Security Evaluation on Hadoop YARN containers

Peipei Wang
pwang7@ncsu.edu

Rui Shu
rshu@ncsu.edu

May 3, 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 the Docker container and those issues that are not solved or introduced by the Docker container.

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

1 Introduction

Hadoop [19] is a framework that allows distributed processing of large data sets across clusters of computers with simple MapReduce programming models. It is designed to scale up from single servers to thousands of machines, while each offering local computation and storage. Nowadays, it has been deployed by lots of companies (e.g., Facebook [12], Amazon [2], Ebay [8], Yahoo [13]). There are also many other applications built on top of Hadoop(e.g., Hive [43, 11], Hbase [9], Pig [16], ZooKeeper [20]).

Hadoop was targeted to use on private clusters at the beginning and thus was not built with security in mind. The focus of Hadoop at that time is improving its efficiency. However, with the popularity of cloud computing, there emerges the requirement of running Hadoop on shared multi-tenant machines with sensitive data. The previous Hadoop could not meet this demand. Therefore, a secure mode for Hadoop is proposed and implemented in current version of Hadoop. The secure mode [7] consists of user authentication to protect Hadoop from unauthorized access to HDFS [38] data, data confidentiality to protect transferred data between Hadoop nodes and clients from being intercepted by attackers, and delegation tokens to protect Hadoop from unauthorized access to MapReduce tasks.

However, secure mode has a disadvantage. It does not provide security isolation for MapReduce jobs. Hadoop executes user-submitted code without inspection. Malicious use could submit jobs with code to compromise Hadoop nodes. Therefore, containerization technology is incorporated into Hadoop. The most recently adopted technology is Docker [3]. Docker provides resource isolation for MapReduce jobs in Docker containers. Each container is an independent runtime environment with its own view of namespace. The compromised Docker container could neither affect other Docker containers nor the host system which runs the Docker services. Thus applying the Docker container into Hadoop through the Docker Container Executor(DCE) [4]could limits the damage of malicious code. There are two security problems with DCE. One big problem is that it is incompatible with Hadoop secure mode [4]. The reason lies in user authentication enabled by Kerberos in Hadoop secure mode. All nodes and users running Hadoop services are registered as Kerberos principals so that Hadoop services could read their permissions in authorization configuration files. Since the Docker container is created and terminated dynamically, it is very difficult for Hadoop secure mode to register the Docker container and define their permissions. The other problem is introduced by a general security problem for Docker. The entire stack of Docker technology runs as a single process and requires root access to the machine. To use the Docker container, the permission of running

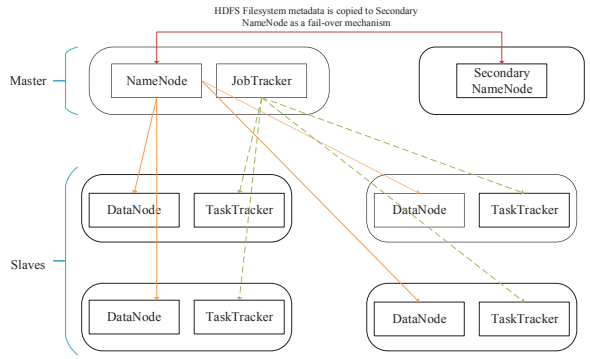

Figure 1: Hadoop 1.x Architecture Overview

Hadoop has to escalate to root.

In this paper, we investigate the implication of the Hadoop's secure mode incompatibility with Docker. Specifically, we want to answer the following three questions:

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

We discuss the current security concerns (i.e., arbitrary code execution, malicious user impersonation, and existing compromised nodes), and evaluate the influence of the combination on security modules. We then explore the introduced risks by evaluating the requirements for using the Docker container. Finally, we discuss the attack surface trade-offs and propose designs that is used to solve the problems that we have mentioned above.

The rest of the paper proceeds 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 the trade-off and proposed designs. Section 6 presents the related work. Finally, the paper concludes in Section 7.

2 Overview

In this section, we first talk about the architecture and components in Hadoop 1.x and Hadoop 2.x. We point out issues unsolved in Hadoop 2.x. Then we introduce Docker, compare Docker with other container technologies, and discuss the impact of incorporating the Docker container into Hadoop.

2.1 Hadoop

Hadoop is an open-source software for large-scale data processing with commodity machines. It could scale up from a single server to thousands of machines, with very high degree of fault tolerance. Since Hadoop assumes each machine is prone to failures, the resiliency of Hadoop clusters comes from the software's ability to detect and handle failures at the application layer instead of relying on high-end hardware,

Hadoop Distributed File System(HDFS) and MapReduce [25] are the two core components of Hadoop. HDFS is a distributed file system that provides high-through access to application data while MapReduce is a framework for parallel processing of large data sets. Figure 1 presents the architecture for Hadoop 1.x. The main components of HDFS are the NameNode, the DataNodes, and the SecondaryNameNode. The NameNode is the master of the system which maintains the name system (directories and files) and manages the blocks which are stored on the DataNodes. The DataNodes are the slaves which are deployed on each machine and provide the actual storage. The SecondaryNameNode is *not* a back up server of the Namenode, but instead is responsible for performing periodic checkpoints for the HDFS system file(i.e.,fsimage). The main components of MapReduce are the JobTracker and the Tasktrackers. The JobTracker is the master of the system which manages the jobs and resources in the cluster. The TaskTrackers are the slaves which are deployed on each machine. They are responsible for running the Map and Reduce tasks under the instruction from the JobTracker.

However, Hadoop 1.x is not scalable enough and has many limitations. Firstly, there is no horizontal scalability of the NameNode and it does not support the NameNode High Availability. Specifically, it can only support up to 4,000 nodes per cluster and there is always only one NameNode server. Secondly, the JobTracker becomes the bottleneck of Hadoop because it is responsible for both resource management and job scheduling and is easily to become overburdened. Finally, the storage system (i.e., HDFS) is not physically separated from the data processing system (i.e., MapReduce). Both HDFS and MapReduce are designed with each other in mind and each are co-deployed. It takes advantages of data locality and provides the ability to move computation to the data not the other way around.

Hadoop 2.x is developed to handle the scalability problem in Hadoop with a new component called Hadoop YARN (Yet Another Resource Negotiator) [46]. Haoop 2.x splits up the two major functionalities of the JobTracker, resource management and job scheduling, into separate daemons. HDFS does not change in Hadoop 2.x. The YARN component is introduced to replace Job-

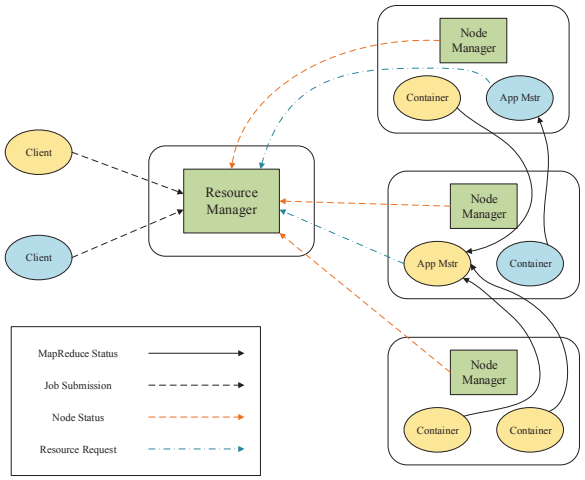

Figure 2: Hadoop YARN components

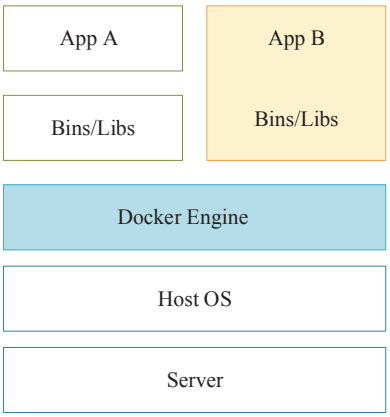

Figure 3: Docker

Tracker. As shown in Figure 2, YARN has three types of components: the ResourceManager, the NodeManager, and the ApplicationMaster. The 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 NodeManagers 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. As discussed above, Hadoop 2.x only solves the second problem of Hadoop 1.x. The problems related to HDFS are not solved yet.

2.2 Docker in Hadoop

Docker is an open platform for developing, shipping, and running applications. It is a Linux-based system 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 namespace to allow independent ”containers” to run within a single Linux instance, avoiding the overhead of starting virtual machines. With Docker we can separate our applications form the infrastructure and treat the infrastructure like a managed application.

As we can see from the Figure 3, the Docker Engine Container compromises just the application and its dependencies. It provides a way to run almost any application securely isolated in a container. It runs as an isolated process in user space on the host operating system, sharing the kernel with other containers, so that many containers are allowed to run simultaneously on the host.

Compared to Virtual Machines(VMs), Docker container provides more resource sharing, less isolation, and weaker security. Each VM has its own different OS and is portable to all type of OSes(e.g., Linux, Windows, OS X). But all Docker containers share the same host OS kernel and limited to Linux at the moment. While VMs are isolated through emulation and only hardware are shared, Docker containers are only isolated by system kernel virtualization. That means Docker containers only have different views of the operating system and some kernel subsystems and devices are shared across containers (e.g., SELinux, Cgroups, /sys, /proc/sys). The lightweight nature of containers means it enjoys the resource isolation and allocation benefits of virtual machines but is much more efficient, and suffers the security vulnerabilities brought by the resource sharing.

Docker used to use Linux Continaers(LXC) [27] to provide isolated Linux virtual environment for containers, but has dropped LXC and replaced LXC with its own libraries [5] for isolated environment. Both LXC and Docker are container technologies. Their difference is that LXC provides base OS containers while Docker provides single application virtualization based on containers. LXC containers can be treated as an OS and applications and services can be installed inside the container. However, Docker containers are limited to a single application and could not install other applications. This is because the Docker container is restricted to just one single process and could not support multiple applications with multiple processes. In summary, Docker is not just

an alternative for LXC and offers high-level services for application containers(e.g., layered file system isolation, image management).

OpenVZ [15] and FreeBSD Jails [14] also belong to containerization systems. FreeBSD Jails is a non-Linux containerization system and an enhancement to chroot. Besides resource isolation, FreeBSD Jails provides segregation of users, processes, and networks as well. Similar to LXC, OpenVZ is a Linux container solution. While OpenVZ, FreeBSD Jails, and Docker all provide application virtualizaiton, Docker also provides application portability through the Docker container images.

Docker has recently been officially integrated into Hadoop 2.0(YARN), where big data tasks such as MapReduce mappers and reducers run on individual containers. The marriage of Hadoop jobs with containers offers considerable benefits compared to running these jobs on VMs(in Hadoop 1.0). These include shorter provisioning time, more light weight migration, better performance isolation, bypassing virtual I/Os, and higher resource utilization since a single machine can accommodate more containers than VMs.

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 modules of Hadoop and the according security vulnerabilities: 1) arbitrary code execution; 2) malicious user impersonation; and 3) compromised nodes.

3.1 Arbitrary code execution

Arbitrary code execution occurs because Hadoop executes the code of the user-submitted jobs without inspection. It is common in Hadoop 1.x because MapReduce jobs submitted by malicious users could be executed with permissions of the TaskTracker while the TaskTracker 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.

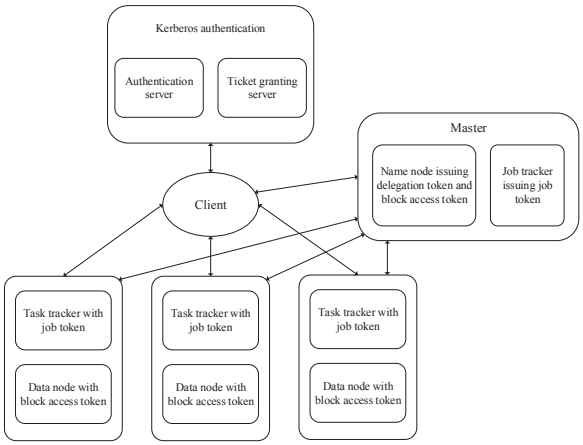

Figure 4: Security Framework

3.2 Malicious user impersonation

As discussed in Section 1, one 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 the 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 [1], Hbase [9]).

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.

In secure mode, Hadoop applies authentication, data confidentiality, and delegation tokens. Kerberos are required for Hadoop authentication in secure mode. Each Hadoop user and service are treated as Kerberos principals, and their permissions are defined in configuration files. To protect transferred data in Hadoop, data encryption are used for Hadoop block data transfer, Hadoop RPC and HTTP communication. The overall architecture of security mechanism is shown in Figure 4.

As seen in Figure 4, the 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: 1) delegation token which identifies a user to a particular HDFS service and addresses the problem of propagating authentication performed at job submission time to when the job is later executed; 2) block access token which gives a read or write permission to a HDFS block since the DataNodes do not have any access control list (ACL) mechanism to handle block access; and 3) job token which identifies a map reduce task its jobs.

Hadoop clients use Remote Procedure Call(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 set-up. 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 integrity of MapReduce job results could be compromised as well.

This problem is described in Hatman [29]. 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 Evaluation

In this section, we present an example which explores the root permission of Docker process. We first describe the experiment set-up, and then discuss the attack example results.

Name	Version
CPU	4 cores, Intel(R) Core(TM) i7-3770 CPU @ 3.4GHz
Memory	16 GB
Operating System	Ubuntu 14.04.2 LTS
Hadoop	Version 2.6.0, per
Docker	Version 1.5.0, build a8a31ef
Disk	Usage less than 90%

```
<property>
    <name>yarn.nodemanager.docker-container-executor.exec-name</name>
    <!--<value>docker -H=tcp://localhost:4243</value>-->
    <value>/usr/bin/docker</value>
    <description>Name or path to the Docker client. This is a required parameter. If this is
empty, user must pass an image name as part of the job invocation(see below).</description>
</property>
<property>
    <name>yarn.nodemanager.container-executor.class</name>
    <value>org.apache.hadoop.yarn.server.nodemanager.DockerContainerExecutor</value>
    <description>This is the container executor setting that ensures that all jobs are started with
the DockerContainerExecutor.</description>
</property>
```

Figure 5: Docker container configurations in yarn-site.xml

4.1 Hadoop setup with Docker configuration

The experiment platform is shown in the table 4.1. Hadoop has requirements of minimum resources(e.g., the free disk space should be larger than 10% of total disk space). The steps for the experiment are described as follows:

- Install Hadoop in Pseudo-Distributed mode.
- Install Docker and download Docker image for Hadoop the Docker container.
- Add the properties to yarn-site.xml shown in Figure 5.
- Change the permission of docker command with “sudo chmod u+s /usr/bin/docker in order for Hadoop to use Docker container.
- Go to the directory of Hadoop installation, and start Hadoop with “sbin/start-dfs.sh and “sbin/start-yarn.sh.
- Run “jps command and check the running processes: NameNode, SecondaryNameNode, DataNode, ResourceManager, and NodeManager.
- Run Hadoop example program Teragen to generate 1000 bytes of data with the command shown in Figure 6 in order to test the usage of the Docker container in Hadoop.

```
bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0.jar \
  teragen \
  -Dmapreduce.map.env="yarn.nodemanager.docker-container-executor.image-name=sequenceiq/
hadoop-docker:2.6.0" \
  -Dyarn.app.mapreduce.am.env="yarn.nodemanager.docker-container-executor.image-
name=sequenceiq/hadoop-docker:2.6.0" 1000 \
  teragen_out_dir
```

Figure 6: Hadoop command to run Teragen with the Docker container

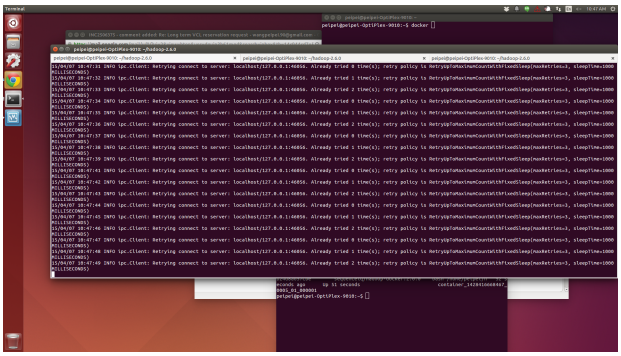

Figure 7: Attack experiment

- Run "docker ps command after the job is launched and finds that there are 3 containers that are launched to run the Teragen program. The 3 containers terminate after the job is finished.

4.2 Example attack

Figure 7 shows the results of launching an attack toward the Hadoop MapReduce job. This attack explores the security vulnerabilities introduced by Docker. Running containers with Docker implies running the Docker daemon which requires root privileges [18]. Therefore, in order to use the Docker daemon to create the Docker container, the user running Hadoop should have root privileges which can be achieved through changing the Docker daemon permissions with "setuid". In this way, the Docker command is able to be run by any user. Malicious users which do not have root privileges can use the command "docker stop containerID to terminate the Docker containers that are running Map or Reduce tasks. Terminating the tasks unexpected could lead to MapReduce job failures as shown in the Figure.

5 Discussion and Design

In this section, we propose solutions to the two existing security problems with Docker and discuss the problems with these solutions.

One proposed solution for requirement of root privileges of Docker daemon is to enable non-root users to run Docker commands [6]. The main idea is to create the *docker* group, configure docker as a system service, and add the users which should have Docker access into docker group. In this way, only trusted users belonging to docker group can run Docker commands and thus it prevents unauthorized users from interrupting the MapReduce job execution. The problem of this solution is that it increase the security risks. By default Docker daemon or service only listen to socket allowed by root users. By allows non-root users to access Docker, non-root users could gain root access on the host.

The other problem with Docker is its incompatibility with Kerberos. Kerberos mainly consists of the KDC key distribution center which is a centralized authentication service. It works in a realm which helps with the fully qualified domain name of the hosts. It contains a database which contains the principle identifiers with their secret keys. The main idea is as follows:

- The host requests access at Key Distribution Center(KDC). The authentication server at the KDC looks up for its identification in the local database and accordingly grants ticket.
- The NameNode checks for the details and if the client is valid accordingly issues the ticket to the client. The client and the DataNode exchange shared key for message exchange.
- The client then sends the ticket along with the shared key to the DataNode for further communication.

There are two difficulties of combining the Docker container and Kerberos. One is that Docker containers are created dynamically and every Docker container needs to communicate with KDC and then to be added to the database. Since there are large number of the Docker containers being created and terminated, the communication with KDC is costly and adds high overhead to the Hadoop Job execution. The second difficulty is that the permissions of the Docker containers needs to be added to the NameNode so that the NameNode could validate whether the Docker container has permissions or not. However, the permissions of the Docker containers depend on the Map or Reduce tasks assigned to them and it is very hard to determine their permissions before the containers are created.

One solution for this incompatibility problem is configuring all Docker containers with same Kerberos principals and same permissions. While using same Kerberos principals avoids the communication overhead and using same permissions allows unchanged configurations, it contradicts with the principle of least privileges [17]. The undifferentiated principals and permissions would grant the Docker containers unnecessary privileges and impose potential security risks to Hadoop nodes.

Another solution for the incompatibility problem is to discard Kerberos and to build new authorization framework that would be compatible with the Docker container. The main idea is as follows:

- Configure permissions of the Docker container when user submit the MapReduce jobs.
- The NameNode validates and records the permissions when launching the Docker containers to perform tasks.
- The NameNode compares the access request from the Docker containers with the recorded permissions and decide whether to grant or reject the request.

6 Related Work

Hadoop Secrecy and Integrity: With the growing use of Hadoop to tackle big data analytics involving sensitive data, data and computation integrity and security are major concerns for users of cloud computing facilities. Both Hadoop MapReduce and HDFS suffer from the integrity assurance vulnerability. 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 [29] 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.

Jason.et.al presented another work [24] that improves the integrity and secrecy of data for HDFS by incorporating Hardware Trust Mechanisms in Hadoop. This paper leverages the Trusted Platform Module(TPM) and technology based on the Trusted Computing Group (TCG) standards and pairs these technologies with HDFS storage at the infrastructure and application levels.

SecureMR [49] is 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.

While it takes merely one malicious worker to render the overall computation result useless, the existing solutions are effective in defeating the malicious behavior of non-collusive workers, but are futile in detecting collusive workers. Verification-based Integrity Assurance Framework(IAF) [47] is one framework to detect both non-collusive and collusive mappers by combining task replication with non-deterministic verification so that consistent but malicious results from collusive mappers can be detected by a trusted verifier.

Our work is different from these works. Instead of proposing frameworks for solving security problems, we explores the newest features in Hadoop, points out the potential risks and discusses the trade-off of different solutions.

Hadoop Security Framework and Evaluation: [50, 44] talk of solutions for Hadoop based set-ups. [35] focuses on access control within HDFS. It discusses a dependable security model and explains how it differs from existing federated systems. Further, it discusses the difference between securities in federated systems and general distributed systems.

[28, 37, 40, 42, 51] are surveys of the security in MapReduce Implementation Systems and big data security in general. Data security and privacy have also been discussed in [31, 45, 48]. These give a slightly higher level picture of the danger in the clouds in terms of data privacy. [21] speaks of a security management framework which can facilitate enforcement of security policies. It basically attempts to prevent a DOS attack.

[34, 10] talks of HDFS encryption in particular. It talks about the affordable low computation overhead of their proposed encrypted HDFS which basically has file protection in data blocks. It does AES data encryption/decryption for the same. While [33] discuss the hadoop security design in details, [23, 22, 30, 41] discuss about the attacks and threats on HDFS. They address the common security concern of Hadoop and propose an encryption scheme crypto framework for Hadoop.

SpongeFiles [26] talks of data skew in Hadoop MapReduce. It proposes a novel distributed-memory abstraction tailored to data processing environments. Although our work is not related to it, it gives an idea of the mitigation of disk spilling problem improving performance.

[39, 32] come up with Kerberos details and integration of Kerberos with Apache Hadoop. We have referred to it in details since we need to analyze the challenges of make Kerberos compatible with the Docker container.

[36] proposes a novel authentication mechanism. They suggest three approaches for security enhancement in Hadoop environment based on triangle properties. An analysis on the security level and complexity of these approaches has been presented in it.

Through the related work, we find that not only are security vulnerabilities existing, which have not been rigorously tested, quality of service and performance using security enhancements also remains a potential concern. Although there are a considerable amount of related work for improving Hadoop security, we still strongly feel that

security enhancement has been done very recently(e.g., adopting the Docker container technology into Hadoop) lacking a evaluation scheme. The discussion of reasons behind the benefits and weakness brought by the Docker container and the challenges are one of the missing parts of the current related work.

7 Conclusion

This paper presents the security evaluation on Hadoop YARN containers. Through analyzing Hadoop frameworks and the Docker container technology, we answer the following three questions:

- How does the Docker container work in Hadoop? Hadoop uses the Docker container through the Docker Container Executor and running Map or Reduce tasks in dynamically created or terminated Docker containers.
- What are the security problems have been solved by the Docker container? The Docker container is adopted in Hadoop to provide isolated runtime environment so that it can restrict the damages of arbitrary code execution.
- What are the security problems have not been solved yet? One problem is the Hadoop authorization because the Docker container can only be used in non-secure mode of Hadoop. The Docker container also introduces the problem of escalated privileges since the Docker daemon services requires the root access.

Besides this, this paper proves the two unsolved security problems related to the Docker container and performs one example attack to show the consequences of allowing non-root users to access Docker. We also propose solutions to these problems and discuss the trade-off of using these solutions.

References

[1] Apache oozie workflow scheduler for hadoop. http://oozie.apache.org/.

[2] Aws—amazon elastic mapreduce(emr)—hadoop mapreduce in the cloud. http://aws.amazon.com/elasticmapreduce/.

[3] Docker. https://www.docker.com/.

[4] Docker container executor. http://hadoop.apache.org/docs/r2.6.0/hadoop-yarn/hadoop-yarn-site/DockerContainerExecutor.html.

[5] Docker drops lxc as default execution environment. http://www.infoq.com/news/2014/03/docker_0_9/.

[6] Enabling non-root users to run docker commands. https://docs.oracle.com/cd/E52668_01/E54669/html/section_rdz_hmw_2q.html/.

[7] Hadoop in secure mode. http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/SecureMode.html#Kerberos_principals_for_Hadoop_Daemons_and_Users/.

[8] Hadoop-the power of the elephant. http://www.ebaytechblog.com/2010/10/29/hadoop-the-power-of-the-elephant/.

[9] Hbase - apache software foundation project home page. http://hbase.apache.org/.

[10] Hdfs encryption. http://blog.cloudera.com/blog/2014/06/projectrhino-goal-at-rest-encryption/.

[11] Hive - apache software foundation project home page. http://hive.apache.org/.

[12] How facebook uses hadoop and hive. http://hortonworks.com/big-data-insights/how-facebook-uses-hadoop-and-hive/.

[13] How yahoo spawned hadoop, the future of big data. http://www.wired.com/2011/10/how-yahoo-spawned-hadoop/.

[14] Jails-freebsd wiki. https://wiki.freebsd.org/Jails.

[15] Openvz-linux containers wiki. https://openvz.org/Main_Page/.

[16] Pig - apache software foundation project home page. http://pig.apache.org/.

[17] Principle of least privilege. http://en.wikipedia.org/wiki/Principle_of_least_privilege.

[18] Security-docker documentation. https://docs.docker.com/articles/security/.

[19] Welcome to apache hadoop. http://hadoop.apache.org/.

[20] Zookeeper - apache software foundation project home page. http://zookeeper.apache.org/.

[21] C. Basescu, A. Carpen-Amarie, C. Leordeanu, A. Costan, and G. Antoniu. Managing data access on clouds: A generic framework for enforcing security policies. In *Advanced Information Networking and Applications (AINA), 2011 IEEE International Conference on*, pages 459–466. IEEE, 2011.

[22] J. Cohen and S. Acharya. Towards a more secure apache hadoop hdfs infrastructure. In *Network and System Security*, pages 735–741. Springer, 2013.

[23] J. C. Cohen and S. Acharya. Towards a trusted hdfs storage platform: Mitigating threats to hadoop infrastructures using hardware-accelerated encryption with tpm-rooted key protection. *Journal of Information Security and Applications*, 19(3):224–244, 2014.

[24] J. C. Cohen and A. Subatra. Incorporating hardware trust mechanisms in apache hadoop. *IEEE, sl*, pages 978–1, 2012.

[25] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[26] K. Elmeleegy, C. Olston, and B. Reed. Spongefiles: mitigating data skew in mapreduce using distributed memory. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 551–562. ACM, 2014.

[27] M. Helsley. Lxc: Linux container tools. *IBM devloperWorks Technical Library*, 2009.

[28] R. C. Jose and S. Paul. Privacy in map reduce based systems: A review. 2014.

[29] S. M. Khan and K. W. Hamlen. Hatman: Intra-cloud trust management for hadoop. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 494–501. IEEE, 2012.

[30] H.-Y. Lin, S.-T. Shen, W.-G. Tzeng, and B.-S. Lin. Toward data confidentiality via integrating hybrid encryption schemes and hadoop distributed file system. In *Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on*, pages 740–747. IEEE, 2012.

[31] S. Madaan and R. K. Agrawal. Implementation of identity based distributed cloud storage encryption scheme using php and c for hadoop file system. In *Tier 2 Federation Grid, Cloud & High Performance Computing Science (RO-LCG), 2012 5th Romania*, pages 74–77. IEEE, 2012.

[32] O. OMalley. Integrating kerberos into apache hadoop. In *Kerberos Conference*, pages 26–27, 2010.

[33] O. OMalley, K. Zhang, S. Radia, R. Marti, and C. Harrell. Hadoop security design. *Yahoo, Inc., Tech. Rep*, 2009.

[34] S. Park and Y. Lee. Secure hadoop with encrypted hdfs. In *Grid and Pervasive Computing*, pages 134–141. Springer, 2013.

[35] Y. Reddy. Access control for sensitive data in hadoop distributed file systems. In *INFOCOMP 2013, The Third International Conference on Advanced Communications and Computation*, pages 72–78, 2013.

[36] G. Sadasivam, K. Kumari, and S. Rubika. A novel authentication service for hadoop in cloud environment. In *Cloud Computing in Emerging Markets (CCEM), 2012 IEEE International Conference on*, pages 1–6. IEEE, 2012.

[37] Z. Shen and Q. Tong. The security of cloud computing system enabled by trusted computing technology. In *Signal Processing Systems (ICSPS), 2010 2nd International Conference on*, volume 2, pages V2–11. IEEE, 2010.

[38] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, pages 1–10. IEEE, 2010.

[39] J. G. Steiner, B. C. Neuman, and J. I. Schiller. Kerberos: An authentication service for open network systems. In *USENIX Winter*, pages 191–202, 1988.

[40] S. Subashini and V. Kavitha. A survey on security issues in service delivery models of cloud computing. *Journal of network and computer applications*, 34(1):1–11, 2011.

[41] G. Sujitha, M. Varadharajan, Y. V. Rao, R. Sridev, M. Gauthaum, S. Narayanan, R. S. Raja, and S. M. Shalinie. Improving security of parallel algorithm using key encryption technique. *Information Technology Journal*, 12(12), 2013.

[42] C. Tankard. Big data security. *Network security*, 2012(7):5–8, 2012.

[43] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy. Hive: a warehousing solution over a map-reduce framework. *Proceedings of the VLDB Endowment*, 2(2):1626–1629, 2009.

[44] S. Tripathi, B. Gupta, A. Almomani, A. Mishra, and S. Veluru. Hadoop based defense solution to handle distributed denial of service (ddos) attacks. 2013.

[45] F.-H. Tseng, C.-Y. Chen, L.-D. Chou, and H.-C. Chao. Implement a reliable and secure cloud distributed file system. In *Intelligent Signal Processing and Communications Systems (ISPACS), 2012 International Symposium on*, pages 227–232. IEEE, 2012.

[46] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, et al. Apache hadoop yarn: Yet another resource negotiator. In *Proceedings of the 4th annual Symposium on Cloud Computing*, page 5. ACM, 2013.

[47] Y. Wang and J. Wei. Viaf: Verification-based integrity assurance framework for mapreduce. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 300–307. IEEE, 2011.

[48] L. Wei, H. Zhu, Z. Cao, X. Dong, W. Jia, Y. Chen, and A. V. Vasilakos. Security and privacy for storage and computation in cloud computing. *Information Sciences*, 258:371–386, 2014.

[49] W. Wei, J. Du, T. Yu, and X. Gu. Securemr: A service integrity assurance framework for mapreduce. In *Computer Security Applications Conference, 2009. ACSAC'09. Annual*, pages 73–82. IEEE, 2009.

[50] H. Zhou and Q. Wen. A new solution of data security accessing for hadoop based on cp-abe. In *Software Engineering and Service Science (ICSESS), 2014 5th IEEE International Conference on*, pages 525–528. IEEE, 2014.

[51] M. Zhou, R. Zhang, W. Xie, W. Qian, and A. Zhou. Security and privacy in cloud computing: A survey. In *Semantics Knowledge and Grid (SKG), 2010 Sixth International Conference on*, pages 105–112. IEEE, 2010.