Kerberos Authentication for Big Data Applications on Cloud Environment

1 Wafaa S. Albaldawi, 2Rafah M. Almuttairi,

1Ministry of Higher Education and Scientific Research, Department of Software, College of Information Technology, University of Babylon, Hillah, Iraq

Ministry of Higher Education and Scientific Research, Department of Computer Science, College of Science, University of Kerbala, Karbala, Iraq

2 Ministry of Higher Education and Scientific Research, Department of Software, College of Information Technology, University of Babylon, Hillah, Iraq

stud.wafaa.shakir@uobabylon.edu.iq
Rafah@uobabylon.edu.iq

Abstract. The large set of User, Sensor, Medical, and Enterprise data can be collected and obtain big data. Therefore, there is a need for a platform for storing, managing, and distributing it across multiple nodes. In the paper, we represent approach which is based on Kerberos in Hadoop distributed file system. It is used to access the right blocks of data by the authorized user only. To provide security in Namenode, Ticket Granting Ticket and Service Ticket have been used for that. Kerberos installation in server and clients are illustrated in details.

1. Introduction

Many big data issues are Management, Processing [1], Security and Storage [2] issues are presented in [3]. They arise in HDFS. When focusing on security issues, there are three approaches Kerberos, Name node and Algorithm to enhance HDFS security. The data in HDFS is sensitive for security issues because there is no roles for access to control the security issues. In addition, in single Hadoop environment, there are embedded data risks of access and thieves. Also, there is a need to protect replicated data from vulnerabilities and failure. As a result, large number of Organizations and government sectors avoid using Hadoop to store the important data. They are using firewall and intrusion detection system to provide security. While some authors avoiding the thieves by using block and file system encryption methods in Hadoop framework. However, other researchers have used encryption algorithms to encrypt the nodes and block. However, the no perfect one has been mentioned to obtain security in Apache Hadoop Framework [3].

Flink, Sentry, and Storm are the Apache Hadoop environment components. They are apt the attack which is caused by weakness in system security procedures; it can be in program
since the Apache Hadoop system is composed totally in Java, which is intensely victimized by cybercriminals and concerned in different security disappointment. In addition, web interface as Hadoop incorporates a frail arrangement since various default settings such as default ports and IP addresses are undefended to XSS scripting assault, and have been as of late exploited. Moreover, in network since the Apache Hadoop framework is a mixture of different databases; diverse sorts of clients in these arrangements are not configured properly, and hence, fine-grained approaches are required at both the service level and the data level [4].

Additionally, in network since the Apache Hadoop system may be a blend of distinctive databases; differing sorts of clients in these configurations are not arranged properly, and consequently, fine-grained approaches are required at both the service level and the data level [4].

Spark's protection is OFF by default. Apache Spark supports multiple types of deployments. Security concerns come in many different forms. It is not necessary that Apache Spark defends against everything [16].

Currently Spark supports remote procedure call (RPC) channel authentication employing a shared secret. Authentication may be exchanged on by setting spark.authenticate. The precise deployment-specific strategy used to deliver and distribute the shared secret [5].

Apache Spark can consequently oversee creating and transmitting the common secret for the Spark on YARN and local deployments. Each application employments a shared secret interesting to it. In YARN's case, this work is based on permitting YARN RPC encryption to secure the dissemination of secrets. For other asset supervisors, it is important to configure spark.authenticate.secret on each of the hubs. All daemons and applications must share this secret [5].

In the paper [6], authors proposed the cloud infrastructure layered security model which is separated into essential, coherent, value-added, and administration security. In addition, they clarified the sticky policy framework architecture for securing the enormous information applications on cloud environment.

Authors in [7] have examined the diverse sorts of assaults such as impersonation, replay, man in middle, denial of service, eavesdropping and repudiation. To perform secure MapReduce handling, we need to discover a appropriate authentication, access control, authorization, availability of data and secrecy of data. Subsequently, Kerberos protocol is prescribed for authentication.

Kerberos [8,9] is an authentication protocol created by Massachusetts Institute of Technology (MIT). Apache Hadoop utilized Kerberos to supply client authentication to get to the Hadoop cluster. It gives a secure communications over a non-secure network by employing a secret key cryptography. It may be a Single Sign-On (SSO) ticket-based framework that depends on Key Distribution Center (KDC). It creates three sorts of tickets for authentication: delegation token may be a secret key between a client and Name Node for authentication, the another token is block access which is utilized to get to a file from HDFS authenticated by Name Node and Data Node to get to a data block on the Data Node, the third token is a job token which is created by JobTracker to authenticate tasks at TaskTrackers.

KDC is a server of Kerberos. It consists of three components: an authentication server which issues a Ticket Granting Ticket (TGT) and perform the first authentication step, a ticket granting server (TGS) which issues consequent service tickets, and a database [9].

Building up client personality with solid authentication is the premise for secure get into Apache Hadoop. Clients have to be dependably recognize themselves and after that have that personality engendered all through the Hadoop cluster to get to cluster assets. Kerberos is an industry used to verify clients and assets inside a cluster[10].

The Kerberos Authentication Process

There are three steps of Kerberos authentication as shown in [Fig.1] [11] as described below:

A client principal requests authentication from the Authentication Server. The Authentication Server returns a Ticket Granting Ticket that is scrambled by using the user principal's Kerberos password,
which is known only to the user principal and the Authentication Server. The client principal decodes Ticket Granting Ticket locally by using its Kerberos password, and from that time forward, until the ticket expires, the user principal can use the Ticket Granting Ticket to induce service tickets from the Ticket Granting Server. Service tickets are what allow a principal to access various services[10].

Client sends a request to TGS for certificate for the specific server. The TGS will answer that client with these certification that is encrypted by the user's key. The certificate consist of a temporary session key (SK) and Service Ticket (ST). It contains the SK and the client identity which are encoded with the key of the server. To access the resources of the server by client, client transmits the ticket to that server and share the SK with it to encrypt the connection between them[11].

Due to hosts and services cannot provide a password anytime to decrypt the TGT, they use a special file, called a keytab. keytab contains the resource principal's authentication certificate. The set of hosts, users, and services over which the Kerberos server has control is called a realm. A realm is the cluster resources. It includes a KDC and a number of Clients [10].

Each principal is for the specific service and sub service in Apache Hadoop. In the realm, a principal name comprises of a primary name and Fully Qualified Domain Name of the node that runs that service. As services do not sign in with a secret word to urge their tickets. Their principal's authentication certificate are saved in a keytab file. It is removed from the Kerberos database and stored in a secured catalog /etc/hadoop/conf as example with the service principal on the service component host [10].

2. Procedure
There are many steps to install the Kerberos server packages in server machine only and to install Kerberos client packages in all client nodes as below:

2.1. Install the Kerberos server packages using the following apt-get command in server machine only:

1- Install krb5-admin-server:
   
   ```
   sudo apt-get install krb5-kdc krb5-config krb5-admin-server -y
   ```
   
   The Kerberos will use the kerberos server domain name as a REALM (ABC.IO). We use the admin server same as the Kerberos server 'krb5.ABC.io'.

2- Generate a new Server password for the Kerberos REALM using the following command:
   
   ```
   sudo krb5_newrealm
   ```

3- The Kerberos Principals and Keytab Files:

   A. Create the kerberos principal files:

   1- Create the admin principal for the KDC Kerberos server called ‘root’:
      
      ```
      sudo kadmin.local
      addprinc root/admin
      ```

   2- Create a principal for hdfs which is utilized for the NameNode, DataNodes, and Secondary NameNode:
      
      ```
      addprinc -randkey hdfs/krb5.abc.io@ABC.IO
      ```

   3- Create the principal of mapred which is used for the NameNode, DataNodes and Secondary NameNode:
      
      ```
      addprinc -randkey mapred/krb5.abc.io@ABC.IO
      ```

   4- Create the yarn principal which is used for the NodeManager and ResourceManager:
      
      ```
      addprinc -randkey yarn/krb5.abc.io@ABC.IO
      ```

   5- Create the HTTP principal:
      
      ```
      addprinc -randkey HTTP/krb5.abc.io@ABC.IO
      ```

   B. Create the Kerberos keytab files:

   1- Create the hdfs keytab file which is used for the NameNode, DataNodes, and Secondary NameNode. It contains the hdfs principal and HTTP principal.
      
      ```
      xst -k hdfs.keytab hdfs/krb5.abc.io@ABC.IO HTTP/krb5.abc.io
      ```

   2- Create the mapred keytab file which is used for the MapReduce Job History Server (in YARN mode). It has HTTP principal and mapred principal.
      
      ```
      xst -k mapred.keytab mapred/krb5.abc.io@ABC.IO HTTP/krb5.abc.io
      ```

   3- Create the yarn keytab file which is used for the NodeManager and the ResourceManager. It will have the yarn principal and HTTP principal.
      
      ```
      xst -k yarn.keytab yarn/krb5.abc.io HTTP/krb5.abc.io
      ```

   4- Using klist command to display the keytab file entries(hdfs)keytab file:
      
      ```
      klist -e -k -t hdfs.keytab
      ```

   5- sudo mv hdfs.keytab mapred.keytab yarn.keytab /etc/hadoop/conf/

   6. sudo chown spuser:hadoop /etc/hadoop/conf/hdfs.keytab
       sudo chown spuser:hadoop /etc/hadoop/conf/mapred.keytab
       sudo chown spuser:hadoop /etc/hadoop/conf/yarn.keytab
       sudo chmod 400 /etc/hadoop/conf/*.keytab
   
   4- Edit vim /etc/krb5kdc/kadm5.acl by adding the following configuration.
   
   root/admin *
   
   The Kerberos service should be restarted:
   
   sudo systemctl restart krb5-admin-server.service
   
   5- Uncomment the 'GSSAPIAuthentication' and enable it by changing the value:
   
   vim /etc/ssh/sshd_config
   
   GSSAPIAuthentication yes
   
   GSSAPICleanupCredentials yes
Restart the ssh service:
 systemct1 restart sshd

2.2. **Install Kerberos client packages by running the following apt command (all client nodes):**

1- `sudo apt-get install -y krb5-user libpam-ccreds libpam-krb5`
Kerberos will use the Kerberos server domain name as a REALM, 'ABC.IO'.
The Admin server same as the Kerberos server 'krb5.abc.io'.
2- The Kerberos Principals and Keytab Files:
(In each client, we have to enter the fully qualified name of it)
A. Create the Kerberos principal files:
In client1.abc.io node:
1- `Kadmin`
2- `addprinc -randkey hdfs/client1.abc.io@ABC.IO`
3- `addprinc -randkey mapred/client1.abc.io@ABC.IO`
4- `addprinc -randkey yarn/client1.abc.io@ABC.IO`
5- `addprinc -randkey HTTP/client1.abc.io@ABC.IO`
B. Create the Kerberos keytab files:
(In each client, we have to enter the fully qualified name of it)
In client1.abc.io node:
1- `xst -norandkey -k hdfs.keytab hdfs/client1.abc.io HTTP/client1 .abc.io`
2- `xst -norandkey -k mapred.keytab mapred/client1.abc.io HTTP/client1 .abc.io`
3- `xst -norandkey -k yarn.keytab yarn/client1.abc.io HTTP/client1 .abc.io`
4- `klist -e -k hdfs.keytab`
5- `sudo mv hdfs.keytab mapred.keytab yarn.keytab /etc/hadoop/conf/
6- `sudo chown spuser:hadoop /etc/hadoop/conf/hdfs.keytab`
sudo chown spuser:hadoop /etc/hadoop/conf/mapred.keytab
sudo chown spuser:hadoop /etc/hadoop/conf/yarn.keytab
sudo chmod 400 /etc/hadoop/conf/*.keytab

3. **Conclusion**
The Kerberos authentication protocol developed by MIT has been widely depended on other organizations with various properties.
Kerberos authenticates the client with the assist of the authentication server and issues a Ticket Granting Ticket to the client. Therefore, it is a trusted third party authentication protocol. In addition, it issues service ticket to the client. With the assistance of the service ticket client can instantly connect to the server.
Over distributed environment, it provides the reliable communication by providing the users personality with the same domain. Ticket system with kerberos protocol authenticate kerberos client.
With Spark being used for production workloads with strict security requirements, fully locking down Spark applications has become critical. In this paper, analyst learns the aspect of securing the spark application. We describe how Spark uses Kerberos for application authentication.

**References**

[1] Changqing, “Big Data Processing in Cloud Computing Environments”, International Symposium on Pervasive Systems, Algorithms and Networks, 2012.
[2] Young-Sae, Song, “Storing Big Data- The rise of the Storage Cloud”, December, 2012.
[3] Saraladevi, B. & Pazhaniraja, N. & Paul, Dr. P. Victer & Basha, Saleem & Dhavachelvan, P.. (2015). Big Data and Hadoop-a Study in Security Perspective. Procedia Computer Science. 50. 596-601. 10.1016/j.procs.2015.04.091.
[4] Bhathal, G.S., Singh, A.: Big data: Hadoop framework vulnerabilities, security issues and attacks. Array 1, 100002 (2019).
[5] Spark Security, Available at: https://spark.apache.org/docs/2.4.3/security.html#standalone-mode-only (Accessed: 18 February 2020).

[6] Sharif, A., Cooney, S., Gong, S., & Vitek, D. (2015). Current security threats and prevention measures relating to cloud services, Hadoop concurrent processing, and big data. 2015 IEEE International Conference on Big Data (Big Data). doi:10.1109/bigdata.2015.7363960

[7] Derbeko, P., Dolev, S., Gudes, E., Sharma, S.: Security and privacy aspects in mapreduce on clouds: a survey. Comput. Sci. Rev. 20, 1–28 (2016).

[8] Tom Yu e a. Kerberos: the network authentication protocol. Available:https://web.mit.edu/kerberos/. [Accessed March 2019].

[9] J. Kohl, C. Neuman, "The Kerberos Network Authentication Service (V5)". RFC 1510, September 1993.

[10] Configuring Authentication with Kerberos Date of Publish: 2018-07-15 http://docs.hortonworks.com.

[11] Mohamed A. Kaafar, "A Kerberos-Based Authentication Architecture for Wireless LANs," Networking 2004, LNCS 3042; Apr. 8, 2004;pp. 1344-1353.