DCFF: a container forensics framework based on Docker

Du Jiang[1, a], Wu Sheng[1,b]

1Chongqing University of Posts and Telecommunications, 2 Chongwen road, Nan'an District, Chongqing, China

[a]clouddu@163.com, [b]dylanwu123@foxmail.com

Keywords: container virtualization, cloud computing, cloud forensics, container forensics

Abstract. As a lightweight and flexible virtualization technology, container virtualization has been adopted by more and more platform as a service (PaaS) system. With the popularity of container virtualization and PaaS, cloud forensics need to find a way of extracting integrate and reliable data from containers. In this paper, we propose a container forensics framework called DCFF which is designed to acquire data simultaneously from containers running on different hosts and transform forensics data into compatible format centrally.

1 Introduction

In recent years, with the development and maturation of container virtualization technology(CVT) [1], more and more cloud service providers, such as Google, Microsoft, Amazon, VMware and IBM, have supported this more flexible and efficient virtualization technology. And many of the latest PaaS system, such as Flynn, Deis, Tsuru, Dawn, Octohost and so on, have built their infrastructure base on Docker [2]. It's easy to tell that CVT is playing a very important role in the field of PaaS.

With the popularity of CVT and PaaS, cloud forensics science needs an answer of how to extract integrate and reliable data from containers. Now, container forensics is mainly facing the following problems.

1) No theoretical framework for container forensics. The widespread application of virtualiztion and cloud computing tech, has coursed the highly attention of experts and professionals in digital forensics field. And now we have many theory and method of virtualzation forensics, but all of them is built on hypervisor vitualzation tech (HVT), instead of CVT.

2) Live forensics. Because CVT is mostly used for cloud service, just like HVT, we cannot just shutdown the container and take their data. Container forensics must be capable of acquiring data from a running container. And the target container should not be able to notice the operation of digital forensics.

3) Container forensics for multiple hosts. As a lightweight virtualization technology, CVT is widely used to construct PaaS platform. But unlike IaaS, PaaS can run on multiple independent hosts, even if hosts from different service provider. And that requires container forensics be able to not just acquiring data from containers in one host, but also managing the forensics process on multiple hosts.

4) Forensics data compatibility. Like other virtualization technologies, CVT saves container data in image of proprietary format. But an exclusive image file is not useful to the follow-up of data analysis. So container forensics not only means extracting data from the container, but also transforming data into a compatible format.

For solving these problems, we propose DCFF which is able to acquire data simultaneously from containers running on different hosts and transform forensics data into compatible format centrally.

2 Analysis of related work

The development of virtualization technology has brought new challenges to the traditional digital forensics field. The subject of digital forensics changes from physical machine to VM(virtual machine), from hardware storage device to VM file. Now, with the virtualization and cloud

computing tech working more closely [3], forensics targeting virtualized environment has became a crucial part of cloud forensics [4].

Dr. Zhou Gang of the Huazhong University of Science and Technology presented a cloud forensics method base on scene migration [5]. The method takes VM instances as the forensics subject. When there is a forensics demand, VM instances will be migrated completely to local server. Then, it use traditional forensics tools to handle VM instance locally. The problem with this method is that the migration process could last a very long time, and the VM instance in migratory status may not work properly. In addition, this method requires a very powerful local server or server cluster when dealing with multiple forensics assignments.

XIE Ya-long, DING Li-ping et al proposed a cloud forensics framework under the IaaS mode (ICFF) [6]. By installing an EC (evidence crawler) in VM instance and a real time process monitor in hypervisor, ICFF can acquire VM's data at the first moment when any irregular process or log entry occurs. However, subject VM in this framework is modified in advance which is not acceptable to it's user, and when the EC is compromised ICFF can get nothing but limited process information from the VM.

Patrick Tobin and Tahar Kechadi et al presented a VM forensics method [7] that can collect data of remote VM in local machine. The core idea is to build an duplicated VM instance in the local. So every data that should be extract from the remote VM can also be acquired locally. In order to achieve that, they need kernel hook in remote VM and VM introspection tools in remote virtualization environment to grab real time data of VM instance. With the data saving every thing happens in remote VM, they can replay the exact change in source VM in chronological order by using injected code. What they have presented is a novel idea, but again, modification of VM is not acceptable to it's user. And a regular security update may be able to fail the code injection.

3 System design

3.1 Docker introduction

Docker is a senior container engine developed by dotCloud. It has three core concepts: container, layered image, Docker Hub. Container is isolated, virtualized environment built from layered image. And the layered image could be shared all over the world through Docker Hub.

For Docker, a container is a dynamic image, and an image is a static container. When we use a image to create a container, Docker will create a r-w(read-write) image layer for the container. Any changes in the container will only affect the r-w image layer. So that, when we want to acquire the container's files, all we need to do is making a copy for the r-w layer. The relationship between container and layered image is shown in Fig. 1. Currently, Docker uses AUFS(Advanced Union File System) [8] in Ubuntu OS to construct the layered image system, while it's thin provision snapshot strategy of device mapper [9] in non-Ubuntu OS.

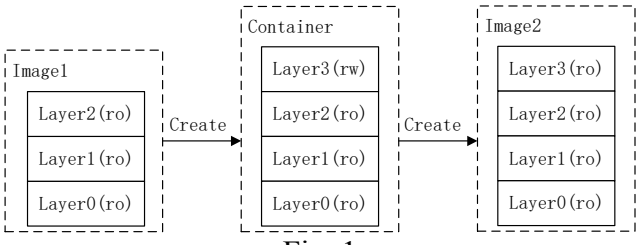

Fig. 1

Layered image is a solution for data sharing between different images, while volume is a means for different containers. A volume can be mounted by more than one container, it's can be a volume container, a host directory, or even a host file. Volume is completely independent from container that no image created from container would include any data in volume. Still, volume is a very important bridge, even a data vault for container.

3.2 Framework

This paper presents DCFF that can acquire data from a running container and convert the data to

compatible image file. As shown in Fig. 2, DCFF mainly consists of 7 modules in three parts, including: forensics controller, image converter and forensics image pool in forensics control center; agent manager, container information collector, container relationship analyzer and container camera in forensics agent; local Docker registry.

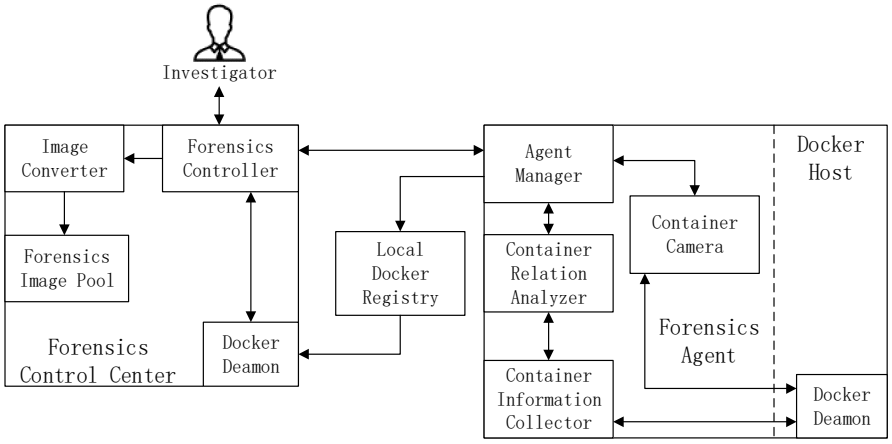

Fig. 2

Forensics control center(FCC). FCC controls forensics task on all the Docker host. Data from different host will converge on FCC and be transformed into compatible forensics image file. The investigator can order forensics assignment of every container runs on every host through FCC. Each order can include more than one container forensics command. And if the order involved with multiple containers on same host, that means data acquired from these containers is what they are like at the same time point.

Forensics agent(FA). Forensics agent is forensics agency unit that working on the Docker host. It can receive and respond to commands from FCC. The major functions of FA is saving containers into container images, getting the runtime state information of containers and acquiring volume used by containers. FA will push container images to LDR, while sending container's runtime state and volume to FCC.

Local docker registry(LDR). LDR [10] is a private Docker image platform similar to Docker hub, which can be used for image forwarding. What actually forwarded through LDR is image layer. So if we have two images sharing same layers that need to be transmitted, we don't have to repeat the transmission for the same layers. Using LDR can greatly improve the image transmission efficiency between local machines.

Docker deamon. Docker daemon is a Docker architecture that runs in the background, through which we can use the services provided by Docker.

3.3 Forensics control center

FCC runs on Ubuntu OS, using layered image conversion method base on AUFS to implement container image transformation. Forensics data from all the host is encrypted and stored in here. FCC contains three modules respectively. They are forensics controller, image converter, forensic image pool.

1) *Forensics controller*(FC). As the investigator interface and container forensics control unit, FC can do Docker host keeping alive, information query, forensics command distribution, forensics data process, forensics image export and deletion.

Docker host keep alive: FC maintains an online host list inside, when it is running. The online host list consists of IP address, domain name, OS information and Docker version of every working host. When FC starts, it broadcasts a request for keep alive package including host's information. Then the same package is transferred between FC and Docker host periodically. And if FC doesn't receive any message from a Docker host in a specific cycle, it would delete the host from online host list, and it wouldn't contact the host until the host start sending keep alive package again.

Information query: As long as the investigator needs, FC could send info query to every host in online host list and display all the data returned. That means except information in online host list, FC is able to reach the detail about all the containers running on online hosts. Besides that, FC can

also search and display container forensics record and local forensics image information.

Forensics command distribution: Forensics command from investigator may involve containers from different hosts. Before sending command to any host, FC need to verify if these containers or hosts are existing, than it can divide the command by host unit and send the divisions to different hosts.

Forensics data process: Forensics data from all hosts are brought together in FCC. And it is FC's job to turn those data into compatible forensics image. Now, we have image converter to do the transformation for single container, but still, data from different containers need to be distinguished decently. So what FC does here is to separate the data of each container from others', invoke image converter to complete the format conversion, and update the forensics and image records.

Forensics image export and deletion: FCC keep forensics images in a place where no one other than FC and IC can reach directly. So it's very important to create an access for investigator. Through FC, investigator can delete or export existing forensics images in FCC.

2) *Image converter*(IC). IC is responsible for transformation from forensics data to forensics image. And there are 3 parts in the convert process, which are reconstruction of container's root file system, putting container's volumes, runtime state info and root file system in a directory as indicated in Fig. 3, turn the directory into a ISO image file with the same name.

As we know, Docker uses two strategy to build layered image system for different type of OS. But luckily, with the participation of LDR, we don't have to worry about which container comes from which kind of OS. We just need to focus on one strategy, and recover the root file system on our own.

AUFS is a much easier way to build a layer image. And that's why we choose the Ubuntu OS to run FCC. Docker marks the layer relationship of images in it's root directory, and that's where we can find the answer for how to construct container's file system.

Volumes come from Docker hosts is concealed in image files. We need to mount they straightly in the directory which shown in Fig. 3.

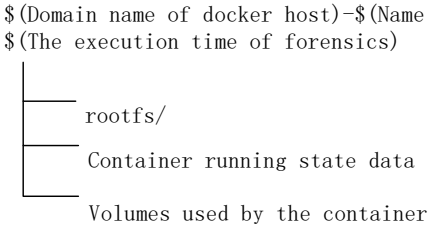

Fig. 3

3) *Forensics image pool*(FIP). FIP is an encrypted data base storing forensics image of containers,. It accepts access from FC and IC only.

3.4 Forensic Agent

Forensic agent(FA) is installed on Docker host, mainly responsible for the data acquisition of local containers. There are three kinds of data that FA would gather for FCC. The first is the container image created from the root file system of the target container. The second is the volume which is being used by the target container. And the third is the runtime state info of the target container.

However, there are two routes for the travel of containers' data. The volume and the running state info will be digested, encrypted and sent to FCC directly, while container images is sending to LDR. And since the LDR wouldn't repeatedly transport the same image layer, forwarding images though LDR is a more efficient way for the framework.

There are 4 modules works on FA. They are agent manager, container information collector, container relation analyzer and container camera.

1) *Agent manager*(AM). AM is the control unit of the FA, it sends the keep alive package to the FCC periodically, and responds to the forensics command form FCC.

Docker host keep alive: Once receiving the keep alive request from FCC, AM gains host info from container information collector, and creates a keep alive package for delivering it to FCC. The package includes IP address, domain name, OS version, Docker version. The keep-alive mechanism

works mutually. When sending the keep alive package repeatedly, AM is expecting the same package could be returned from FCC. If AM didn't receive the keep alive package from FCC in a specific time window, it would presume that the AM is down and suspend FA until receiving FCC's keep alive request again.

Information query response: AM has the obligation to update local container information to FCC after receiving a information query. It could gets ID, name, start command, created time, status, boot image and port mapping of local containers from container information collector and send them to FCC.

Forensics command response. After receiving forensics command from FCC, AM will start the digital forensics of target containers. The work flow of container forensics is shown in Fig. 4. It includes 6 steps: ① Sending target list(list involving target containers) to container relation analyzer; ② Getting pause list(list involving container that need to be paused during forensics), V/C table(table involving relationship of container and volume) and volume directory from container relation analyzer; ③ Sending target list, pause list and volume directory to container camera; ④ Receiving forensics execution time, container images, volume images and target container's running state from container camera; ⑤ Pushing the container image to LDR; ⑥ Digesting and sending forensics execution time, volume images, V/C tables, target container's runtime state to FCC.

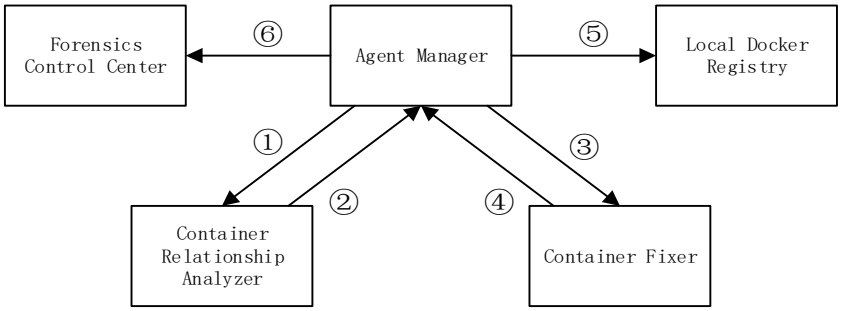

Fig. 4

2) *Container information collector*(CIC). CIC provides info query service for other module on FA. It maintains Docker and container configuration tables internally. By using file monitor strategy, These tables can be updated immediately when any changes happen. The work procedure of CIC is shown in Fig. 5.

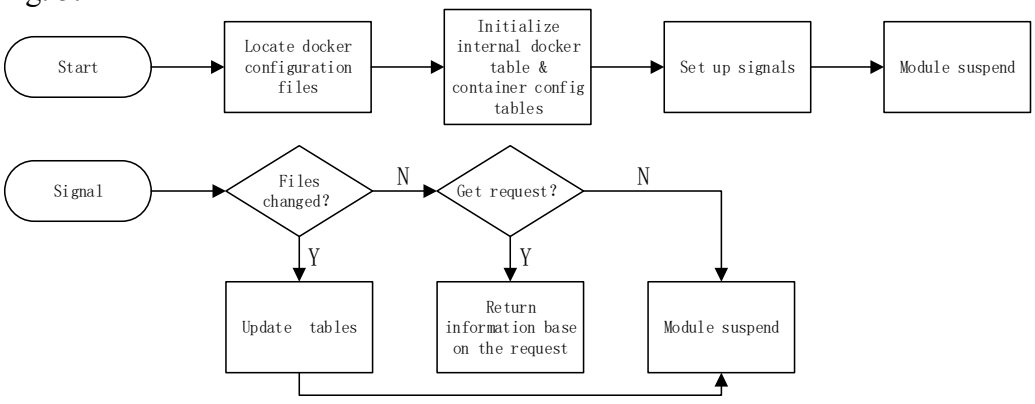

Fig. 5

3) *Container relation analyzer*(CRA). Since one volume can be mounted in different containers. CRA needs to find out what other containers are using the same volume or where is the volume's location. The work procedure of CRA is shown in Fig. 6.

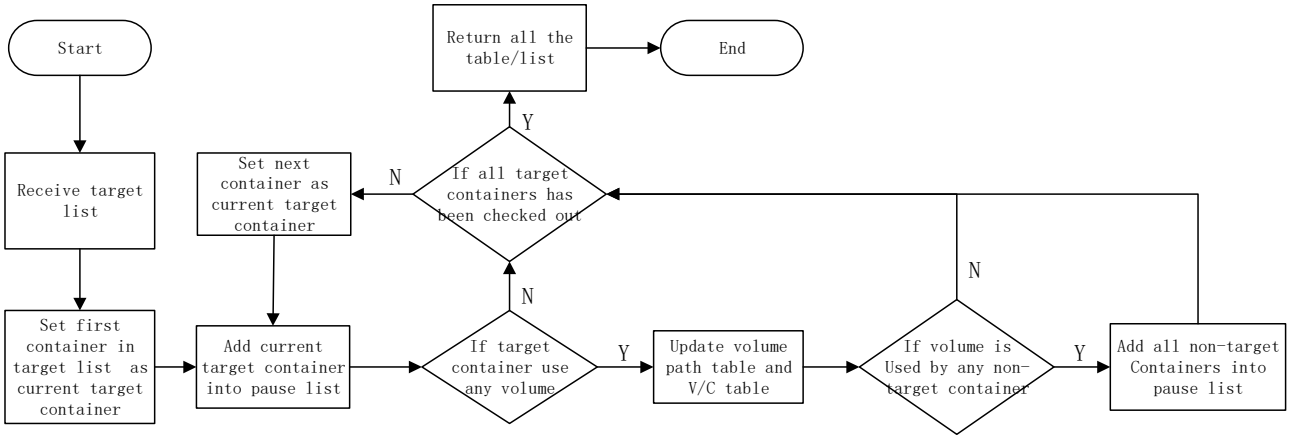

Fig. 6

4) *Container camera*(CC). CC takes charge of container data immobilization. It preserves container data of a certain time by memorizing containers' runtime state and creating images from containers and volumes.

There are 4 steps in CC's procedure, which is container pausing, data storing, container unpausing and data submission. Firstly, CC pauses all the container in pause list which is sent from AM. Then, it invokes Docker API to get target containers' runtime state and build layered images for target containers. During this time, CC also creates ISO images from volume directory and memorize forensics execution time. After that, CC will unpause all containers that is paused previously. And finally, CC will submit all the data it got to AM.

4 Conclusion and future work

With the widely deployment of containers and PaaS in public and private cloud, it is becoming increasingly important to produce a theory for forensics of container in PaaS. The container forensics framework we proposed in this paper has considered the circumstance of multiple hosts, which means it can be effective to containers not just on one machine, but also on PaaS platform. The way we treat the container and the forensics data is very different from the existing virtualization forensics. DCFF can extract data from a running container without letting it knows and send those data to forensics center in a low-cost way. The strategy here is also very convenient to investigators and hosts. Because the investigator can operate on single node and the host doesn't need to do any unnecessary work. In the future we plan to study a PaaS forensics framework with stronger compatibility base on DCFF.

References

[1] Dua Rajdeep, Raja A Reddy, Kakadia Dharmesh, Virtualization vs containerization to support PaaS[C], IEEE International Conference on Cloud Engineering, 2014, pp 610-614.

[2] Anderson Charles, Docker[J], IEEE Software, May 2015,pp 102-105.

[3] Ding, Weimin, Ghansah Benjamin; Wu Yanyan, Research on the Virtualization technology in Cloud computing environment[J], International Journal of Engineering Research in Africa, 2016, vol. 2, pp 191-196.

[4] Keyun Ruan, Joe Carthy, Tahar Kechadi, Mark Crosbie. Cloud Forensics[J]. IFIP Advances in Information and Communication Technology, 2011, vol. 361, pp 35-46.

[5] ZHOU G. Research on Scene Migration of Computer Forensics in Could Computing Environment[D]. Wuhan: Huazhong University Science and Technology, 2011.

[6] XIE Ya-long, DING Li-ping, LIN Yu-qi, ZHAO Xiao-ke. ICFF: a cloud forensics framework under the IaaS model[J], Tongxin Xuebao/Journal on Communications, May 2013, pp 200-206.

[7] Patrick Tobi, Tahar KechadiVirtual, Virtual machine forensics by means of introspection and kernel code injection[C], 9th International Conference on Cyber Warfare and Security, 2014, pp 292-298.

[8] AUFS, http://http://aufs.sourceforge.net.

[9] Device-mapper, http://sourceware.org/dm

[10] Docker Registry 2.0, https://github.com/docker/distribution.