Securing Hadoop Distributed File System using Intelligence based Sensitivity bits

S. Suganya¹, S. Selvamuthukumaran², S.Swaminathan³, V.Kalaichelvi⁴

¹Research Scholar, Faculty of ICE, Anna University, Chennai, TamilNadu, India
²Professor & Director, Department of Computer Applications, A.V.C. College of Engineering, Mayiladuthurai, TamilNadu, India
³Asst. Professor, Department of ECE, Srinivasa Ramanujan Centre, SASTRA University, Kumbakonam, TamilNadu, India
⁴Senior Asst. Professor, Department of CSE, Srinivasa Ramanujan Centre, SASTRA University, Kumbakonam, TamilNadu, India

ks_suganya@yahoo.co.in¹, smksmk@gmail.com², s.swami@src.sastra.edu³, kalaichelvi2k@src.sastra.edu⁴

Abstract—Hadoop is a widely used analytical tool for analyzing large volume of data which are in terms of Exa & Zetta bytes. Hadoop Distributed File System (HDFS) is a well-known storage system which provides distributed storage using commodity hardware. Since Big Data consists of highly sensitive data like Electronic Health Records, Financial Data etc., harnessing big data access and security plays a vital role. The bottom most layer of security can be provided by encryption. Since the data stored is highly unstructured, wider in variety and the velocity at which data is stored is also unpredictable, traditionally used ciphers like AES, RSA, RC4 etc. will not work efficiently on Big Data. The solution must have to consider at-risk sensitive data, variety of data which needs intelligence to classify the data. When data blocks are stored in data nodes, with each block 4-bits called sensitivity bits has to be added. Two bits indicate the level of sensitivity and two more bits indicate the type of data. These 4-bits are used by the map and reduce functions to effectively encrypt the required data blocks alone and not all, in a time efficient manner by using the available commodity hardware.

Keywords: HDFS, Sensitive Data, Sensitivity bits, Encryption, Intelligence, MapReduce

1. INTRODUCTION

Hadoop is a widely used analytical tool for analyzing large volume of data which are in terms of Exa & Zetta bytes [11]. Hadoop Distributed File System (HDFS) is a well-known storage system which provides distributed storage using commodity hardware [10]. The HDFS storage system is rapidly used in cloud environments also. When precise importance is given for volume, velocity and variety of data, it ignores the security of the data being stored [3]. Since Big Data consists of highly sensitive data like Electronic Health Records, Financial Data etc., harnessing big data access and security plays a vital role. Before entering into the security provided by the HDFS it is better to understand about the architecture and working principle of Hadoop Distributed File System.
2. HDFS WORKING PRINCIPLE

![Diagram of HDFS operations]

The Fig. 1 shows the read and write operations that are done simultaneously on two racks. Reading and Writing by clients to HDFS is done using various tools and APIs; almost the same process is followed by all of these tools. The client always uses Hadoop library for read and write processes which is aware of HDFS and its semantics.

A. WRITING

The Writing of files to HDFS is a complicated process. At first, the client makes a write request to the Namenode to open a file for write operation, along with its name. If the user has write permission, the metadata entry is created for the new file. A response will be sent to the client to indicate that the open request was successful and that it may now start writing data. When the client writes the data to the stream, it is split into packets. For each packet, the client requests the Namenode, a set of Datanodes to which replicas of the next block should be written. Each Datanode in the replication pipeline acknowledges each packet as it is successfully written.

When the client finishes sending data, it flushes any remaining packets out to disk and closes the data stream. It then updates the Namenode that the file is now complete. Obviously, things are not always this simple, and failures may also occur [19].

B. READING

The client sends a read request of a file to the Namenode. The Namenode is not doing any authentication; it directly checks the availability of the file, its owner and permissions of the file. If the file exists and the user has access to it, the Namenode responds to the client with the first block ID and the list of Datanodes on which a copy of the block can be found, sorted by their distance to the client. Distance to the client is measured according to Hadoop’s rack topology.

Using the block ID and Datanode hostnames, the client can now contact the most appropriate Datanode directly and read the block data. For each block the same process is repeated.

If the process or host on which it runs fails, the library will automatically attempt to read another replica of the block from next Datanode. If all replicas are unavailable, the read operation fails and the client receives an exception [19].
3. SECURITY IN HDFS

The primary focus is given to the 3Vs, even we can say that the security and privacy of the stored data are not even touched [18]. We will discuss the security provided with Hadoop Distributed File System in the following key aspects [1]:

A. Authentication
The clients are not authenticated by either the Namenode or Datanodes. The clients are trusted as they are.

B. Authorization
The authorization of a trusted client is done by the Namenode only. But since the Datanodes can also be directly accessed by the clients, the authorization also fails.

C. Data Encryption
Data encryption provides the last layer of defense. When all other security mechanisms fail, encryption helps to ensure the confidentiality of highly sensitive data like medical, financial data etc. There is no inbuilt mechanism to encrypt the sensitive data.

D. Auditing
Log files can be audited by Map functions.

4. EXISTING WORK

The user authentication was done at perimeter level by using Firewalls or Gateway Authentication Protocols like Apache Knox [8]. At the storage level, Kerberos like authentication protocols was used. The process of integration of Kerberos with HDFS is quite difficult [18].

HDFS file permission was implemented by third party tools like Apache Ranger, Apache Sentry [7]. Data encryption is done by using any traditional ciphers like AES [9], DES, RC4 etc. [17]. But it was proven that these algorithms are inefficient for unstructured data, real time, high velocity data [2].

Log files can be audited by Map functions. It is a must to identify flooding and modification attacks. But the logs do not have the information of “who did? ” a particular transaction, that is, the user information. Also the size of log file is smaller and very less number of transactions can be stored in them. It makes the auditing inefficient [5].

5. PROPOSED WORK

The bottom most layer of security can be provided by encryption. Since the data stored is highly unstructured and wider in variety and the velocity at which data is stored is also unpredictable, traditionally used ciphers like AES, RSA, RC4 etc. will not work efficiently on big data. The solution must have to consider at-risk sensitive data and the variety of data which needs intelligence to classify the data.
A. Underlying Architecture

The fig. 5.1 shows the overall architecture of generation of sensitivity bits. The sensitivity of the data can be obtained by getting user input when the client submits the data blocks to the HDFS. But this information is purely from user perspective. But the audit logs contain all the necessary transaction details which must be periodically read by the Map function and analyzed to get insight about which kind of data are prone to attacks at high rates. This needs machine learning algorithms which works in an intelligent way rather than by predefined algorithms. These inputs are also used to update the sensitivity bits.

The next two bits which represent the type of data can be obtained by reading the file extension while submitting the file for write operation.

B. Sensitivity Bits

When data blocks are stored in data nodes, with each block 4-bits called sensitivity bits has to be added. The first two bits indicate the level of sensitivity of the data and the next 2-bits indicate the type of data as discussed above.

The two MSBs are

|   |   |  Level of Sensitivity |
|---|---|-----------------------|
| 0 | 0 | Not sensitive         |
| 0 | 1 | Less Sensitive        |
| 1 | 0 | More Sensitive        |
| 1 | 1 | Highly Sensitive      |

The two LSBs are
These four bits are used to determine the level of security needed for a particular block of data. For example

| 0  | 0  | Video          |
|----|----|----------------|
| 0  | 1  | Audio          |
| 1  | 0  | Image          |
| 1  | 1  | Text           |

These 4-bits called sensitivity bits are stored at the beginning of each block so that the block can be encrypted accordingly.

These 4-bits are used by the map and reduce functions to effectively encrypt the required data blocks alone and not all, in a time efficient manner by using the available commodity hardware.

6. CONCLUSION
This paper represents a method to effectively secure the data stored in Hadoop Distributed File System. This method will identify the data at-risk and provides necessary security based on the level of sensitivity of the stored data. Variations on a cipher may be defined to work according to the sensitivity of data.

7. REFERENCES
[1]. S. Suganya, S. Selvamuthukumaran, “Hadoop Distributed File System Security – A Review” IEEE, 2018
[2]. Youngho Song, Young-Sung Shin, Miyoung Jang, Jae-Woo Chang, "Design and Implementation of HDFS Data Encryption Scheme using ARIA Algorithm on Hadoop", IEEE, 2017
[3]. Madhvaraj M Shetty, Manjaiah D.H ,"Data Security in Hadoop Distributed File System “, IEEE, 2016
[4]. Abdullah Al-Shomrani, Fathy Fathy, Kamal Jambi , "Policy Enforcement for Big Data security “, IEEE, 2017
[5]. Devika Tondon,Monika Khurana ,"Security of Big Data in Hadoop Using AES-MR with Auditing”, International Journal of Advanced Research in Computer Science and Software Engineering , 2017
[6]. Bowen Tian, Yun Tian, Yijie Sun, Trevor Hurt, Brandon Huebert, Waymon Ho, Yuting Zhang, Danqi Chen,"A Secure Data Allocation Solution for Heterogeneous Hadoop Systems: SecHDFS", IEEE, 2016
[7]. Xiao Fu, Yun Gao, Bin Luo, Xiaojiang Du, and Mohsen Guizani,"Security Threats to Hadoop: Data Leakage Attacks and Investigation”, IEEE, 2017
[8]. Boel Nelson, Tomas Olovsson, “Security and Privacy for Big Data: A Systematic Literature Review”, IEEE International Conference on Big Data, 2016

[9]. Shrikant Rangrao Kadam, Vijaykumar Patil, “Review on Big Data Security in Hadoop”, International Research Journal of Engineering and Technology (IRJET), Vol: 04, Issue: 01, Jan- 2017

[10]. Karim Abouelmehdi, Abderrahim Beni-Hssane, Hayat Khaloufi, Mostafa Saadi, “Big Data Emerging Issues: Hadoop Security and Privacy”, IEEE, 2016

[11]. Seref Sagiroglu, Duygu Sinanc, “Big Data: A Review”, IEEE, 2013

[12]. Dylan Maltby, “Big Data Analytics”, ASIST 2011, October 9-13, 2011

[13]. Erhard Rahm, “Big Data Analytics”, IT—Information Technology, 2016

[14]. Big Data Analytics, Vrajaraman, RESONANCE “ August 2016

[15]. Youssef Gah, Mouhine Guennoun, Hussein T. Mouftah, “Big Data Analytics: Security and Privacy Challenges”, IEEE Symposium on Computers and Communication, 2016

[16]. Zhihan Lv, Houbing Song, Pablo Basanta-Val, Anthony Steed, Minho Jo, “Next-Generation Big Data Analytics: State of the Art, Challenges, and Future Research Topics”, IEEE, 2016

[17]. Ahlam Kourid, Salim chikhi, Seng-phil Hong, “A Comparative study of recent advances in Big Data security and Privacy”, AJMAHS, 2017

[18]. Raj R. Parmar, Sudipta Roy, Debnath Bhattacharyya, Samir K. Bandyopadhyay, Tai- Hoon Kim, “Large Scale Encryption in Hadoop Environment: Challenges and Solutions”, IEEE, 2016

[19]. Eric Sammer, “Hadoop Operations”, O Reilly, 2012

[20]. Sulyukova, “Analysis of Low power and reliable XOR-XNOR circuit for high Speed Applications”, Journal of VLSI Circuits And Systems 1 (01), 23-26, 2019.

[21]. Z ZAIN, “High Speed And Lowpower Gdi Based Full Adder”, Journal of VLSI Circuits And Systems, 1 (01), 5-9, 2019.