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 [12] 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 [9], Amazon [2], Ebay [6], Yahoo [10]). There are also many other applications built on top of Hadoop(e.g., Hive [19, 8], Hbase [7], Pig [11], ZooKeeper [13]).

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 [5] consists of user authentication to protect Hadoop from unauthorized access to HDFS [18] 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

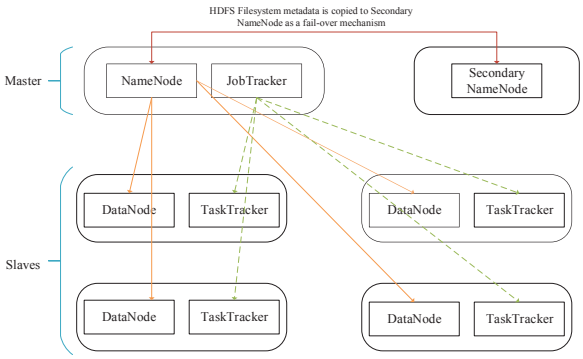

Figure 1: Hadoop Overview

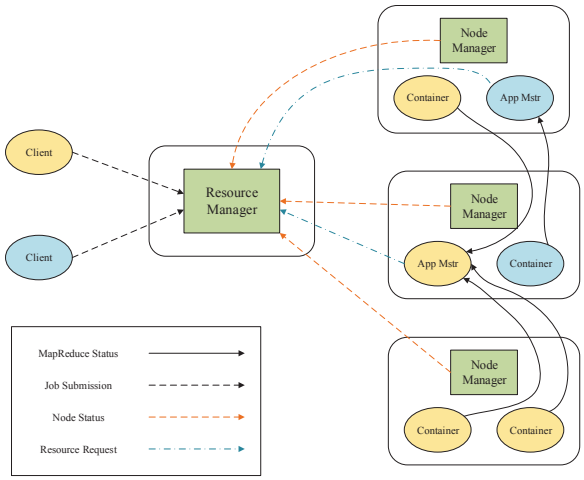

Figure 2: Hadoop YARN components

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 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?

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

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 [15] (processing) are the two core components of Hadoop. As it is depicted from Figure 1,the main components of HDFS are NameNode, DataNodes, and Secondary NameNode. NameNode is the master of the system which 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 not a back up server of Namenode, which is responsible for performing periodic checkpoints. 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, see Figure 2. 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. HDFS does not change in Hadoop 2.x. The YARN component is introduced to replace JobTracker. It has three components: ResourceManager, NodeManager and ApplicationMaster. 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.

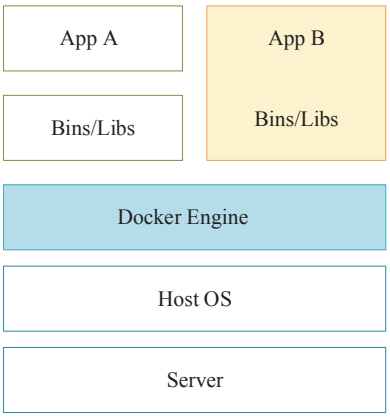

Figure 3: Docker

2.2 Docker in Hadoop

Docker is an open platform for developing, shipping, and running applications. It is a Linux-based system that makes use of LXC [16] 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 userspace on the host operating system, sharing the kernel with other containers, so that many containers are allowed to run simultaneously on the host. The lightweight nature of containers, which run without extra load of a hypervisor, means it enjoys the resource isolation and allocation benefits of virtual machines but is much more portable and efficient.

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 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

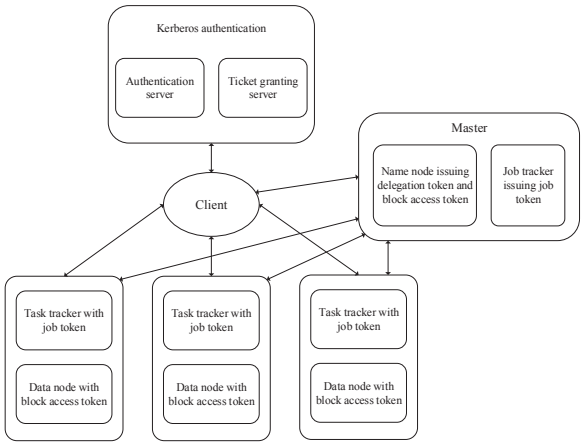

Figure 4: A high-level architecture of our approach

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 [1], 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.

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 [17]. 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

4.1 Experiment platform

The experiment platform is shown in the table.

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%

4.2 Hadoop setup with Docker configuration

We suppose that Hadoop is already installed in Pseudo-Distributed mode, and Docker is also installed. Firstly, we download docker image for Hadoop docker container.Then, we 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 installation, and start Hadoop with sbin/start-dfs.sh and sbin/start-yarn.sh.

```
<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>
```

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:

```
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
```

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.

4.3 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.

5 Discussion and Design

5.1 Experiment platform

The experiment platform is shown in the table.

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%

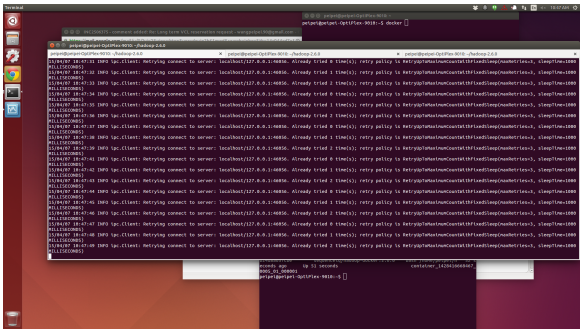

Figure 5: Attack experiment

5.2 Hadoop setup with Docker configuration

We suppose that Hadoop is already installed in Pseudo-Distributed mode, and Docker is also installed. Firstly, we download docker image for Hadoop docker container.Then, we add the following properties to yarn-site.xml:

```
<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>
```

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 installation, 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:

```
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
```

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.

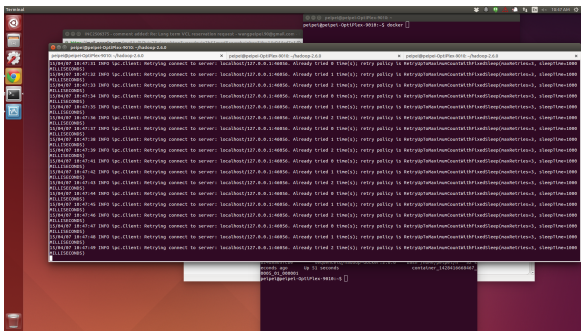

Figure 6: Attack experiment

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 Related Work

Hatman [17] 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 [14],

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 [21], 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 [20] 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.

7 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 ?

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. https://hadoop.apache.org/docs/r2.6.0/hadoop-yarn/hadoop-yarn-site/DockerContainerExecutor.html.

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

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

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

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

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

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

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

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

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

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

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

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.