Designing The Security of A Big Data Environment

Student's Name

Institutional Affiliation

Designing The Security of A Big Data Environment

Introduction

The term ‘Big Data’ refers to the massive amounts of digital information collected by companies. Different industries and organizations roughly estimate the growth rate of data to double every two years; Big Data can not be referred to as a specific technology since it is only a collection of attributes and capabilities. Big Data is mainly used in computer science to refer to datasets whose size, as well as the structure, is exceedingly beyond the ability of traditional software tools or traditional database systems to store, process, and analyze within the available time. Hadoop was specially designed for such large scale data operations; Hadoop is best defined as a computing environment built on top of a distributed clustered file system; hence the term Hadoop Distributed File System (HDFS). Hadoop can efficiently process large amounts of data due to its framework that makes it possible to harness a large number of computers; this ability enables Hadoop to process humongous chunks of data that could range up to hundreds of petabytes more quickly and efficiently. The other factor that makes extensive data processing more easily is Hadoop’s built-in fault tolerance.

Development

The chosen Hadoop architecture for my discussion is Apache. Apache Hadoop has always been a decentralized free software computing platform that offers extremely reliable, scalable, dispersed processing of large amounts of sets of data. The software collection is perhaps a platform that stores and processes vast unorganized datasets spread through clusters utilizing simple programming frameworks (Bhathal & Singh, 2019). Hadoop is being used to scale from either one server to many computers, every providing local computing as well as storage. It

is easier to use Hadoop once the scale of an organization's data is in petabytes as well as terabytes.

Securing Hadoop data as well as its environment at large is an essential and complicated task for many administrators, despite the robust built-in authorization features, Hadoop is inherently insecure. During the evolution of Hadoop, security was not a priority since the main concern for the development was data processing. Traditional client-server systems provide robust data security due to the availability of central servers to authenticate users, Hadoop does not have an authentication mechanism either does it have a security gateway. Any user given access to a NameNode can delete the data and even impersonate other users and access data they are not supposed to as well. Data is distributed across all DataNodes; accessing the Nodes and attacking a cluster by an unauthorized user is therefore possible.

There are several security tools used in the architecture to implement the various security elements, and there are three main security concepts involved in Hadoop; that is, authentication, authorization, and auditing. As Hadoop becomes much more mainstream, security issues are consistently addressed by the development of new tools such as Sentry and Knox. The security issues are also addressed through the development of well-established mechanisms like Kerberos. Hadoop's ability to process large amounts of semi-structured and unstructured data has made it into being the go-to data technology. HDFS is flexible and, therefore, essential in the storage of diverse data types regardless of whether the data is semi-structured, unstructured, or structured; Hadoop is consequently a necessity in big data applications that gather collect data from disparate data sources that are in different formats.

Authentication

Authentication in layman's language is simply the process of recognizing a user's identity; it specifically answers the question "Who are you?". Authentication is a mechanism in which incoming requests are associated with a set of identifying credentials, and the input credentials are then compared to existing files in a database on either a local operating system or within an authentication server. Traditional authentication methods that are known to be strong include Lightweight Directory Access Protocol (LDAP), Kerberos, and Active Directory (AD). In Hadoop, they are done outside, mostly at the client site or within the webserver.

The best and most efficient way of providing authentication in Hadoop is through Kerberos security. Kerberos is a network-based tool whose main objective is to provide strong authentication based on the supply of secure encrypted tickets between the clients that request access to servers providing the requested access. Kerberos is distributed by the Massachusetts Institute of Technology, and its model is user-friendly since clients are only required to register with the Kerberos Key Distribution Center (KDC) and, after that, share their password.

In an instance where the client requires to access a resource like a file server, Kerberos sends a request to the KDC with some of the portions encrypted with the password. The KDC then attempts to decrypt the material. If the decryption is successful, the KDC sends a Ticket Generating Ticket (TGT) to the client as a feedback, the TGT forwarded to the client has material encrypted with its unique passcode. After the client receives the TGT, it sends a request for access to the file server to the KDC. The KDC then sends back a ticket that entails encrypted bits with the file server's passcode. The client and the file server then, from after that, use the ticket to authenticate.

The conception is that the file server, which at times might be very busy with many different client requests, is not annihilated with the mechanics of keeping the multiple user passcodes. The file server only shares its passcode with the KDC and finally uses the ticket received by the client from the KDC to authenticate. Kerberos is speculated to be tedious both to set up and maintain; the Hadoop community, therefore, has some actual work to come up with a more straightforward and more effective authentication mechanism.

Authorization of Access to Resources

Authorization is a security mechanism that determines access levels or either user or client concessions related to a system's resources, including services, files, data, and application features as well as computer programs. Authentication is simply the process of either granting or revoking access to a network resource that might allow a user to access different resources based on the user's identity. Most of the web security systems are primarily based on a two-step process, the first step is authentication and the second step is authorization. Authentication ensures about a user's identity, while authorization allows access to the various resources based on the user's identity. Authorization types vary, and passwords might be a requirement in some cases but not in others.

Authorization is a crucial security concern in any environment, and in our case, authorization is essential since it determines whoever can access a Hadoop cluster's data. Access Control Lists (ACLs) are used to configure authorization. Apache Sentry lets one configure a robust and fine-grained authentication for data. Authorization is a bit complex in Hadoop; restricting users by granting them partial access to the data is impossible since Hadoop stores all its data in a file system like a Linux system and not in tables as the case in a relational database.

There is no central authorization system in Hadoop that would enable an administrator to limit the data access by granting partial access to the data files; however, Apache Sentry or Apache Ranger makes this possible.

Services such as Apache Sentry and Apache Ranger would enable an individual to configure fine-grained authorization in Hadoop as long as there is the availability of a database such as Hive, along with an authorization mechanism. Sentry effectively makes it possible for an individual to create rules that specify the possible actions on a table and develop roles as well, which are sets of rules. By the use of Sentry, one can determine portions of file data as Hive tables, and Sentry can then, after that, help with the configuration of fine-grained permissions for specific portions of data. Hadoop's ACL capabilities help an individual specify fine-grained read permissions as well as write permissions to specified users without altering the file permissions for all the other users.

Data Confidentiality

Confidentiality in the computer system's context allows only authorized users to access sensitive and protected data, specific mechanisms ensure confidentiality and safeguard data from intruders that might be harmful. It is a security principle that focuses on the notion that the intended recipients only access information. For instance, sending a letter in the mail to a friend, it would be deemed confidential if the intended recipient (friend) is the only one able to read it. However much it might seem straightforward enough; there are several critical security concepts that are mandatory to ensure that data confidentiality holds. For instance, how do you know the letter you sent is received and read by the intended friend? If the intended recipient reads the letter, how does he know that the message came from you? For the sender and the recipient to

take part in this confidential information passage, an identity to uniquely distinguish themselves from any other person is essential.

Data Integrity

All Hadoop users expect that their data won't be corrupted or lost during storage or processing. However, the chances of data corruption to occur, especially when large data volumes are involved is high; this is because every input or output operation on a network or a disk is associated with a small chance of introducing errors into the data being written or read. Data is usually checked if corrupted by computing a *checksum* when the data is first added in the system, and again each time the data is transmitted across an unreliable channel that could corrupt it. Data is deemed corrupt if the original checksum doesn't exactly match with a newly generated one; this is only an error detection technique and doesn't, therefore, offer any way of fixing the corrupt data. The checksum at times might be corrupt and not the data; this rarely happens though since the checksum is much smaller in comparison to the data.

HDFS checksums all a user's data that is written to it and varies checksums as well by default while reading data. It is a DataNode's responsibility to verify data received before storing it and its checksum, and this applies to data obtained from clients as well as other datanodes during replication. A client sends his written data to a pipeline of datanodes, and the checksum is then verified by the last datanode in the pipeline. Should the datanode detect an error, the relevant client receives a subclass of 10Exception. Clients verify checksums as well in the process of reading data from datanodes, and the checksums are compared with the ones stored at the datanodes. Each datanode knows the last time each of its blocks was verified by keeping a persistent log of checksum verifications.

Registration and Audit

HDFS is basically at Hadoop's core, and it provides a distributed file system that makes Hadoop so successful. The HDFS audit logs are the hdfs-audit.log, which is intended for user activity and SecurityAuth-hdfs.audit, which is designed for service activity. The two logs are implemented with Apache Log4j, which is a common and generally known mechanism for logging in java. The properties of the Log4j can be configured in the log4j.properties file.
An example log for user fianciso after a listing of files and an attempted copy to a directory. /user/doc which was denied is illustrated below.

2017-09-06 12:15:10,123 INFO FSNamesystem.audit: allowed=true ugi=fianciso@POST.COM (auth:KERBEROS) ip=/192.168.2.22 cmd=getfileinfo src=/user/fianciso dst=null perm=null

2017-09-06 12:15:10,125 INFO FSNamesystem.audit: allowed=true ugi=fianciso@POSTCOM (auth:KERBEROS) ip=/192.168.2.22 cmd=listStatus src=/user/fianciso dst=null perm=null

2017-09-06 12:15:46,167 INFO FSNamesystem.audit: allowed=false ugi=fianciso@POST.COM (auth:KERBEROS) ip=/192.168.2.10 cmd=rename src=/user/fianciso/delorean dst=/user/doc perm=null

Availability and Performance

Hadoop is an ecosystem of libraries, and each library is dedicated to performing a specific task. Data is only written once to the server, but the written data can be read and reused more than once. Hadoop is considered the perfect solution when dealing with a large variety of data since it exhibits a high speed. Task Tracker slave nodes are managed and jobs executed by the Job Tracker, which acts as the master node. Whenever a user requires data, a request is sent

to the master node, which is the NameNode; in this case, the data is then passed to all the other DataNodes, which servers the data requested.

Hadoop uses MapReduce or YARN for scheduling and processing. MapReduce is responsible for executing a sequence of jobs, and each situation is specifically a java application running on the data. To give a data hunter durable power as well as flexibility, other querying tools like Hive Hadoop and Pig Hadoop can be used instead of MapReduce.

Conclusion

It is evident and a general knowledge that Big Data has taken the world by storm. Researches argue that Big Data will dominate the next decade, and all companies will be using the data available to them so as to learn about their company's ecosystem; the findings would be essential for improving fallbacks. Globally, all the major universities, as well as companies, have dramatically focused on investing in tools that might help in understanding and developing useful insights from data that can be accessed. One such tool that has proven to be helpful in analyzing as well as processing Bid Data is Hadoop.

References

Culwin, F., & Lancaster, T. (2001). Plagiarism issues for higher education. VINE, 31(2), 36-41, https://doi.org/10.1108/03055720010804005Bhathal, G. S., & Singh, A. (2019). Big data: Hadoop framework vulnerabilities, security issues,and attacks. Array, 1, 100002.

Cohen, J., & Acharya, S. (2013, June). Towards a more secure apache hadoop hdfsn infrastructure. International Conference on Network and System Security (pp. 735-741). Springer, Berlin, Heidelberg.Spivey, B., & Echeverria, J. (2015). Hadoop Security: Protecting your big data platform. "O'Reilly Media, Inc.."