Journal of Big Data

RESEARCH Open Access

Sandbox security model for Hadoop file system

Gousiya Begum[1,4*], S. Zahoor Ul Huq[2] and A. P. Siva Kumar[3]

*Correspondence:
gousiyabegum@gmail.com
[1] Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology, Gandipet, Hyderabad, India
Full list of author information is available at the end of the article

Abstract

Extensive usage of Internet based applications in day to day life has led to generation of huge amounts of data every minute. Apart from humans, data is generated by machines like sensors, satellite, CCTV etc. This huge collection of heterogeneous data is often referred as Big Data which can be processed to draw useful insights. Apache Hadoop has emerged has widely used open source software framework for Big Data Processing and it is a cluster of cooperative computers enabling distributed parallel processing. Hadoop Distributed File System is used to store data blocks replicated and spanned across different nodes. HDFS uses an AES based cryptographic techniques at block level which is transparent and end to end in nature. However cryptography provides security from unauthorized access to the data blocks, but a legitimate user can still harm the data. One such example was execution of malicious map reduce jar files by legitimate user which can harm the data in the HDFS. We developed a mechanism where every map reduce jar will be tested by our sandbox security to ensure the jar is not malicious and suspicious jar files are not allowed to process the data in the HDFS. This feature is not present in the existing Apache Hadoop framework and our work is made available in github for consideration and inclusion in the future versions of Apache Hadoop.

Keywords: HDFS, MapReduce, Fsimage, Hadoop, Kerberos

Introduction

Apache Hadoop has emerged as the widely used open source framework for Big Data Processing. Big Data processing is used in healthcare, social media, banking, insurance, good governance, stock markets, retail and supply chain, ecommerce, education and scientific research etc. to gain deep insights of the data, their associations and make better decisions [1]. Apache Hadoop addresses the two major challenges of Big Data viz. storage and processing. Data is stored in Hadoop using HDFS and processing through Map Reduce Programming. Apache Hadoop is a cluster of cooperative computers. The anatomy of Hadoop cluster can be easily understood from the Fig. 1.

The number of Data nodes can vary from cluster to cluster but every Hadoop cluster must contain Name node, Resource Manager and Secondary Name node. In Hadoop, files are stored using HDFS.

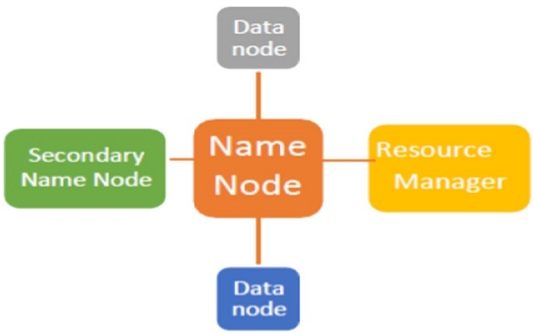

Fig. 1 Typical Hadoop cluster

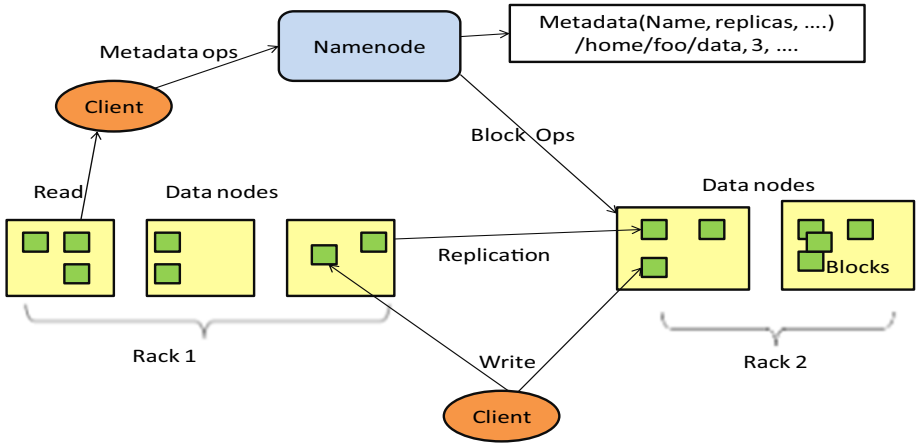

Fig. 2 HDFS architecture

Hadoop Distributed File System (HDFS)

In HDFS each file will be divided into blocks with default size of 128 MB each and these blocks are scattered across different data nodes with a default replication factor of three. Name node maintains the information about each file and their respective blocks in FSimage file. Hence Name node is considered to be Single Point of Failure because if Name node is down we cannot access any file blocks [2]. However Secondary Name node is always available as an immediate alternative whenever Name node goes down. Changes made in the data nodes are updated in FSimage using heartbeat messages from data nodes to Name node sent every 3 sec through which Name node gets updated immediately. HDFS architecture is described in Fig. 2.

Map Reduce programming model

Map Reduce is a Java based programming model facilitates distributed parallel processing and it is based on the principle of data locality. As part of principle of data locality, code is moved to the place where data blocks are available instead of moving data to code which reduces huge amount of disk access and improves performance to a great extent [3]. A Map Reduce job contains two programs called Mapper and Reducer. Mapper is a class where the logic to process the file is written and when

map reduce job executes, mappers will run simultaneously on each block. The results generated by the mappers are aggregated by the reducer. Execution of map reduce jobs are monitored by the Resource manager called Yarn. Typical Yarn architecture is described in Fig. 3.

A Map Reduce Job is deployed as a jar file and is submitted to the Job Tracker Daemon present in the Namenode. Since Namenode is aware of the location of input file blocks, Job tracker forwards the map reduce job to respective data nodes. The Task Tracker Daemon present inside data nodes will invoke a Java Virtual Machine (JVM) instance for every block of the input file and executes the code present in the mapper. Any legitimate user can issue the command to run a map reduce job on the Hadoop cluster to process any file and there is no mechanism of investigating the jar files for presence of any malicious code which may harm the files stored inside HDFS. Such map reduce jobs make the file system vulnerable to security attacks especially tamper of data, stealing of data and collection of metadata etc. In the next section we will present existing security features in Hadoop framework and also their limitations in detail.

Security features in Hadoop and its limitations

Since Hadoop was developed for internal use of Nutch search engine of Yahoo, security was not a major concern because of its limited use. However Hadoop framework was found to be comprehensive solution to Big Data processing problem and it was soon made open source under Apache License. Many new services were contributed to Apache Hadoop framework after it became open source. The amount of security features were very less in Hadoop 1.0 and security features incorporated in Hadoop 2.0 were also inadequate [4]. In this section we will present the existing security features of Hadoop framework along with their limitations.

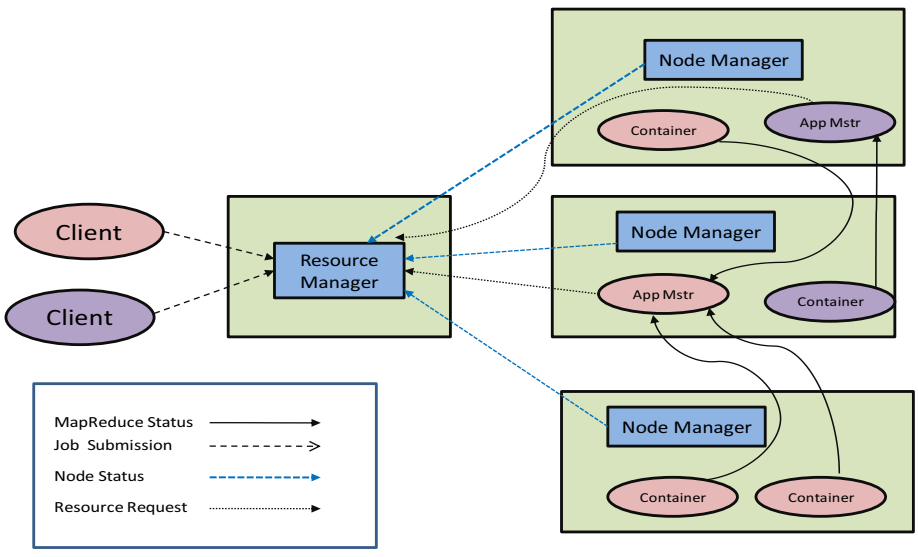

Fig. 3 Yarn architecture

Apache Knox

Many Hadoop services provide a graphical user interface able to run on certain port numbers. Apache Knox facilitates a gateway application for communication between REST API's and Hadoop Services. Some of the services that Apache Knox can connect to are Ambari, Hbase, Hive, Storm, Yarn, Oozie, WebHDFS etc. [5]. The purpose of Apache Knox is to enforce policies related to authentication, authorization, auditing and dispatching.

Apache Sentry

Apache Sentry is developed as an independent role base authorization service which can be integrated with various components of Hadoop. Apache Sentry provides the ability to enforce, control the privileges on data stored in HDFS for authenticated users only [6]. The Apache Sentry server holds the metadata and the actual authorization rules are defined in the respective client applications like Hive, Impala etc. Apache Sentry will only validate whether a particular user is allowed to perform certain operation on the data. For example if a user issues a Hive query to process Student table, Apache Sentry will check whether the user has enough privileges to access Student table or not. However Apache Sentry does not check whether an issued query is harmful to HDFS or not.

Apache Ranger

Apache Ranger is a centralized web based application consisting of authorization, policy administration, audit and reporting facilities. Authorized users can access the Apache Ranger web console to manage the security policies. These security policies are deployed as lightweight processes on Namenode [7].

Kerberos

Kerberos is implemented in Hadoop 1.0 and Hadoop 2.0. Kerberos is a conventional network authentication protocol implemented to authenticate every map reduce job that is issued by users. For every interaction with Namenode, the user and Namenode undergo a mutual authentication based on Kerberos ticketing mechanism. Only after proper authentication, the Namenode will check whether the given user is authorized to perform the operation which has been requested.

Limitations of existing security features in Apache Hadoop

1. Apache Knox acts only as gateway between Hadoop and other REST based applications. Performs only authentication.
2. Apache Sentry and Ranger are used for authorization purposes in different contexts as discussed in "Apache Sentry" and "Apache Ranger".
3. Kerberos is a conventional network authentication protocol integrate with Hadoop for authentication.
4. Most of the programs that run on HDFS are in the form of Jar files. In Hadoop there is no security mechanism in place to examine jar files for presence of any harmful code which is a very serious security.

5. Vulnerability: A legitimate user knowingly or unknowingly may execute a jar which contain harmful code that can tamper the data in the HDFS.
6. Apart from malicious jars, there are many other vulnerabilities in Hadoop Framework including distributed cache etc. [10].

More vulnerabilities in Apache Hadoop

Apache Hadoop Distribution contains basic services like HDFS, Map Reduce, PIG, HIVE etc. More services can be added by the admin using add service option in the Hadoop Admin console. However the services which are added to Hadoop can be vulnerable to security attacks and in turn make entire cluster vulnerable. One such example is a ransom ware attack made on mongoDB service which was running on a Hadoop cluster.

Most of the Hadoop security is based on 3 A's viz. authentication, authorization and audit. Beyond this Hadoop is vulnerable to all other security attack. Any authenticated user can run any jar file using Hadoop jar command. If the jar file contains any malicious code then it can destroy the whole cluster.

Apache Hadoop is also prone to malware attacks which was proved by recent malware attacks called DemonBot and XBash both of them were purposefully used for Distributed Denial of Service attacks (DDOS).

Most of the research carried out on Hadoop focused on authentication and cryptographic solutions. A legitimate user can also harm the file system and in turn corrupt the Hadoop cluster.

Method applied

Sandbox Security for Map Reduce jobs

There exists many security vulnerabilities in Hadoop Framework [8]. However the scope of this paper is confined to handling of map reduce jar files that execute on HDFS. In existing Hadoop framework, there is no mechanism in place to validate and ensure whether a given jar file contains any harmful code or not. In this section we demonstrate a security sandbox for map reduce jobs where malicious jar files are prevented from execution and accessing HDFS [9]. This is an improvement in security of HDFS where map reduce jars are validated first and if the jar file is found to be suspicious, it is prevented from execution, thus creating a sandbox for map reduce jobs. Implementation of sandbox security is described in Fig. 4.

Step 1. The .java files can be extracted from jar files using following command depicted in Figs. 5 and 6.

Step 2. In step 2 we need to define the characteristics based on which a jar is considered to malicious. If the jar is executed from any user account apart from root must have some limited access to the file system. Generally a non root user will execute map reduce jobs to process files present on the HDFS. In general a non root user may not be interested in metadata about the file which include location of file blocks, recent update time stamp etc. A non root user must not attempt to run administrator commands using File system API. Thus any such attempt is found in the mapper or reducer class then the jar file is considered to be suspicious and prevented from execution.

Fig. 4 Flowchart to represent implementation of Sandbox security

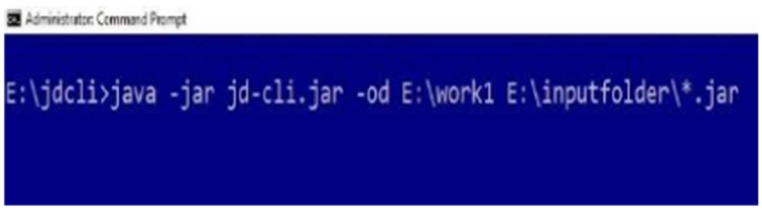

Fig. 5 Command to extract .java files from Jar

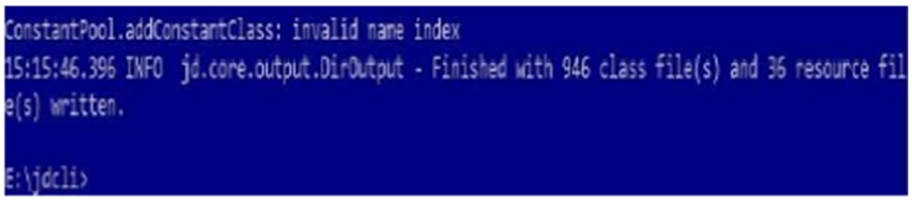

Fig. 6 Java files extracted to E:\work1 folder

Step 3. In order to check whether the extracted java files are suspicious or not, the extracted java files are read by a shell script which will check whether the map reduce job was issued by user with root privileges or not and searches for the HDFS commands which are already enumerated and considered not to be executed by a non root user.

Step 4. If any jar file is found to be suspicious then it is not allowed to execute on the file system. The pseudo code of the shell script is described in Fig. 7.

Step 5. Apache Spark has emerged as a lighting fast Big Data processing facility which can be deployed to run on HDFS. The sandbox security model can be applied on Spark program also to ensure prevention of inappropriate code from execution on HDFS.

```
If[[$EUID –ne 0]]; then
        echo “Not Root”
        word= ‘fsck’| ‘dfsadmin’|’FileSystem’|’balancer’|’expunge’|’chgrp|’cacheadmin’|
          ‘getconf’|’safemode’
        totalCount=0 counts=( )
        while X=read –r line; do
        count=$(grep –o “$word”<<<”$line” | wc –l)
                counts+-($count)
                ((totalCount+=count))
        done<BDSmapper.java
        exit 1
        if [count>0] then
                echo “suspicious jar file found”
        fi
```

Fig. 7 Pseudo code —Shell Script to probe java files

Step 6. In Hadoop framework, only map reduce code will be executed on the file system. No other application can directly access the file system and hence every application/service deployed on Hadoop cluster will be converted into a map reduce job which is executed on the file system. For example if a user writes a PIG script or HIVE query, they will also be converted into map reduce jobs and processed. Even on PIG and HIVE applications we can deploy our customized model to check for inappropriate access to HDFS leading to tampering of data or metadata.

Results and discussion

The key finding is that the map reduce jar file may contain malicious code and Hadoop jar command will not able to detect it and allows the user to execute the jar file on the input file. There is no mechanism to check a jar file for presence of malicious code. In our work we defined the characteristics by which a jar file can be considered to be malicious. Any attempt to read metadata, change file permissions, attempt to run dfsadmin commands etc. by non root user will be considered to suspicious and the jars containing such code inside their mappers and reducers will be treated as malicious. We have developed our own utility to extract the map reduce jar file using apache commons library. After extracting the java file from jar, we used a shell script to read the java files and search for the predefined, enumerated keywords and presence of any one word will make the java code suspicious and the jar is treated as malicious. The enumerated keywords include HDFS dfsadmin commands, fsck (to know information about file blocks),expunge, balancer etc. In existing Hadoop framework, a legitimate user can execute any map reduce job using Hadoop jar command and the Job tracker daemon present inside Namenode will forward the same to the respective data nodes where Task trackers will invoke a new instance of JVM for every file block to execute the map reduce job in a distributed parallel processing manner. Our work provides a sandboxing facility where unwanted or

harmful jar files can be prevented from executing on the file system. The shell script can be customized to filter jars based on requirement. This feature adds improvement to the execution environment of Hadoop framework and can be considered for inclusion in the future versions of Hadoop framework with suitable amendments.

Sno	Nature of Jar submitted to sandbox	Sandbox security outcome
1	Conventional word count jar	Allowed to execute
2	Jar containing mapper which used fsck command to enquire about file blocks	Not allowed to execute
3	Jar containing mapper which used expunge, name node safe mode enter	Not allowed to execute
4	NlineInputDemo is a jar file to process .csv file	Allowed to execute

In the Fig. 8, we have shown a map reduce job, the mapper code has shown clearly, we have not used any code shown in shell script above i.e. we have not used any commands to gather metadata related to file system nor we have written commands to delete file so this jar file has been executed successfully.

In the Fig. 9, we have shown one more map reduce job but it will not be executed successfully because we are using the commands shown in shell script. Hence the finding is sandboxing technique is used to secure Hadoop framework.

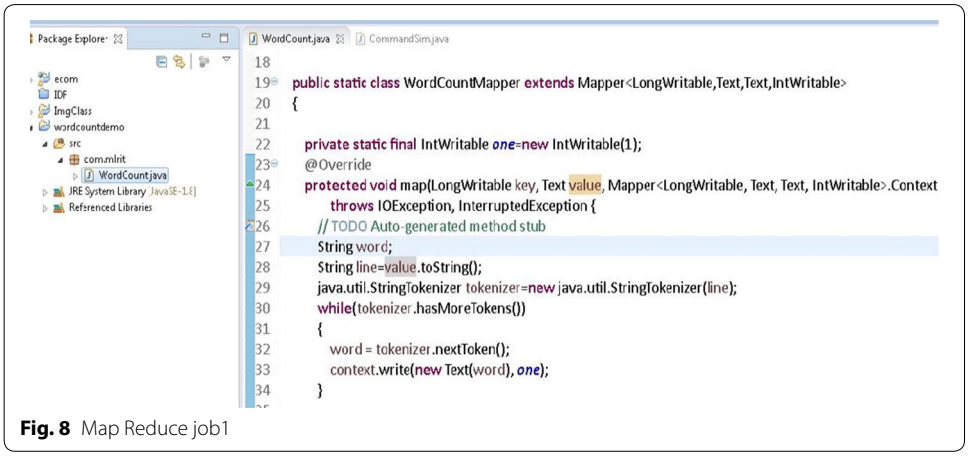

Fig. 8 Map Reduce job1

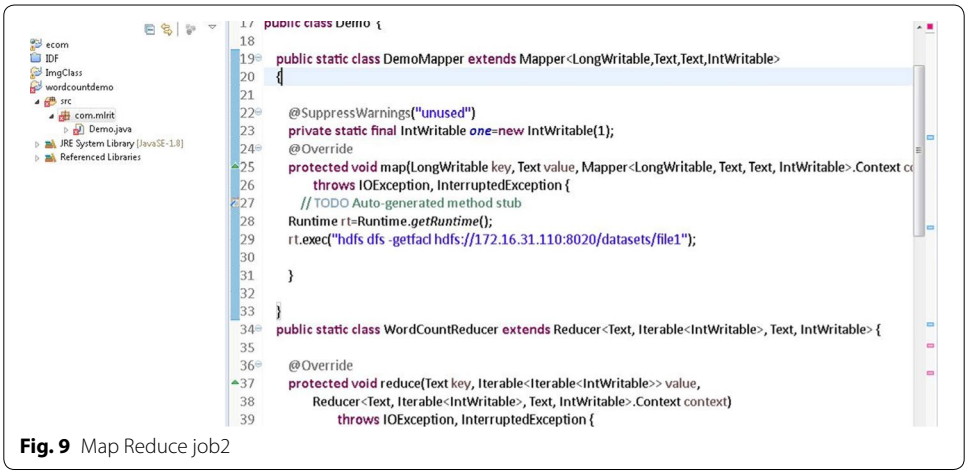

Fig. 9 Map Reduce job2

Conclusion

Sandboxing of map reduce jobs and other jar files will enhance the security of the HDFS by not allowing harmful jars to execute. Our work advances the field of study pertaining to Hadoop Security by addressing a problem which was not addressed in the previous literature. Our work will be made available in github for reference. Our sandbox security is customizable and can be enhanced to address more security vulnerabilities in Hadoop. Sandbox security can be extended so that it can detect presence of vulnerabilities in any new service that is added to Hadoop framework. However there are many other vulnerabilities exist in Hadoop framework. Hence there is scope for research in Hadoop Security. Some of the security vulnerabilities include distributed cache [11], ability of all users to access cluster home and retrieve metadata is also a vulnerability because all users need not see the files created by others and get metadata about file system. More number of other vulnerabilities do exist in Hadoop file system which can be explored and continue research in addressing such vulnerabilities.

Abbreviations

CCTV: Closed Circuit Television; HDFS: Hadoop Distributed File System; API: Application Programming Interface.

Acknowledgements

I would like to thank my guides for supporting my work and for suggesting necessary corrections.

Authors' contributions

GB: As part of my Ph.D. work, I am submitting the work I have done in the form of paper. SZUH: Supported me in compiling the paper. APSK: Suggested necessary amendments and helped in revising the paper. All authors read and approved the final manuscript.

Funding

No funding.

Availability of data and materials

If any one is interested in our work, we are ready to provide more details of the map reduce job which we have executed and the data processing techniques applied. However the data is used in our work, is freely available in many repositories.

Competing interests

The authors declare that they have no competing interests.

Author details

[1] Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology, Gandipet, Hyderabad, India. [2] Department of Computer Science and Engineering, GPREC, Kurnool, Andhra Pradesh, India. [3] Department of Computer Science and Engineering, JNTUA, Anantapuramu, Andhra Pradesh, India. [4] JNTU Anantapur, Anantapuramu, Andhra Pradesh, India.

Received: 4 April 2020 Accepted: 3 September 2020
Published online: 30 September 2020

References

1. Yao Y, Gao H, Wang J, Sheng B, Mi N. New scheduling algorithms for improving performance and resource utilization in Hadoop YARN clusters. IEEE Transactions on Cloud Computing. 2019.
2. Ge, Yi, et al., "File storage processing in HDFS", U.S. Patent No. 10,210,173, 19 Feb 2019.
3. Glushkova D, Jovanovic P, Abelló A. Mapreduce performance model for Hadoop 2. x. Information systems, vol 79. New Jersey: Elsevier; 2019. pp. 32–43.
4. Martis M, Pai NV, Pragathi RS, Rakshatha S, Dixit S. Comprehensive survey on Hadoop security. Emerging research in computing, information, communication and applications, vol 906. Springer: Singapore. AISC-. 2019. pp. 227–236.
5. Knox Gateway: REST API and Application Gateway for the Apache Hadoop Ecosystem. knox.apache.org. 2019.
6. Bhatal GS, Singh A. Big data: Hadoop framework vulnerabilities, security issues and attacks. Array open access, vol 1–2. New Jersey: Elsevier; Article 100002.
7. Awaysheh FM, Alazab M, Gupta M, Pena TF. Next-generation big data federation access control: a reference model. Future generation computer systems. New Jersey: Elsevier; 2020. pp 1–16.

8. Nellutla R, Mohammed M. Survey: a comparative study of different security issues in big data", emerging research in data engineering systems and computer communications. Springer, Singapore, AISC-volume 1054; 2020, pp. 247–257.
9. Langton, Asher J, et al. Configuring a sandbox environment for malware testing. U.S. Patent No. 10,380,337, 13 Aug 2019.
10. Newberry E, Zhang B, on the power of in-network caching in the Hadoop distributed file system. Proceedings of the 6th ACM Conference on Information-Centric Networking; 2019. pp. 89–99.
11. Ji Y, Fang H, Haichang Y, He J. FastDRC: fast and scalable genome compression based on distributed and parallel processing, 19th International Conference on Algorithms and Architectures for Parallel Processing, Springer; 2019. pp. 313–319.

Publisher's Note