Securing Authentication Within Hadoop

Michael Kanyeba and Lasheng Yu*
School of Information Science and Engineering, Central South University, Changsha, Hunan, P.R. China
*Corresponding author

***Abstract*—In this paper, we discuss authentication issues for Hadoop security in a cloud environment. The main focus is on security issues that are associated with authentication in Hadoop which is framework for storing data on large clusters of commodity hardware — everyday computer hardware that is affordable and easily available — and running applications against that data* We also discuss various possible solutions to those issues in hadoop security and authentication. Hadoop security is developing at a rapid pace which includes computer security, network security, information security, and data privacy. As the amount of enterprise-critical data that resides in the cluster increases, the need for securing access becomes just as critical. Hadoop was not built with security to begin with. Although, cloud computing, big data and its applications, advantages are likely to represent the most promising new frontiers in science. Then it appears essential to get know Hadoop in term of security.**

Keywords-component; kerberos; hadoop; mapreduce; HDFS

I. INTRODUCTION

Hadoop may have started in laboratories with some really smart people using it to analyse data for behavioural purposes; but it is increasingly finding support today in the corporate world. Indeed more and more companies are realizing the benefits it offers for managing and processing very large data volumes. Although there are some changes it needs to undergo to survive in this new environment such as added security; integration with organization existing IT.

When talking about Hadoop security; you have to consider how Hadoop was conceptualized. Doug Cutting and Mike Cafarella initially started developing Hadoop without a regard to security goals. Certainly; it was not even considered as part of the initial design. Hadoop was meant to process large amounts of web data in the public domain; and hence security was not the focus of development. That's why it lacked a security model and only provided basic authentication for HDFS which was not very useful; since it was extremely easy to impersonate another user.

II. HADOOP ENVIRONNMENT

Hadoop is a framework for storing data on large clusters of commodity hardware. It consists of two main components: a distributed processing framework named MapReduce and a distributed file system known as the Hadoop distributed file system; or HDFS.

A. MapReduce

Hadoop Map Reduce is a framework used to write applications that process large amounts of data in parallel on clusters of commodity hardware resources in a reliable; fault-tolerant manner.

A Map Reduce job first divides the data into individual chunks which are processed by Map jobs in parallel. The outputs of the maps sorted by the framework are then input to the reduce tasks.

Generally the input and the output of the job are both stored in a file-system (HDFS).

B. HDFS

HDFS; which stand for Hadoop Distributed File System; is a distributed file system that provides scalable and reliable storage for large volumes of data and that is designed to span large clusters of commodity servers.

HDFS contains a metadata server called the NameNode that stores the hierarchical file and directory name space and the corresponding metadata; and a set of DataNodes that stores the individual blocks of each files. Each block; identified by a block id; is replicated at multiple DataNodes. Client perform file metadata operations such as create file and open file; at the NameNode over an RPC protocol and read/write the data of a file directly to DataNodes using a streaming socket protocol called the data-transfer protocol.

C. Others Parts of Hadoop Environment

Hadoop was not designed and developed as a cohesive system with predefined modules; but was rather developed as a collage of modules that either correspond to various open source projects or a set of (proprietary) extensions developed by various vendors to supplement functionality lacking within the Hadoop ecosystem.

The present Hadoop ecosystem consists of the Hadoop kernel; the Map-Reduce(that provides Analysis of data in clustered environment); the Hadoop Distributed File System (which provides Storage of data) and a number of related components such as Apache Hive; HBase; Oozie; Pig and Zookeeper and these components are explained as below:

- HDFS: A highly faults tolerant distributed file system that is responsible for storing data on the clusters.
- MapReduce: A powerful parallel programming technique for distributed processing of vast amount of data on clusters.
- HBase: A column oriented distributed NoSQL database for random read/write access.
- Pig: A high level data programming language for analyzing data of Hadoop computation.

• Hue A Hadoop administration interface with handy GUI tools for browsing files; issuing Hive and Pig queries; and developing Oozie workflows

• Mahout A library of machine learning statistical algorithms that were implemented in MapReduce and can run natively on Hadoop

• Hive: A data warehousing application that provides a SQL like access and relational model.

• Sqoop: A project for transferring/importing data between relational databases and Hadoop.

• Oozie: An orchestration and workflow management for dependent Hadoop jobs.

The following figure shows as a typical Hadoop ecosystem (or else Hadoop environment) and how various ecosystem components and stakeholders interact with each other. Implementing the security controls in each of these interactions requires elaborate planning and careful execution.

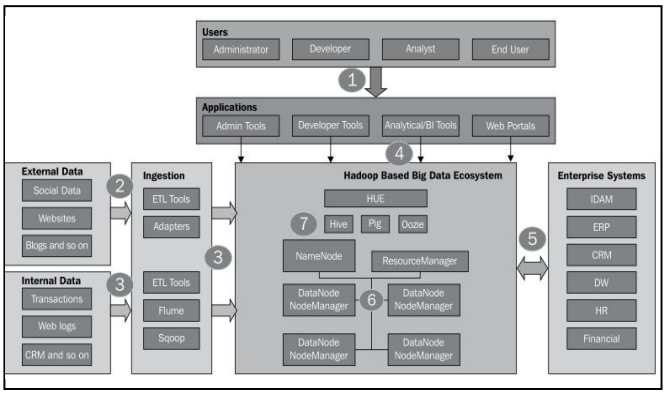

FIGURE I. A TYPICAL HADOOP ECOSYSTEM

III. KERBEROS OVERVIEW

Kerberos is an authentication protocol for "trusted hosts on untrusted networks." It simply means that Kerberos assumes that all the hosts it's communicating with are to be trusted and that there is no spoofing involved or that the secret key it uses is not compromised. To use Kerberos more effectively; consider a few other key facts:

• Kerberos continuously depends on a central server. If the central server is unavailable; no one can log in. It is possible to use multiple "central" servers (to reduce the risk) or additional authentication mechanisms (as fall back).

• Kerberos is heavily time dependent; and thus the clocks of all the governed hosts must be synchronized within configured limits (5 minutes by default). Most of the times; Network Time Protocol daemons help to keep the clocks of the governed hosts synchronized.

• Kerberos offers a single sign-on approach. A client needs to provide a password only once per session and then can transparently access all authorized services.

• Passwords should not be saved on clients or any intermediate application servers. Kerberos stores them centrally without any redundancy.

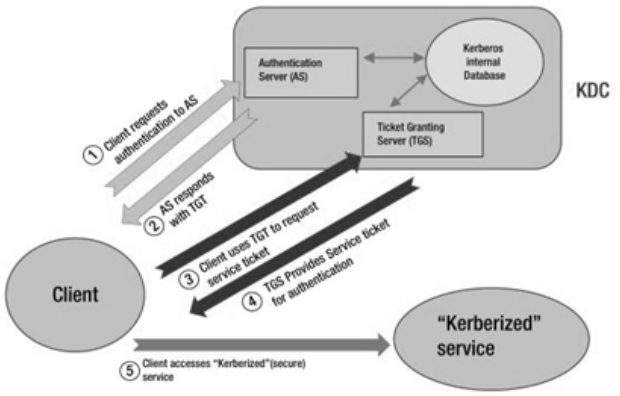

FIGURE II. KERBEROS DISTRIBUTION CENTER WITH ITS MAIN COMPONENTS

A client requests access to a Kerberos-enabled service using Kerberos client libraries. The Kerberos client contacts the Kerberos Distribution Center; or KDC (the central Kerberos server that hosts the credential database) and requests access. If the provided credentials are valid; KDC provides requested access. The KDC uses an internal database for storing credentials; along with two main components: the Authentication Server (AS) and the Ticket Granting Server (TGS).

The Kerberos authentication process contains three main steps:

• The AS grants the user (and host) a Ticket Granting Ticket (TGT) as an authentication token. A TGT is valid for a specific time only (validity is configured by Administrator through the configuration file). In case of services principles (logins used to run services or background processes) requesting TGT; credentials are supplied to the AS through special files called keytabs.

• The client uses credentials to decrypt the TGT and then uses the TGT to get service ticket from the Ticket Granting Server to access a "kerberized" service. A client can use the same TGT for multiple TGS requests (till the TGT expires).

• The user (and host) uses the service ticket to authenticate and access a specific Kerberos-enabled service.Finally; complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar:

IV. SECURITY ISSUES

There are numerous security issues for Hadoop authentication as it was designed without a security approach.

Hadoop services do not authenticate users or other services. As a result; Hadoop is subject to the following security risks.

-A user can access an HDFS or MapReduce cluster as any other user. This makes it impossible to enforce access control

in an uncooperative environment. For example; file permission checking on HDFS can be easily circumvented.

-An attacker can masquerade as Hadoop services. For example; user code running on a MapReduce cluster can register itself as a new TaskTracker.

DataNodes do not enforce any access control on accesses to its data blocks. This makes it possible for an unauthorized client to read a data block as long as she can supply its block ID. It's also possible for anyone to write arbitrary data blocks to DataNodes

By default; Hadoop doesn't authenticate the services; and hence a user can run custom services on any of the machines; and this machine can be registered as DataNode or TaskTracker/NodeManager. Hadoop will replicate the data to all the Hadoop DataNodes; and hence the malicious user machine that registers with NameNode will automatically start receiving the data blocks from the Hadoop cluster. Hadoop has a setting that restricts the machines which can register as DataNodes to NameNode. If the dfs.hosts property in hdfs-site.xml points to a file that contains one host per line; only those hosts will be allowed to connect with NameNode and register. By default; this setting is turned off. This brings up a security hole where any Hadoop client can connect to any DataNode and add malicious data blocks or read any data block using the block ID.

A. Requirements

- Users are only allowed to access HDFS files that they have permission to access.
- Users are only allowed to access or modify their own MapReduce jobs.
- User to service mutual authentication to prevent unauthorized NameNodes; DataNodes; JobTrackers; or TaskTrackers.
- Service to service mutual authentication to prevent unauthorized services from joining a cluster's HDFS or MapReduce service.
- The acquisition and use of Kerberos credentials will be transparent to the user and applications; provided that the operating system acquired a Kerberos Ticket Granting Tickets (TGT) for the user at login.

V. SOLUTION

Hadoop comes with numerous security issues due to its initial development: one without security regards.

A. User and service authentication

User authentication to NameNode and JobTracker services is through Hadoop's remote procedure call using the Simple Authentication and Security Layer (SASL) framework. Kerberos is used as the authentication protocol to authenticate the users within SASL. All Hadoop services support Kerberos authentication. A client submits the MapReduce job to JobTracker. MapReduce jobs are usually long-running jobs and they need to access the Hadoop resources on behalf of the user. This is achieved using the Delegation Token; Job Token; and the Block Access Token.

B. Delegation Token

A Delegation Token authentication is a two-party authentication protocol based on JAVA SASL Digest-MD5. A Delegation Token is used between the user and NameNode to authenticate the user. Once the user authenticates themselves with NameNode using Kerberos; the user is provided with the Delegation Token by NameNode. The user doesn't have to perform Kerberos authentication once he/she obtains the Delegation Token. The user also designates the JobTracker or ResourceManager process as the user that will renew the Delegation Token as part of the Delegation Token request.

The Delegation Token is secured and shared with JobTracker or ResourceManager after authentication; and JobTracker will use the Delegation Token for accessing the HDFS resources on behalf of the user. JobTracker will automatically renew this Delegation Token for long-running jobs.

C. Job Token

A job runs on the TaskNodes and the user access has to be secured in TaskNodes. When the user submits MapReduce job to JobTracker; it will create a secret key that will be shared with TaskTracker that will run the MapReduce job. This secret key is the Job Token. The Job Token will be stored in the local disk of TaskTracker with permission only for the user who submitted the job. TaskTracker starts the child JVM task (mapper or reducer) using the user ID that submitted the job. Thus; the child JVM run will be able to access the Job Token from the local directory and communicate securely with TaskTracker using this Job Token. Thus; the Job Token is used to ensure that an authenticated user submitting the job in Hadoop has access to only the folders and jobs for which he is authorized in the local file system of TaskNodes.

Once the Reduce jobs are started in TaskTracker; this TaskTracker contacts TaskTracker that ran the Map task and fetches the mapper output files. The Job Token is also used by TaskTrackers to securely communicate with each other.

D. Block Access Token

Any Hadoop client requesting for data from HDFS needs to fetch the data blocks directly from DataNode after it fetches the block ID from NameNode. There should be a secured mechanism where the user privileges are securely passed to DataNode. The main purpose of the Block Access Token is to ensure that only authorized users are able to access the data blocks stored in DataNodes. When a client wants to access the data stored in HDFS; it requests NameNode to provide the block IDs for the files. NameNode verifies the requested user's permissions for the file and provides the list of block IDs and DataNode locations. The client then contacts DataNode to fetch the required data block. To ensure that the authentication performed by NameNode is also enforced at DataNode; Hadoop implements the BAT. BAT is the token provided by NameNode to a Hadoop client to pass data access authentication information to DataNode.

The Block Access Token implements a symmetric key encryption where both NameNode and DataNode share a common secret key. DataNode receives this secret key once it registers with NameNode and is regenerated periodically. Each of these secret keys is identified by keyID.

BAT is lightweight and contains expirationDate; keyID; ownerID; blockID; and accessModes. The access Mode defines the permission available to the user for the requested block ID. The BAT generated by NameNode is not renewable and needs to be fetched again once the token expires. BAT has a lifetime of 10 hours. Thus; BAT ensures that the data blocks in DataNode are secured; and only authorized users can access the data blocks.

The following figure shows the various interactions in a secured Hadoop cluster:

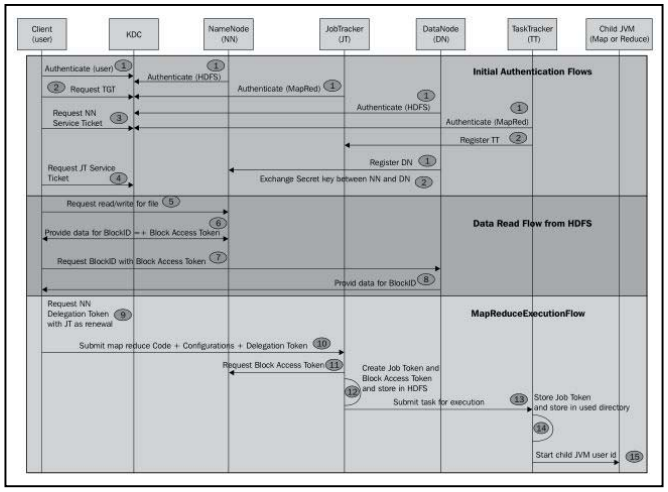

FIGURE III. INTERACTION IN A SECURE HADOOP SYSTEM

VI. CONCLUSION

During the initial days of Big Data implementations using Hadoop; the prime motivation was to get data into the Hadoop cluster and perform analytics on it. As organizations have matured their understanding of Big Data; the data security and privacy policies of such implementations are being questioned. Though Hadoop lacks a robust security and privacy framework; the increasing interest in this area is ensuring that appropriate solutions are developed. While security and privacy issues can be addressed to an extent using existing Hadoop mechanisms such as Kerberos for authentication; more robust tools and techniques are needed and being developed every day.

This paper shows a Hadoop authentication using Kerberos. The security issue is pointed more in order to increase the security in big data. We can improve security in big data by combining approaches.

VII. REFERENCES

[1] Sudheesh Narayanan; Securing Hadoop; 1st ed.; Birmingham - Mumbai: Packt Publishing; 2013.

[2] Bhushan Lakhe; Practical Hadoop Security; ed.; New-York: Apress; 2014.

[3] Dirk deRoos; Paul C. Zikopoulos; Bruce Brown; Rafael Coss; and Roman B. Melnyk; Hadoop For Dummies; Hoboken; New Jersey; John Wiley & Sons; Inc.; 2014

[4] Alex Holmes; Hadoop in Practice; Manning; New York; 2012

[5] Jerry Shenk; Layered Security: Why It Works; SANS Institute; 2013

[6] Kevin Hamlen; Murat Kantarcioglu; Latifur Khan; Bhavani Thuraisingham; Security Issues For Cloud Computing; ser. Lecture Notes in International Journal of Information Security and Privacy; 4(2); 39-51; April-June 2010.

[7] Venkata Narasimha Inukollu; Sailaja Arsi and Srinivasa Rao Ravuri; Security Issues Associated With Big Data In Cloud Computing; in International Journal of Network Security & Its Applications (IJNSA); Vol.6; No.3; May 2014

[8] Saraladevi; Pazhaniraja; Victer Paul; Saleem Basha; Dhavachelvan; Big Data and Hadoop-A Study in Security Perspective; in 2nd International Symposium on Big Data and Cloud Computing (ISBCC'15); Elsevier; 2015

[9] Can Uzunkaya; Tolga Ensari; Yusuf Kavurucu; Hadoop Ecosystem and Its Analysis on Tweets; in World Conference on Technology; Innovation and Entrepreneurship; Elsevier; 2015

[10] Priya P. Sharma and Chandrakant P. Navdeti; Securing Big Data Hadoop: A Review of Security Issues; Threats and Solution; in Priya P. Sharma et al; / (IJCSIT) International Journal of Computer Science and Information Technologies; Vol. 5 (2) ; 2014; 2126-2131

[11] Kevin T Smith; Big Data Security: The Evolution of Hadoop's Security Model

[12] M. Wegmuller; J. P. von der Weid; P. Oberson; and N. Gisin; "High resolution fiber distributed measurements with coherent OFDR;" in Proc. ECOC'00; 2000; paper 11.3.4; p. 109.

[13] R. E. Sorace; V. S. Reinhardt; and S. A. Vaughn; "High-speed digital-to-RF converter;" U.S. Patent 5 668 842; Sep. 16; 1997.

[14] (2007) The IEEE website. [Online]. Available: http://www.ieee.org/

[15] Kevin T Smith; Big Data Security: The Evolution of Hadoop's Security Mode

[16] l on INFOQ. [Online].Available: http://www.infoq.com/articles/Hadoop SecurityModel

[17] https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/SecureMode.html#Authentication.