the fact", i.e., after the intrusion has been detected [6], [13], [5], [7], [8]. The collection processes described in those works can be started manually or automatically, by integrating them with an intrusion detection mechanism. In the specific case of volatile memory, that form of collection cannot describe what the memory was like before the intrusion, since the process is only activated after the attack has been detected. This limitation can be harmful to the investigation, given that some analyses depend exactly on their ability to compare two moments of the memory [9]. Among the works studied, the only proposal found that considers this need is [17], which proposes that the data are stored in the very equipment under analysis. Unfortunately, however, the application of this approach to the cloud scenario is not very viable, since it can lead to losing important information in case the VM or the container comes to be deactivated and has its resources released.

There are also works aimed at collecting information during the system execution, in which data are constantly collected with no distinction between what happened before or after the event of interest. This is the case of works such as [13], [5], [8], which adopt the strategy of isolating and halting the VM for then conducting the collection process. Albeit interesting, the approaches described in these works may lead to a massive volume of data collected, besides not treating the scenario from which evidence has to be collected when the virtual resources containing the information are released.

B. Ability to reproduce the process and to obtain the same results

If, during a forensic analysis, different analysts obtain distinct results when executing the same collection procedure, the

evidence generated has no credibility and may come not to be accepted in a legal process. For this reason, the reproducibility of the collection process is an important part in evidence generation for forensic analysis. None of the proposals currently found in the literature allows this reproducibility in cloud scenarios in which VMs or containers are deactivated and their physical resources are released; they all depend on the existence of the virtual resource for repeating the collection process.

C. Not violating the privacy or jurisdiction of the parts not involved in the investigation

In a public cloud environment, removing the hardware for later analysis may lead to violating the privacy of users, once this scenario multitenancy makes the same physical machine store information about different clients, some of which may not be involved in the investigation in course. Different works in the literature adequately deal with this issue using two major strategies: the first, adopted by [6], [7], [13], [5], consists in collecting data pertinent to the investigation and in storing them outside the cloud.; the second, employed in [8], which consists of a specific case of [7], depends on the cooperation of the cloud service provider for obtaining the information necessary to the investigation. Yet depending on the cloud service provider is not a recommended strategy because (1) the volume of users' data may force the providers to limit the size of the logs stored, and (2) in case unavailability caused by an attack occurs, the goal of the provider will be that of restoring the service, rather than preserving evidences [19].

IV. Solution proposed: Dizang

This proposal aims mainly at collecting memory from virtual computational resources so as to succeed in: (1) identifying the evidence source, even if the virtual resource no longer exists; (2) describing the system before and after the incident; (3) transporting and storing the memory collected so as to guarantee its integrity and confidentiality; and (4) not violating the jurisdiction and privacy of other users that might have resources allocated in the same physical server. The solution here presented, called Dizang, is described in detail as follows.

A. Description

In computational systems executed on a physical infrastructure (i.e., non-virtualized), a direct association can be made between any resource, such as memory information, disk image or packets travelling in the network, and their corresponding origin. In systems built in a virtual infrastructure, especially when it is self-scalable, the computational resources are highly volatile and can thus be removed at any time. To succeed in correlating a piece of evidence to its volatile origin, this implementation uses Linux containers (LXC) to persist in the source-evidence relationship. Although a container is a type of software, and is thus also volatile, each image compiled and its execution in the form of a container are normally linked to a hash that univocally identifies this relationship. The container also allows more precisely identifying the source of a piece of evidence, once it is possible to divide the parts of a system into containers; for example, one container for the engine of dynamic web pages (e.g. Apache), another with business logics (e.g. Golang) and a third one for a databank (e.g. Cassandra).

Memory copy is not an atomic activity, since it is executed jointly with other processes. Hence, in case one of these processes is a malicious code erasing traces of its existence from the container memory, possibly important information may eventually be lost. Aiming to make the memory copy process more atomic, Dizang temporarily interrupts the execution of the container, makes a copy of its memory and then resumes its execution. This technique, similar to that adopted by [20] for VMs, produces a photograph of the container volatile memory; this allows its analysis at a repose state, that is, without the need of having the container at work. When conducting the data collection at adequate time intervals, it is possible to build the history of the memory state during its execution in the container.

Most of the forensic techniques currently most widely employed are directed to obtaining information in its totality, be it via bit by bit copy, be it by obtaining the physical hardware [21] [22]. Even though these techniques may sound interesting at first, many times they are eventually responsible for a problem: the growing volume of information investigators have to analyze [23]. To mitigate this difficulty, Dizang adopts two strategies. The first is defining a volume of data that may be considered enough for conducting an investigation; the second is defining a maximum age for the evidence while the system works under normal conditions, i.e., when it is not under attack. To detect and to analyze intrusions in the processes memory, it is necessary to have a copy of the memory before and after the intrusion [9]. The solution proposed thus implements a window of memory photographs covering a pre-defined time interval, as illustrated in Fig. 1. In normal operation conditions, the pieces of evidence are collected with a certain periodicity and collections reaching a given age are discarded. In contrast, after an attack event is detected, (e.g. by an intrusion detection system), Dizang stops discarding the oldest collections from the monitoring log, enabling to know the system before and after the attack and hence assess its evolution.

To persist in the evidence-origin relationship and to guarantee its integrity, this proposal calculates the pair hash [evidence, container image identifier] and stores the triple [hash, container image identifier, evidence]. To prevent possible problems with the storage of these data in countries with different jurisdictions from those in which they must be applied in the investigation in question, the evidence collected is stored in a physical place outside the cloud, after being transported by a safe channel (e.g. via Transport Layer Security – TLS [24]).

B. Implementation

The methods proposed were implemented in a test platform aiming to assess the Dizang efficacy in collecting the memory information from the containers in a reproducible

Figure 1. Sliding window of evidence collection

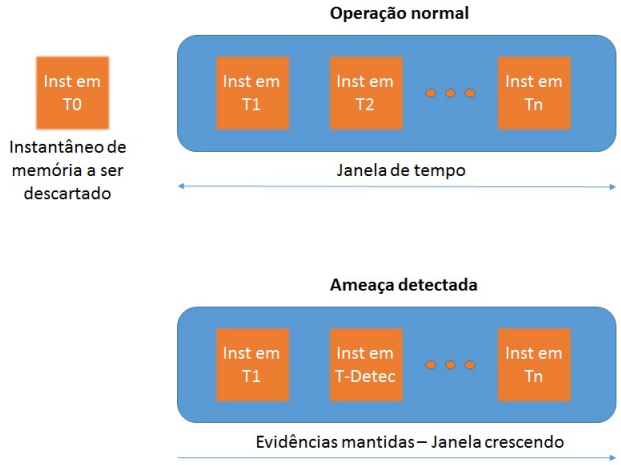
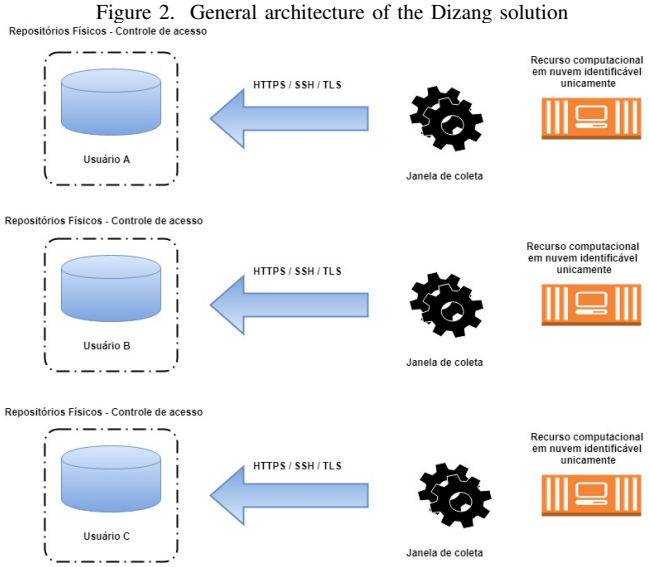


Figure 2. General architecture of the Dizang solution



way, without violating jurisdictions or users' privacy. The solution, illustrated in Fig. 2, consisted in creating a t2.micro instance in the Ohio zone of the AWS with 3.3 Mhz, 1Gb RAM and 64-bit operational system. At this AWS instance, Docker Engine 1.10 and API Docker 1.21 were installed, from which 3 containers were created, executing nginx 1.0 at different ports. Using a Java application that discovers the process identifier associated to each container, it was possible to copy the content of the *non-uniform memory allocation descriptor* (**/proc/pid/numa_maps**), containing the allocation of the memory pages, the nodes associated to these pages and what was allocated in their respective access policies [25]. The copy and recording of the file is such that, every minute, the application (1) pauses the container in question, (2) copies the **numa_maps** directory, (3) concatenates the data obtained with the container image identification hash, (4) calculates the set hash, and (5) saves the result in a file whose name is the container image identifier and whose extension is **.mem**. The safe transport of the evidence to a physical storage outside the AWS was implemented using a t2.micro instance in the AWS Ohio zone whereby an OpenVPN server had been installed. The EC2 instance containing the evidence was configured to accept connections solely from machines in the VPN. A physical machine outside the AWS used the OpenVPN client to establish a VPN connection with the instance containing the evidence and transported it to the disk of the physical machine. After concluding the transport process, the physical machine verifies whether there are .mem files in disk older than a certain "t" time interval, and discards them.

C. Experimental results

To assess the Dizang efficacy in collecting evidence, some experiments were performed using the environment implemented (described in Section IV-B). Firstly, the system was configured for memory collections at 1-minute intervals, to save them in a disk in a physical machine outside the cloud and to erase the samples collected over 5 minutes before. The system was then executed for 30 minutes, a time along which the following were collected as metrics: (1) the use of space in disks used by the memory photographs saved, (2) the time necessary for pausing the container for copying them, and (3) the time taken for transporting the evidence to the physical machine outside the cloud. At each collection, command `du -sh *.mem` do *Unix* was executed in the physical storage disk outside the cloud to return the list of the files in which the memory photographs were stored and the disk space they occupied.

The disk space occupied by the memory photographs captured during the experiment is shown in Fig. 3. The graph shows that the increase in use of the disk space is linear and the growth ceases when the time limit configured for the window is reached, once the collections with a lifespan larger than that limit are erased from the disk. The solution thus keeps the disk space occupied by the samples collected under control. Concurrently, memory photographs saved by the solution after the containers and the AWS instance are removed and kept in the disk of the physical machine outside the cloud; they may be associated to their origin, as expected for a forensic analysis. This ability is kept after a threat has been detected since, in this case, older collections stop being erased so it is possible to describe the state of the system before and after the incident [9], allowing, for example, code injection attacks to the memory to be analyzed..

A potential limitation of the solution proposed is that the pause of a container to collect data may, in principle, cause performance losses of the application being executed. To assess this impact, the times for copying the container memory were measured along the experiment. The results are shown in the graph of Fig. 4. Note that, after the application is initialized, the time for making a copy is very small, varying between 20 and 40 milliseconds. Especially for containers executing the engine of dynamic web pages, as is the case of the experiment in question, this latency can be not very perceptible by end users.

Another concern is the time for transporting the evidence to the storage outside the cloud. The evidence transport to the storage outside the cloud can take longer than generation interval, leading to evidence loss during transport. To assess this impact, the evidence transport times for storage outside the cloud were measured along the experiment, as shown by the results in the graph of Fig. 5. The transport time is verified to stabilize after the window size is reached. The time for transporting evidence is approximately 30 seconds on average. The topology is a factor that may contribute both positively and negatively to the transport time. In this experiment, the evidence generator is in North America, whereas the physical machine where the evidence was transported to is in South America.

D. Limitations

Since the solution described focuses on collecting memory information from the user space, it cannot access the kernel space. In principle, therefore, Dizang does not provide support to malware investigation techniques based on information from the kernel space, such as comparing the information from the Process Environment Block (PEB), which are in the user space, with information from the Virtual Address Descriptor, which lies in the kernel space. Analyses of threats that directly manipulate the objects of the kernel (D.K.O.M – Direct Kernel Object Manipulation) do not benefit from the solution proposed, either.

V. Final Considerations

Digital threats acting directly on the system memory do not usually leave traces in disk after their corresponding resources are removed, hindering later forensic analyses. This problem is especially notable in cloud computing systems, in which the allocation and removal of virtualized resources (e.g. VMs and containers) are frequent. This characteristic, together with aspects such as multitenancy and multi-jurisdiction of computational clouds, hinders evidence collection for investigating incidents.

In this scenario, the proposal presented aims to relate the memory photograph to its origin, using the calculated hash of the container image as a stored evidence identifier. To prevent an excessive use of disk space, the volume of data stored uses a storage window, which allows describing the memory before and after an attack (e.g. memory injection). Combined with a tool for identifying threats, these Dizang characteristics make it a robust solution to provide evidence and thus make forensic analyses in the cloud viable.

As future work, we intend to analyze the containers memory of the same system using the collections made by Dizang . With these results, the intention is to implement a more flexible tool, which dispenses with accessing the machine full memory, a common requirement of the current malware analysis tools (e.g., FROST [5] e Volatility Framework [26]).

References

[1] A. M. Morsy, J. Grundy, and I. Muller, "An Analysis of the Cloud Computing Security Problem," in *APSEC Cloud Workshop*. Sydney, Australia: Cornell University, 2010. [Online]. Available: https://arxiv.org/abs/1609.01107

[2] P. Gilbert and S. Sujeet, *Advances in Digital Forensics IV*, 1st ed. Orlando: Springer-US, 2008, vol. 1.

[3] J. Dykstra and A. Sherman, "Acquiring forensic evidence from infrastructure-as-a-service cloud computing: Exploring and evaluating tools, trust, and techniques," *Digital Investigation*, vol. 9, pp. S90–S98, 2012, (Proc. of the 12th Annual DFRWS Conference). [Online]. Available: dx.doi.org/10.1016/j.diin.2012.05.001

[4] S. Rahman and M. N. A. Khan, "Review of live forensic analysis techniques," *International Journal of Hybrid Information Technology*, vol. 8, no. 2, pp. 379–388, 2015. [Online]. Available: www.sersc.org/journals/IJHIT/

[5] J. Dykstra and A. T. Sherman, "Design and implementation of FROST: Digital forensic tools for the OpenStack cloud computing platform," *Digital Investigation*, vol. 10, pp. S87–S95, 2013, (Proc. of 13th Annual DFRWS Conference). [Online]. Available: dx.doi.org/10.1016/j.diin.2013.06.010

[6] Z. Reichert, K. Richards, and K. Yoshigoe, "Automated forensic data acquisition in the cloud," *IEEE Int. Conf. on Mobile Ad Hoc and Sensor Systems*, pp. 725–730, 2015.

[7] S. George, H. Venter, and F. Thomas, "Digital Forensic Framework for a Cloud Environment," in *IST Africa*. Tanzania: IIMC, 2012, pp. 1–8.

[8] T. Sang, "A log-based approach to make digital forensics easier on cloud computing," *Intelligent System Design and Engineering Applications (ISDEA)*, pp. 91–94, 2013.

[9] A. Case, M. Ligh, L. Jamie, and A. Walters, *The Art of Memory Forensics: Detecting malware and threats in Windows, Linux and Mac memory*. Hoboken, NJ: Wiley, 2014.

[10] S. Vömel and J. Stüttgen, "An evaluation platform for forensic memory acquisition software," *Digit. Investig.*, vol. 10, pp. S30–S40, 2013, elsevier Science Publishers. [Online]. Available: http://dx.doi.org/10.1016/j.diin.2013.06.004

[11] P. Mell and T. Grance, "The NIST definition of cloud computing," NIST SP 800-145, p. 7, 2011. [Online]. Available: csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[12] DevOps and ClusterHQ, "Container market adoption survey 2016," https://clusterhq.com/assets/pdfs/state-of-container-usage-june-2016.pdf, 2016.

[13] R. Poisel, E. Malzer, and S. Tjoa, "Evidence and cloud computing: The virtual machine introspection approach," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 4, no. 1, pp. 135–152, 2013. [Online]. Available: citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.469.937

[14] D. Barbara, "Desafios da perícia forense em um ambiente de computação nas nuvens," Univ. do Planalto Catarinense, Tech. Rep., 2014, revista.uniplac.net/ojs/index.php/tc_si/article/view/1911.

[15] B. Dolan-Gavitt, T. Leek, M. Zhivich, J. Giffin, and W. Lee, "Virtuoso: Narrowing the semantic gap in virtual machine introspection," in *IEEE Symposium on Security and Privacy*. Plymouth, UK: IEEE, May 2011, pp. 297–312.

[16] A. Aljaedi, D. Lindskog, P. Zavarsky, R. Ruhl, and F. Almari, "Comparative analysis of volatile memory forensics: Live response vs. memory imaging," in *IEEE 3rd Int. Conf. on Privacy, Security, Risk and Trust*, 2011, pp. 1253–1258.

[17] F. Dezfouli, A. Dehghantanha, R. Mahmoud, N. Sani, and S. Shamsuddin, "Volatile memory acquisition using backup for forensic investigation," in *Int. Conf. on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec)*. Plymouth, UK: IEEE, June 2012, pp. 186–189.

[18] R. B. van Baar, H. M. A. van Beek, and E. J. van Eijk, "Digital Forensics as a Service: A game changer," *Digital Investigation*, vol. 11, pp. S54–S62, 2014. [Online]. Available: dx.doi.org/10.1016/j.diin.2014.03.007

[19] S. Alqahtany, N. Clarke, S. Furnell, and C. Reich, "Cloud forensics: A review of challenges, solutions and open problems," in *Int. Conference on Cloud Computing (ICCC)*. Plymouth, UK: IEEE, April 2015, pp. 1–9.

[20] M. Rafique and M. N. A. Khan, "Exploring Static and Live Digital Forensics: Methods, Practices and Tools," *IJSER*, vol. 4, no. 10, pp. 1048–1056, 2013. [Online]. Available: www.ijser.org/researchpaper{\%

}5CExploring-Static-and-Live-Digital-Forensic-Methods-Practices-and-Tools.pdf

[21] S. Simou, C. Kalloniatis, E. Kavakli, and S. Gritzalis, "Cloud forensics: Identifying the major issues and challenges," in *Advanced Information Systems Engineering (CAiSE 2014)*, vol. 8484. Cham, CH: Springer International Publishing Switzerland 2014, 2014, pp. 271–284.

[22] D. Bem, F. Feld, E. Huebner, and O. Bem, "Computer forensics - past , present and future," *Journal of Information Science and Technology*, vol. 5, no. 3, pp. 43–59, 2008.

[23] D. Quick and K. K. R. Choo, "Impacts of increasing volume of digital forensic data: A survey and future research challenges," *Digital Investigation*, vol. 11, no. 4, pp. 273–294, 2014. [Online]. Available: dx.doi.org/10.1016/j.diin.2014.09.002

[24] R. E. Dierks T, "The Transport Layer Security (TLS) Protocol," IETF: https://tools.ietf.org/html/rfc5246, Fremont, CA, 2008.

[25] Unix Man Pages, "Numa Maps - Non Uniform Memory Architecture," man7.org/linux/man-pages/man7/numa.7.html, acessado em: 24-06-2017.

[26] Volatility Foundation, "Volatility Framework," www.volatilityfoundation.org/, 2014, acessado em: 24-06-2017.

Figure 3. Evolution of disk space use under Dizang .

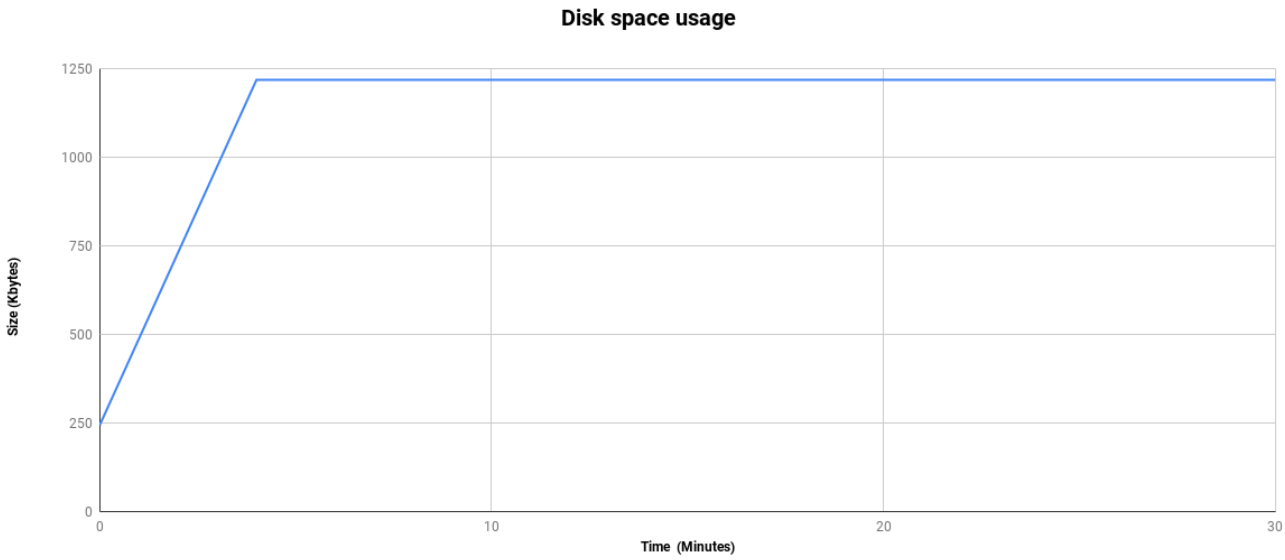


Figure 4. Time for copying the container

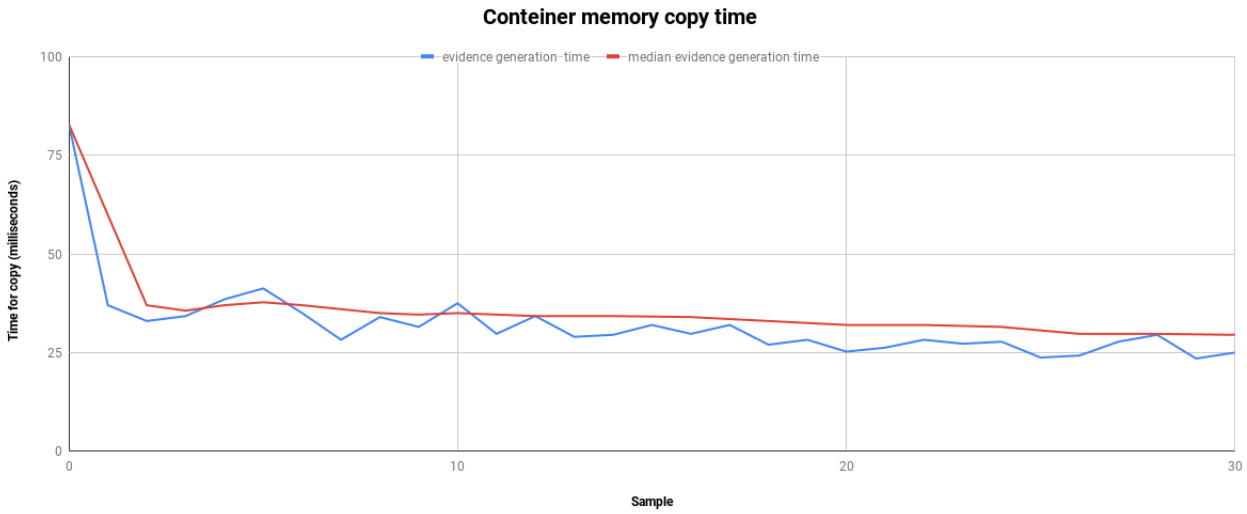


Figure 5. Evidence transport time