Security Evaluation on Hadoop YARN containers

Peipei Wang
pwang7@ncsu.edu

Rui Shu
rshu@ncsu.edu

April 22, 2015

Abstract

Hadoop is a widely used computing framework and Docker is a popular container technology that emerged last year and then quickly adopted by a lot of commercial companies. This paper examines the combination of Hadoop with Docker container, explores how Docker container works in Hadoop, and analyzes the security issues solved by adopting Docker container and the security issues not solved or introduced by Docker container.

Our analysis shows that YARN NodeManager executes Docker command to launch multiple Docker containers in order to perform Hadoop MapReduce jobs. This method on one hand provides isolated MapReduce Job runtime environment, protecting secrecy and integrity of the executed programs to a large extent. This is due to the reason that the containers limit the damages caused by arbitrary code execution during MapReduce job execution time, and prevents attackers on the compromised node from adding errors into job computations. It is also because that the launched containers will be killed after it finishes the assigned tasks. On the other hand, Docker container could not protect data integrity and secrecy of HDFS. This is because Docker container requires Hadoop to run in the non-secure mode, disabling user authorization and credentials. Docker container also introduces other security risks, namely the problem of setuid. The permission of Docker has to be changed to enable launching Docker containers.

1 Introduction

Hadoop [**?**] is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, while each offering local computation and storage. It has been deployed by lots of companies, such as Facebook, Amazon, Adobe, Ebay, etc. There are also many other softwares built on top of Hadoop(e.g., Hive [**?**, **?**], Hbase [**?**], Pig [**?**], ZooKeeper [**?**]).

At the beginning, Hadoop was built with security in mind. While a lot of work on Hadoop has focused on improving its efficiency, people are gradually paying attention to its security concerns and building security modules for Hadoop. However, there exists no structured evaluation of security vulnerabilities in Hadoop considering the security enabled module.

Docker [**?**] combines an easy-to-use interface to Linux containers with easy-to-construct image files for those containers. In short, Docker launches very light weight virtual machines. Containers take advantage of the Linux kernels ability to create isolated environments that are often described as a "chroot on steroids [**?**]. Containers and the underlying host share a kernel. However, each container is assigned its own mostly independent runtime environment as with the help of Control Groups (cgroups) [**?**] and namespaces. Each container receives its own network stack and process space, as well as its instance of a file system. However, at the moment, Docker does not provide each container with its own user namespace, which means that the launched user is only root and it offers no user ID isolation.

The latest activity of Hadoop community is the involvement of Docker through Docker Container Executor(DCE) [**?**]. DCE allows the YARN [**?**] NodeManager to launch YARN containers into Docker containers. These containers provide a custom software environment in which the user's code runs, isolated from the software environment of the NodeManager. Docker for YARN provides both consistency and isolation because all YARN containers will have the same software environment and there is no interference with whatever is installed on the physical machine.

In this paper, we examined the combination of Docker container with Hadoop from the viewpoint of Hadoop security. Specifically, we want to answer the following three questions:

- How does container work in Hadoop?
- What are the security problems have been solved by YARN container?
- What are the security problems have not been solved yet?

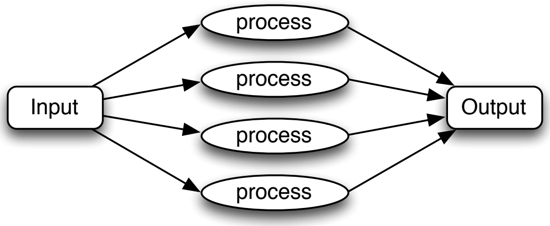

Figure 1: A high-level architecture of our approach

We went through the current security concerns, namely arbitrary code execution, malicious user impersonation, existing compromised nodes; evaluated the influence of the combination on security modules; and explored introduced risks by evaluating the requirements for using Docker container.

The rest of the paper is organized as follows. We introduce Hadoop framework and Docker container in Section 2. In Section 3, we discuss the security vulnerabilities and also security modules of Hadoop. Section 4 presents our experiments to evaluate Docker container in Hadoop. Section 5 discusses related work. Finally, the paper concludes in Section 6.

2 Overview

This section presents details about Hadoop and Docker.

2.1 Hadoop

Hadoop is a large-scale data processing framework that is designed to scale up from a single server to thousands of machines, with very high degree of fault tolerance. Rather than relying on high-end hardware, the resiliency of these clusters comes from the software's ability to detect and handle failures at the application layer.

HDFS(storage) and MapReduce [?] (processing) are the two core components of Hadoop. The most important aspect of Hadoop is that both HDFS and MapReduce are designed with each other in mind and each are co-deployed such that there is a single cluster and thus provides the ability to move computation to the data not the other way around. Thus, the storage system is not physically separate from a processing system.

The main components of HDFS are NameNode, DataNodes, and Secondary NameNode. NameNode is the master of the system. It maintains the name system (directories and files) and manages the blocks which are present on the DataNodes. DataNodes are the slaves which are deployed on each machine and provide the actual storage. Secondary NameNode is responsible for performing periodic checkpoints. Notice that Secondary NameNode is not a back up server of Namenode.

The main components of MapReduce are JobTracker and Tasktracker. JobTracker is the master of the system which manages the jobs and resources in the cluster (TaskTrackers). TaskTrackers are the slaves which are deployed on each machine. They are responsible for running the map and reduce tasks as instructed by the JobTracker.

However, Hadoop 1.x is not scalable enough and has many limitations. There is no horizontal scalability of NameNode and it does not support NameNode High Availability. It can only support up to 4,000 nodes per cluster. Map and Reduce task slots are static. The JobTracker is responsible for both resource management and job scheduling, so it is so overburdened that it becomes the bottleneck of Hadoop.Therefore, Hadoop 2.x or Hadoop YARN is developed to handle the scalability problem in Hadoop. The fundamental idea of Haoop 2.x is to split up the two major functionalities of the JobTracker, resource management and job scheduling, into separate daemons. The following figure shows the architecture of Hadoop 2.x.

HDFS does not change in Hadoop 2.x. The YARN component is introduced to replace JobTracker. It has three components: Resource Manager,Node Manager, and Application Master. ResourceManager is the master that arbitrates all the available cluster resources and thus helps manage the distributed applications running on the YARN system. The NodeManager is YARNs per-node agent, and takes care of the individual compute nodes in a Hadoop cluster, such as keeping up-to date with the ResourceManager, overseeing containers life-cycle management; monitoring resource usage (memory, CPU) of individual containers, etc. The ApplicationMaster is, in effect, an instance of a framework-specific library and is responsible for negotiating resources from the ResourceManager and working with the NodeManager(s) to execute and monitor the containers and their resource consumption. It has the responsibility of negotiating appropriate resource containers from the ResourceManager, tracking their status and monitoring progress.

2.2 Docker in Hadoop

Docker is a Linux-based system that makes use of LXC [?] that automates the deployment of applications inside software containers, by providing an additional layer of abstraction and automation of operating system level virtualization on Linux. Docker uses resource isolation features of the Linux kernel such as cgroups, and kernel namespaces to allow independent ”containers” to run within a single Linux instance, avoiding the overhead of starting virtual machines.

Hadoop YARN container represents a collection of physical resources, including CPU cores, disk along with

RAM. Docker container can perfectly implement this concept. By using Docker containers, resources can be isolated, services restricted, and processes provisioned to have a private view of the operating system with their own process ID space, file system structure, and network interfaces. Docker Container Executor (DCE) is thus developed. Other container executors are Default Container Executor and Linux Container Executor which is implemented with cgroups.

3 Hadoop Security

In this section, we discuss the security vulnerabilities and also security modules of Hadoop, including arbitrary code execution, malicious user impersonation, and compromised nodes.

3.1 Arbitrary code execution

Users submit jobs to Hadoop in the form of MapReduce jobs, and Hadoop executes the code without inspection. Arbitrary code execution is very common in Hadoop 1.x because malicious users can submit job executing with permissions of the TaskTracker which Task Tracker is an independent Java process.

Since it is difficult to inspect MapReduce source code, the best way is to restraint the damages of arbitrary code execution. YARN container could isolate tasks and limits task privilege. It has three implementations of container executor: 1) Default Container Executor, 2)Linux Container Executor, and 3)Docker Container Executor. The current resource types in container are 1)memory: physical/virtual memory, ratio and 2) CPU: percentage, vcores.

3.2 Malicious user impersonation

A big security concern of Hadoop is insufficient authentication. Hadoop does not authenticate users, and does not authenticate services. Another security concern is no integrity or secrecy protection of data and messages. The default network transport is insecure and there is no encryption over message communication.

Because of these two concerns, Hadoop could allow a malicious user to impersonate other user accounts to access data or submit MapReduce jobs. This is especially common in Hadoop ecosystem (e.g., Oozie [?], Hbase).

To handle the problem of malicious user impersonation, Hadoop builds two modes to differentiate security level and requirements: non-secure mode, and secure mode.

In non-secure mode, Hadoop has no authentication, end user accounts which send out communication messages with Hadoop are regarded as Hadoop user accounts by default. For example, different users on different systems with same user name are regarded as same Hadoop user.

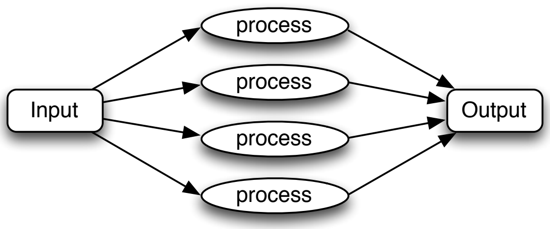

Figure 2: A high-level architecture of our approach

In secure mode, Hadoop applies authentication and data confidentiality. Kerberos Security module is added, and Hadoop RPC and HTTP communication messages are encrypted. The security framework of Hadoop secure mode is shown as below.

As seen in the figure, kerberos authentication module has the authentication server and the ticket granting server. Tokens are used to control access to the jobs and data blocks inside HDFS.There exists three types of tokens for HDFS access namely: delegation token, block access token, and job token.

Hadoop clients use RPC which is a Hadoops library to access most Hadoop services. In insecure versions of Hadoop, the users login name is determined from the client OS and sent across as part of the connection setup. For authenticated clusters, all RPCs connect using simple authentication and secure layer(SASL). Data transfer Protocol is used when reading or writing data to HDFS, by clients.

3.3 Compromised nodes

Compromised nodes are common in cloud system which means an attacker who has compromised just one node could introduce arbitrary errors into computations, so that the integriy of MapReduce job results could be compromised as well.

This problem is described in Hatman:Intra-cloud Trust Management for Hadoop in Cloud Computing , 2012. Data and computation integrity and security are major concerns for users of cloud computing facilities. Many production-level clouds optimistically assume that all cloud nodes are equally trustworthy when dispatching jobs; jobs are dispatched based on node load, not reputation. This increases their vulnerability to attack, since compromising even one node suffices to corrupt the integrity of many distributed computations. Hatman is the first full-scale, data-centric, reputation-based trust management system for Hadoop clouds. Hatman dynamically assesses node integrity by comparing job replica outputs for consistency.

Since containers provide isolated environment, it could at some extent prevent malicious user on the compromised nodes to tamper MapReduce job results.

4 Design

Protocol/Architecture/Design/. . .

5 Evaluation

5.1 Experiment platform

The experiment platform is shown in the table.

5.2 Hadoop setup with Docker configuration

We suppose that Hadoop is already installed in Pseudo-Distributed mode, and Docker is also installed. The path of docker command is /usr/bin/command.

The first step is to download docker image for Hadoop docker container. The command is: sudo docker pull sequenceiq/hadoop-docker:2.6.0

Next, add the following properties to yarn-site.xml:

In order for Hadoop to use Docker container, we need to change the permission of docker command with sudo chmod u+s /usr/bin/docker.

Go to the directory of Hadoop installtion, and start Hadoop with sbin/start-dfs.sh and sbin/start-yarn.sh.

The results of jps command will present you the running processes: NameNode, SecondaryNameNode, DataNode, ResourceManager, NodeManager.

To test the usage of Docker container in Hadoop, we use one Hadoop example program teragen to generate 1000 bytes of data. The command is as follows:

After the job is launched to execute, docker ps command shows that there are 3 containers that are launched to run the Teragen program. The 3 containers terminate after the job is finished.

5.3 Example attack

Since docker is being able to run by any user, use docker stop containerID could terminate the containers which are running MapReduce jobs. In this way, the malicious user could introduce failures to jobs. The screenshot is shown as below.

6 Discussion

The security concerns regarding to Docker containers in Hadoop are as follows:

6.1 Permissions of container executor

The container-executor program (in this paper, it is docker command) must have a very specific set of permissions and ownership in order to function correctly. In particular, it must:

- Be owned by root
- Be owned by a group that contains only the user running the YARN daemons
- Be setuid
- Be group readable and executable

The default user to run Docker is only root, and in order to run DockerExecutorContainer, the command /usr/bin/-docker has to be changed to be setuid. With setuid, any user of the machine can run docker command. This not only compromise the normal Docker container runtime, but also could introduce failures to MapReduce job execution.

6.2 Non-Secure mode

Docker Container Executor runs in non-secure mode of HDFS and YARN. It will not run in secure mode, and will exit if it detects secure mode.

However, we have discussed the details about non-secure mode, and talked about the malicious impersonation problem in non-secure mode.

6.3 Running Hadoop in Docker Container

There is another way to combine Docker container with Hadoop, namely, running Hadoop cluster inside Docker containers.

Docker now is building tools for launching cloud servers, systems for clustering, and a wide range of functions. Instead of running Hadoop processes on the host machine, launching Docker containers and restraint Hadoop inside container is also an interesting method. The practicality to use this method is unknown. So is its security concerns.

Expected results: YARN containers could not stop aggressive network/disk usage YARN containers implemented with Docker cannot stop user manipulating data illegally Solutions for containers to manage network or disk resources Solutions for containers (Docker) in the secure mode

7 Related Work

Hatman [**?**] is the first full-scale, data-centric, reputation-based trust management system for Hadoop clouds. Hatman dynamically assesses node integrity by comparing job replica outputs for consistency.

Data and computation integrity and security are major concerns for users of cloud computing facilities. Many production-level clouds optimistically assume that all cloud nodes are equally trustworthy when dispatching jobs; jobs are dispatched based on node load, not reputation. This increases their vulnerability to attack, since compromising even one node suffices to corrupt the integrity of many distributed computations.

With the growing use of Hadoop to tackle big data analytics involving sensitive data, a Hadoop cluster could be a target for data exfiltration [**?**],

long persistent attack: Advanced Persistent Threat (APT)

By implementing open standards based Trusted Computing technology at the infrastructure and application levels; a novel and robust security posture and protection is presented.

SecureMR [**?**], a practical service integrity assurance framework for MapReduce. SecureMR consists of five security components, which provide a set of practical security mechanisms that not only ensure MapReduce service integrity as well as to prevent replay and denial of service (DoS) attacks, but also preserve the simplicity, applicability and scalability of MapReduce.

MapReduce suffers from the integrity assurance vulnerability: it takes merely one malicious worker to render the overall computation result useless. Existing solutions are effective in defeating the malicious behavior of non-collusive workers, but are futile in detecting collusive workers.

The basic idea of VIAF [**?**] is to combine task replication with non-deterministic verification, in which consistent but malicious results from collusive mappers can be detected by a trusted verifier.

8 Conclusion

7.1 How does container work in Hadoop ? 7.2 What are the security problems have been solved by YARN container ? 7.3 What are the security problems have not been solved yet ?