**Digital Forensics Incident Response - DFIR Docker**

**Docker Forensics**

In this article, I will be speaking on how to conduct digital investigations and also how to set up your docker environment for DFIR.

Docker is an open-source software for the administration and deployment of software containers. This understanding defines software containers as a runtime environment within an operating system that isolates processes or process groups from each other while using a single common kernel

Docker is already found widespread adoption in Agile software development due to the flexible and fast deployment model and the ability for rapid software development based on a large ecosystem. The high prevalence in combination with a high adaptation rate means that many servers are already being operated as Docker hosts on which many micro-services run in so-called “containers”.

Conducting investigations on Docker images is done just like any other regular system. Performing a dump of the hard disk and ideally, the main memory is created. Without considering the specifics of the Docker container it can result in incomplete results. There are various aspects that need to be considered when analyzing Docker hosts to provide a comprehensive forensic analysis. These aspects are:

* File Recovery and Accountability
* Recovery of Files of the R/W Layer
* File Carving
* File System Analysis

File Recovery and Accountability

Docker containers access files through specific file system drivers. The file system is not mapped to block devices as in traditional operating system environments but is based on layered file system based structures. The Layered File System has three main layers, they are Image layer, Container layer, and Container mount. When investigating there are some questions that need to be asked on a Docker host,

Which image provided the given file; Which containers used the given file; Was the file deleted at Container level.

Recovery of Files of the R/W Layer

When a file that was stored in the r/w layer of the container is deleted, this file (reminder: which is stored as a regular file in the host file system) is deleted in the file system of the host. Data of data which in other words is called Metadata is retained in the file system and how it can be recovered depends on the actual host file system (such as ext3, ext4, zfs etc ...)

There are two coming file recovery methods:

1. File Carving
2. File System Analysis

File Carving

File carving is a method used to linearly search a volume, disk image, or a file for characteristic patterns for the beginning and or end of files. This method can recover both allocated and unallocated space that has not been overwritten.

File System Analysis

Unlike file carving, the file system analysis uses management structures stored in the file system, such as the MFT (master file table) in the case of NTFS or the inode tables of the group descriptors in the case of ext file systems. Deleted files can be recovered depending on the file system.

If you’re using Docker, you may manually explore the copy-on-write diffs inside /var/lib/docker. But we can go one step further: Docker will capture all file modifications, since the container was started into a new layer on the copy-on-write file system. Simply commit an existing container, running or not, into a new image: docker commit $CONTAINER\_ID imagename. Committing a container into a new image does not otherwise modify the container in any way.

Memory

Containers offer additional benefits in terms of memory isolation. Containers are not a first-class primitive in linux; they are an abstraction on top of multiple subsystems, such as process namespaces and cgroups. Processes running inside a container manage memory the same as any other process. By default, processes inside a container may not interact or interfere with memory controlled by any process outside the container. This provides the following characteristic: if you capture all of the memory for each process in a container, you have captured all allocated memory for that container

**Malware Analysis using SysAnalyzer**

Docker Containers for Malware Analysis

Malware analysis apps as Docker containers offer several benefits. Apps with conflicting dependencies can run on the same host. No unwanted files lying around after you’re done with the analysis.

Some level of isolation around the analysis application container.

SysAnalyzer is an application that allows for quick analysis of malware by observing its activities in different stages of the system.

Before starting the ‘malicious sample’, the software creates a snapshot of the current state of our environment, which after starting the malware, is the basis for determining changes in the system.

The main components of SysAnalyzer work off of comparing snapshots of the system over a user-specified time interval. The reason a snapshot mechanism was used compared to a live logging implementation is to reduce the amount of data that analysts must wade through when conducting their analysis. By using a snapshot system, we can effectively present viewers with only the persistent changes found on the system since the application was first to run.

Remember that any malware analysis should be carried out in a controlled environment. Running a virtual machine on a private or office computer, which we work on every day is not the best idea.

Reference: <https://dl.acm.org/doi/abs/10.1145/3058060.3058085>

<https://static.ernw.de/whitepaper/ERNW_Whitepaper64_IncidentForensicDocker_signed.pdf>

<https://zeltser.com/media/archive/docker.pdf>

<https://www.stackrox.com/post/2017/08/csi-container-edition-forensics-in-the-age-of-containers/>

Contributed by: Daval Gregory from Jamaica