Challenges in Big Data Security and mitigating Threats by leveraging Apache Hadoop

Pravat Sutar

Abstract— Data breaches are on the rise in recent days. The HIPAA Breach Reporting Tool website of the Department of Health and Human Services shows a total of 105 breaches affecting more than 2.5 million individuals and added to the tally so far in 2020. Ryuk Ransomware continues to target medical facilities despite the ongoing COVID-19 pandemic. While the healthcare industry is focused on preventing the spread of coronavirus and working on the vaccine, hacking groups are targeting the industry in full force. On March 2020, its reported that more than 5 billion records were exposed due to an unprotected Elasticsearch database managed by a UK based security firm. Marriott's Data Breach could end up costing the hospitality business of $3.5 billion. A higher education institution in Victoria, Australia, has disclosed a data breach impacting the personal data of around 90,000 staff, students, and suppliers. Over 3,800 data breaches were reported through June 2019 and exposed over 4.1 billion records which is 50% or greater increase over the last four years. An attack on Facebook's computer network exposed the personal data of over 50 million users on Sept 2018. Lenovo confirmed 36TB Data Leak Security Vulnerability on July 2019. In terms of data breach sources, malicious outsiders and accidental loss are at the top of the list in 2020. A report from Market Research forecasts that the Hadoop market will grow at a compound annual growth rate (CAGR) of 58 percent through 2022.

With major data breaches hitting well-known entities, data security becomes more and more critical and challenging, and security in big data cannot be overlooked. Data protection regulations and standards including the European Union's General Data Protection Regulation (GDPR), New York's Cybersecurity Requirements for Financial Services Companies and Australia's Notifiable Data Breach (NDB) schemes are introduced to help organizations better protect customers' privacy and security by design. The volume of data generated in enterprise environments is growing exponentially. More and more organizations are exploring the Big Data infrastructure that support new opportunities, cost savings and transformation. Organizations are collecting extremely large datasets from diverse data sources for advanced data analytics. Hence, data protection and data privacy in Big Data Hadoop need to be considered from day one while adhering to expanding compliance requirements. Goals for data management systems, such as confidentiality, integrity, and availability, require that the system be secured across several dimensions.

The goal of this paper is to provide a comprehensive and holistic view on security challenges in Big Data Hadoop and techniques to mitigate the risks associated with the security threats. The paper also addresses the challenges in big data security and provides the solutions by leveraging the Apache Hadoop. Different data security solutions such as access to the Hadoop cluster through perimeter security (i.e. authentication and network isolation), protecting data in the cluster from any unauthorized access (i.e. encryption, row level filter and data masking), defining what users and applications can do with data (i.e. authorization, permission), reporting on where data came from and how it's been used (i.e. data lineage and auditing and data governance) and using SSL or TLS network security to authenticate and ensure privacy of communications between nodes, name servers, and applications are covered in this paper. The paper also highlights the recommendations and best practices of each security components in Big Data Hadoop defined.

Index Terms— Big Data Hadoop, Data Security, Authorization, Authentication, Data Encryption, Data Lineage, Data Governance, Data Audit and Reporting, Data Breach, Data Masking, Row Level Data Filter, Access Control List, Key Management Service (KMS), Secure Socket Layer (SSL) and Transport Layer Security (TLS), Active Directory, Lightweight Directory Access Protocol

1 Introduction

The CIA (Confidentiality, Integrity and Availability) model is a high-level principle that can be applied to a wide range of information systems and computing platforms. Confidentiality is a security principle focusing on the notion that information is only seen by the intended recipients. For example, if Alice sends a letter in the mail to Bob, it would only be deemed confidential if Bob were the only person able to read it. This might look like a straightforward approach, there are several important security concepts involved to ensure the confidentiality.

To ensure that the letter Alice is sending is being read by the right Bob and the correct Bob should know that the letter is received from right Alice, both should have an identity that uniquely distinguishes themselves from any other person. In addition, both Alice and Bob need to prove their identities via a process known as authentication. Identity and authentication are key components of Hadoop security to ensure confidentiality.

Encryption, an important concept of confidentiality is a mechanism to apply a mathematical algorithm to a piece of information where the output is something that unintended recipients are unable to read. Only the intended recipients can decrypt the encrypted message back to the original unencrypted message. Encryption of data can be applied both at-rest and in-transit. At-rest data encryption means that data resides in an encrypted format when not being accessed. In-flight encryption or in-transit encryption applies to data sent from one place to another over a network. Both modes of encryption can be used together or independently.

Integrity of data is a critical component of information securi-

Author: Pravat Sutar, E-mail: pravat.sutar01@gmail.com, LinkedIn: https://www.linkedin.com/in/pravat-sutar/

International Journal of Scientific & Engineering Research, Volume 11, Issue 5, May-2020
ISSN 2229-5518

ty, especially the industries which deal with highly sensitive data. Even if confidentiality is guaranteed, data that doesn't have integrity guarantees is at risk of substantial damage. For example, Alice sends a letter to Bob. if Charles intercepts the letter in transit and makes changes to it without the knowledge of Alice and Bob, the integrity is compromised. Some measures such as file permissions and user access control must be in place in big data cluster. To detect any changes in data that might occur as a result of an electromagnetic pulse or server crash, some mechanism must be in place. Some data might include checksums, and some includes cryptographic checksums for verification of integrity. Backups or redundancies must be available to restore the affected data to its correct state.

Confidentiality and integrity are closely aligned to well-known security concepts, but availability is largely covered by operational preparedness. For example, if Alice sends letter to Bob but the letter is not sent to Bob because the post office is closed, then the letter is unavailable to Bob. The availability of data or services can be impacted by regular outages such as scheduled downtime for upgrades or applying security patches. It can also be impacted by security events such as distributed denial-of-service (DDoS) attacks. High Availability (HA) and Disaster Recovery (DR) are essential for keeping the Hadoop cluster available all the time. Data loss or interruptions in connections occurs because of unpredictable events such as natural disasters and fire. To prevent data loss from such occurrences, a backup copy may be stored in a geographically isolated location. Firewalls and proxy servers can guard against downtime and unreachable data blocked by malicious denial-of-service (DoS) attacks and network intrusions as well.

Hadoop powered data lakes can provide a robust foundation for a new generation of analytics and insight. Securing the data before launching or expanding a Hadoop initiative is highly important. By ensuring that data protection and governance are built into their Big Data environment, enterprises can exploit the full value of advanced analytics without exposing their businesses to new risks. Few pillars of Big Data security elaborated in this paper are given below.

Sl No.	Areas of Big Data Security
1	Authentication and identity propagation using Kerberos
2	Kerberos SPNEGO authentication
3	Perimeter Security using Apache Knox
4	Authorization using Apache Ranger
5	Authorization with Apache Sentry
6	A comparative study of Apache Sentry and Apache Ranger
7	Data Protection (both data-at-rest encryption and in-transit data encryption)
8	Access Control List (ACL)
9	Hadoop Group Mapping for LDAP/AD with SSSD
10	SSL for Hadoop web components
11	Data Governance
12	Auditing and Reporting

Security features described throughout this paper apply to the versions of the associated project listed in below table.

Apache Hadoop 3.1.1	Apache Ambari 2.7.5
Apache Knox 1.0.0	Apache HBase 2.1.6
Apache Ranger 1.2.0	Apache Hive 3.1.0
Apache Atlas 2.0.0	Apache Kafka 2.0.0
Kerberos 5	Apache Spark 2.3.2
Apache Sentry 2.1.0	Apache ZooKeeper 3.4.6
Transport Layer Security (TLS 1.3)	Apache Zeppelin 0.8.0

2 Challenges in Data Security

Data security is the process of protecting the most critical business assets / data against unauthorized or unwanted usage including deploying the right data security products and combining people and processes with the technology you choose to protect data throughout its lifecycle. Much like other forms of cyber-security, the big data variant is concerned with attacks that originate either from the online or offline spheres.

These threats include the theft of information stored online, ransomware, or DDoS attacks. The issue can be even worse when companies store information that is sensitive or confidential, such as customer information, credit card numbers, or even simply contact details. Additionally, attacks on an organization's big data storage could cause serious financial repercussions such as losses, litigation costs, and fines or sanctions.

Implementing Big data security ensures to keep out on unauthorized users and intrusions with firewalls, provides strong user authentication, end-user training, and intrusion protection systems and intrusion detection systems. In addition, encrypting the enterprise data in-transit and at-rest. Along with this network security strategy, big data environments provide few additional layers of security because security tools must operate during three data stages that are not all present in the network. These stages are at data ingress / data that coming in, data storage and data output / outbound which goes out to applications and reports.

Incoming Data from different Data Sources.

The first challenge is data ingestion – data is ingested from variety of source systems such as user-generated data alone can include CRM or ERM data, transactional and database data, and vast amounts of unstructured data such as email messages or social media posts. In addition, the machine generated data including logs and sensors which could be intercepted or corrupted in transit. The data in-transit from sources to the platform must be secured.

Data in Storage

When the data is at the storage layer, it can be stolen or held hostage while resting on cloud or on-premise clusters. Protecting data at storage layer takes mature security toolsets including encryption at rest, strong user authentication, and intrusion protection and planning. The security toolsets must be configured across a distributed cluster platform with many servers

International Journal of Scientific & Engineering Research, Volume 11, Issue 5, May-2020
ISSN 2229-5518

and nodes, and it must protect log files and analytics tools as they operate inside the platform.

Data output or outbound for Analytics

The data used for analytics from output or outbound layer could provide an access point for hackers or other malicious parties. As the entire reason for the complexity and expense of the big data platform / data lake is being able to run meaningful analytics across massive data volumes and different types of data, it could provide an access point for hackers or other malicious parties. These analytics output results to applications and data visualization. This valuable intelligence makes for a rich target for intrusion and it is critical to encrypt the output. And for security compliance at this stage, make certain that results going out to end-users do not contain regulated data. These three challenges play a crucial and critical role in creating a flexible end-to-end big data security architecture for any organization.

3 Big Data Hadoop Security Components Overview

Hadoop-powered data Lake can provide a robust foundation for a new generation of analytics and insight, but it's important to consider security before launching or expanding a Hadoop initiative. By making sure that data protection and governance are built into your Big Data environment, you can leverage the full value of advanced analytics without exposing your business to new risks.

A holistic approach on data security comprises of below components –

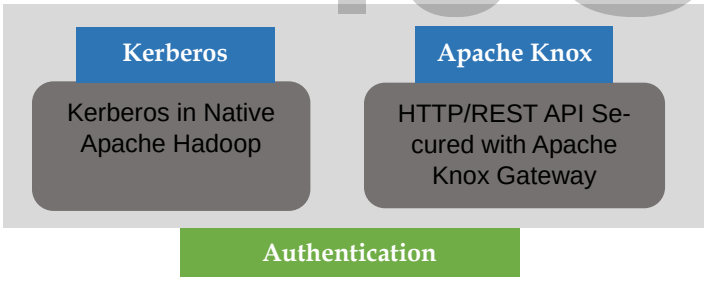


Figure 1: Authentication Components

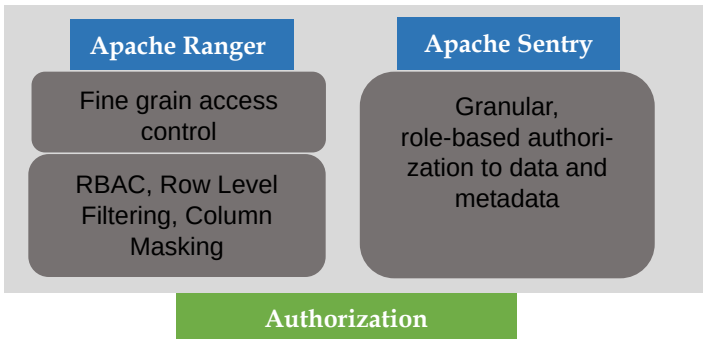


Figure 2: Authorization Components

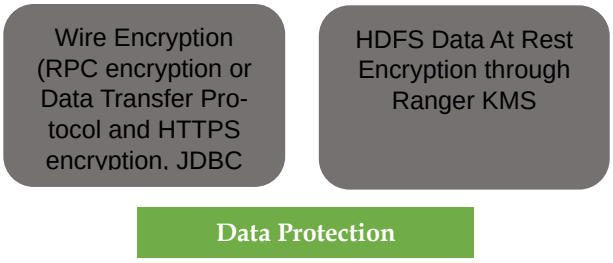


Figure 3: Data Protection Components

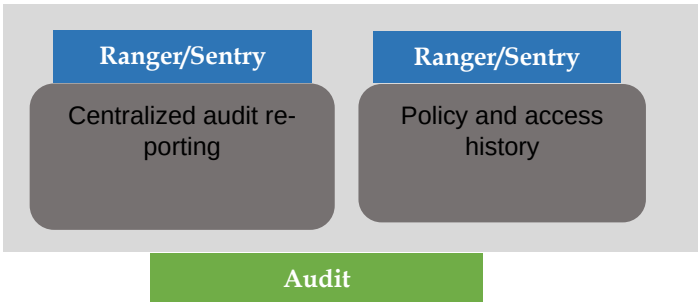


Figure 4: Data Audit & Reporting Components

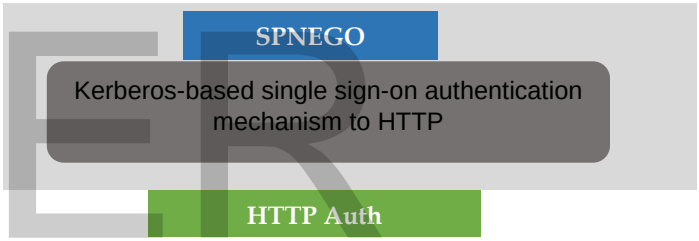


Figure 5: Kerberos SPNEGO Authentication

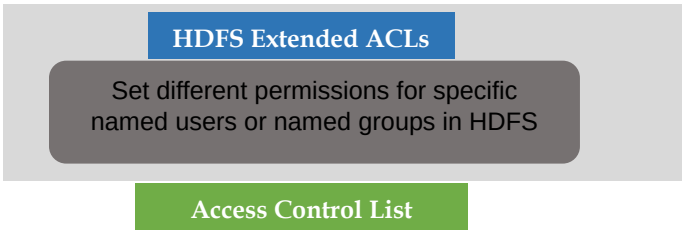


Figure 6: HDFS Extended ACLs

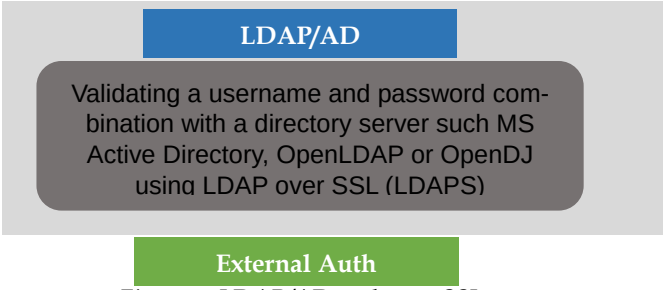
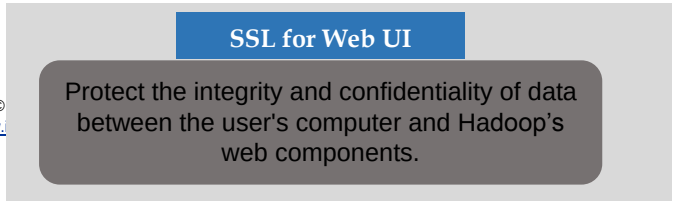


Figure 7: LDAP/AD auth over SSL



International Journal of Scientific & Engineering Research, Volume 11, Issue 5, May-2020
ISSN 2229-5518

Figure 8: HTTPS for Hadoop Web Components

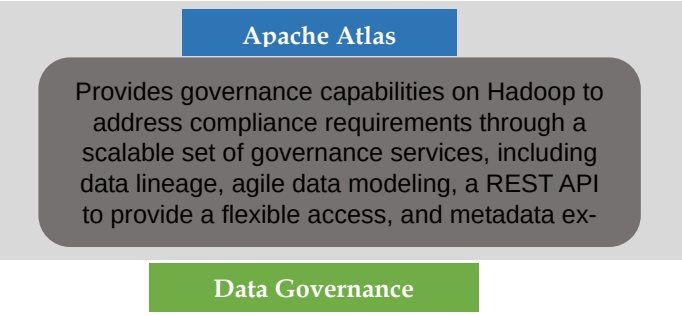


Figure 9: Data Governance & Management

4 Authentication and Identity Propagation using Kerberos

Authentication is broadly classified into two categories. They are service authentication and user authentication.

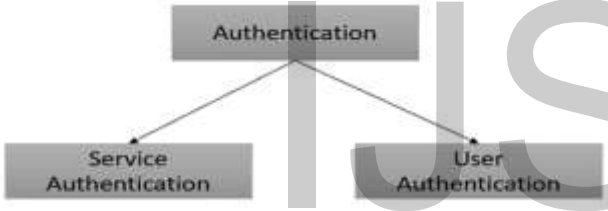


Figure 10: Authentication categorization

4.1 Service Authentication

Service authentication refers to the identity verification process among different service components. There are many components in Big Data ecosystems such as HDFS, YARN, Map Reduce, Zookeeper, HBase, Oozie, Spark, Kafka etc. A typical data lake consists of master node, edge node and data node. For example – Name node, Resource manager, Zookeeper, HBase master and other master components are setup in master node. Similarly, node manager, data node, HBase region server and Ambari agent etc. are setup in data node. Inter communication among the services are done as part of processing a request. If a new node is provisioned with required service component installed and the services are allowed to join the cluster without authenticating its identity, the data may be illegally obtained. If a particular service enables service authentication, only nodes with valid identity information can join the cluster.

Example- When Hive is integrated with its metadata store, the metadata store credentials are configured with Hive. When Apache Nifi is integrated with Ranger, proper credentials are being configured for identity.

The intercommunication among the service components can be secured with through wire encryption. Wire encryption protects data as it moves in and out of Hadoop cluster over RPC, HTTP, Data Transfer Protocol (DTP), and JDBC. A client communicates directly with Hadoop cluster and the data can be protected through RPC encryption and Data Transfer Protocol. A client uses RPC to connect to the NameNode (NN) to initiate file read and write operations. RPC connections in Hadoop use Java's Simple Authentication & Security Layer (SASL) with encryption well supported. When the client request to read and write the data from the datanode, the NameNode gives the client the address of the DataNode (DN). The actual data transfer between the client and a DN uses Data Transfer Protocol.

To access the Hadoop web components, users typically uses a browser. For example, Ambari Web UI, Ranger Admin web UI, Resource Manager Web UI etc. The user also uses command line tools (CLI) and applications use REST APIs or Thrift. Encryption over the HTTP protocol is implemented with the support for SSL across a Hadoop cluster and for the individual components.

HiveServer2 implements encryption with Java SASL protocol's quality of protection (QOP) setting. With this the data moving between a HiveServer2 over JDBC and a JDBC client can be encrypted. Additionally, HTTPS encryption should be used in the shuffle phase especially when data moves between the Mappers and the Reducers over the HTTP protocol.

4.2 User Authentication

User authentication is a process that allows a device to verify the identity of a user/client who connects to a network resource. Without the user authentication, the service simply trusts the identity information provided by clients.

On most of the scenarios, a password is used to prove the user's identity. For example - on a distributed network system, the password is transmitted over the network from the edge node. Example – the client queries the Hive table. As this password is the one secret piece of information that identifies a user, anyone knowing a user's password can effectively be that user on Hive, querying the data, updating critical data. As per the Electronic Communications Privacy Act of 1986 (ECPA), it's a federal crime.

Hence, it is necessary to prevent anyone from intercepting or eavesdropping on the transmitted password. In addition, it is necessary to provide a means of authenticating users: any time a user requests a service, they must prove their identity. This user authentication is greatly achieved by Kerberos.

Strongly authenticating and establishing a user's identity is the basis for secure access in Hadoop. Hadoop uses Kerberos as the basis for strong authentication and identity propagation for both user and services. Kerberos is a computer-network authentication protocol that works on the basis of tickets to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner. It's a client–server based model and provides mutual authentication - both the user and the server verify each other's identity. Kerberos protocol messages are protected against eavesdropping and replay attacks.

Kerberos builds on symmetric key cryptography and requires a trusted third party and use public-key cryptography during certain phases of authentication. The Kerberos server

International Journal of Scientific & Engineering Research, Volume 11, Issue 5, May-2020
ISSN 2229-5518

itself is known as the KDC (Key Distribution Center). At a high level, it has three parts:

a. A database of the users and services (also known as principals) that it knows about and their respective Kerberos passwords
b. An AS (Authentication Server) which authenticates the Kerberos client against the user database, and grants a Ticket Granting Ticket (TGT) for the client
c. A TGS (Ticket Granting Server) that issues subsequent service tickets based on the initial TGT

It validates and the client is allowed to access the requested Kerberos service and issues a service ticket for that service. The TGS acts as the trusted third party in the Kerberos protocol. Based on the user principal requests, the AS returns a TGT which is encrypted using the user principal's Kerberos password and its known only to the user principal and the AS. The user principal decrypts the TGT locally using its Kerberos password and from that point forward, until the ticket expires, the user principal can use the TGT to get service tickets from the TGS. Service tickets are what allow a principal to access various services. The cluster resources like the host and services cannot provide a password each time to decrypt the TGT so they use a special file called a keytab file which contains the resource principal's authentication credentials. Kerberos server has control on the set of hosts, users, and services is known as realm.

Term	Description
Key Distribution Center (KDC)	The trusted source for authentication in a Kerberos-enabled environment.
Kerberos KDC Server	The machine or server that serves as the Key Distribution Center (KDC).
Kerberos Client	Any machine in the cluster that authenticates against the KDC.
Principal	The unique name of a user or service that authenticates against the KDC.
Keytab	A file that includes one or more principals and their keys.
Realm	The Kerberos network that includes a KDC and several Clients.
KDC Admin Account	An administrative account used by Ambari to create principals and generate keytabs in the KDC.

Kerberos Architecture

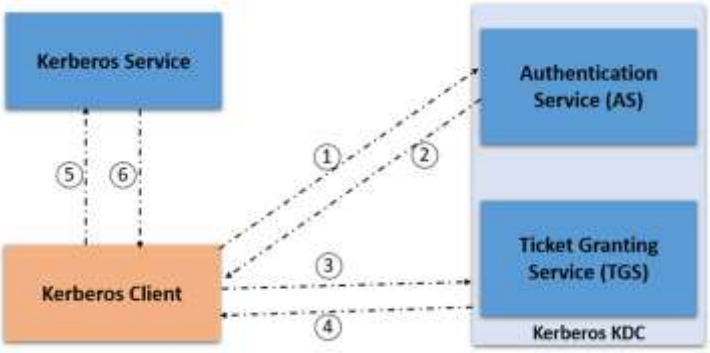


Figure 11: Kerberos Architecture

Step 1: The Kerberos client sends its user ID in a cleartext message which does not include the client's password, nor its secret key based on the password to the authentication server (AS).

Step 2: The authentication server (AS) validates if the client is in the user database and generates the secret key for the client by hashing the client's password if it is found. The authentication server then sends a client or TGS session key and a TGT to the Kerberos client. The session key is encrypted with the secret key of the client.

Step 3: The Kerberos client then decrypts the client or TGS session key and sends a request message containing the TGT and the ID of the Kerberos service to be accessed and an authenticator message containing the client ID and the timestamp and encrypted with the client or TGS session key to the ticket granting service.

Step 4: The TGS decrypts the ticket in the request message to retrieve client or TGS session key and decrypts the authenticator message. The Kerberos client is then verified by the if it is authorized to access the Kerberos service requested. If it is authorized, it sends a service ticket and a client/server session key encrypted along with the client or TGS session key back to the Kerberos client.

Step 5: The Kerberos client accesses the Kerberos services by sending the service ticket and a new authenticator message encrypted with the client/server session key to the Kerberos service.

Step 6: The Kerberos service decrypts the service ticket to retrieve the client/server session key, then decrypts the authenticator message to retrieve the client's timestamp. The Kerberos service sends a service confirmation message including the timestamp and encrypted with the client/server session key back to the Kerberos client.

The mutual authentication is now complete after the Kerberos client decrypts the service confirmation message and verifies the timestamp is correct. The Kerberos client can now start issuing service requests, and the Kerberos service can provide the requested services for the client.

Kerberos authentication relies on a secure user database storing user IDs and passwords. Using secret keys for encryption requires that the password on the Kerberos client or Kerberos service must match the one stored in the database on the KDC. If the passwords do not match, the secret keys hashed from the passwords do not match either and decrypting messages fails. As a result, the Kerberos client cannot access the services configured with Kerberos. A typical flow of accessing hive server 2 in Kerborized Hadoop cluster.

International Journal of Scientific & Engineering Research, Volume 11, Issue 5, May-2020
ISSN 2229-5518

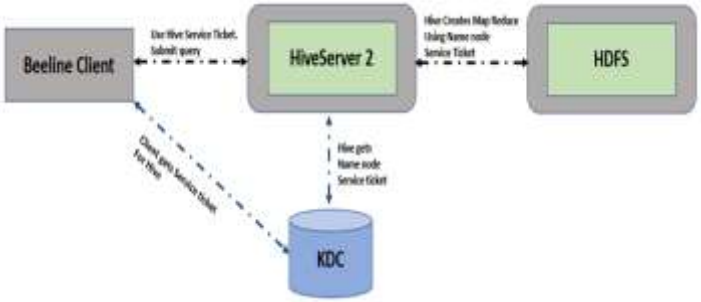


Figure 12: Accessing Hive in Kerborized Hadoop cluster

Recommendations

Encryption settings in MIT Kerberos are often set to a variety of encryption types, including weak choices such as DES by default. Its recommended to remove weak encryption types to ensure the best possible security. Weak encryption types are easily exploitable. When using AES-256, Java Cryptographic Extensions need to be installed on all nodes in the cluster to allow for unlimited strength encryption types. It is important to note that some countries prohibit the usage of these encryption types. Always follow the laws governing encryption strength for your country.

In environments where users from Active Directory (AD) need to access Hadoop Services, setting up one-way trust between Hadoop Kerberos realm and the AD (Active Directory) domain is recommended.

As Kerberos is a time-sensitive protocol, all hosts in the realm must be time-synchronized, for example, by using the Network Time Protocol (NTP). If the local system time of a client differs from that of the KDC by as little as 5 minutes (the default), the client will not be able to authenticate.

5 Kerberos Spnego Authentication

Simple and Protected GSSAPI Negotiation Mechanism (SPNEGO) is a GSSAPI "pseudo mechanism" used by client-server software to negotiate the choice of security technology. SPNEGO is used when a client application wants to authenticate to a remote server, but neither end is sure what authentication protocols the other supports.

SPNEGO's most visible use is in Microsoft's "HTTP Negotiate" authentication extension. It was first implemented in Internet Explorer 5.01 and IIS 5.0 and provided single sign-on capability later marketed as Integrated Windows Authentication. The negotiable sub-mechanisms included NTLM and Kerberos, both used in Active Directory. The HTTP Negotiate extension was later implemented with similar support in Mozilla and Google Chrome.

Kerberos authentication over HTTP refers to the use of the HTTP negotiate protocol to perform Kerberos authentication at the transport layer between a client and a service in Hadoop cluster. The tokens required for the authentication are transmitted in HTTP headers. The SPNEGO specification suggests using SSL/TLS to provide confidentiality with the authentication mechanism.

By default, access to the HTTP-based services and UIs for the Hadoop cluster are not configured to require authentication. Kerberos authentication can be configured for the Web UIs for HDFS, YARN, MapReduce2, HBase, Oozie, Falcon, Zeppelin Notebook UI and Storm.

Recommendations

Issues encountered in SPNEGO can be hard to debug. Enable the flag "-Dsun.security.spnego.debug=true" in krb5.conf which can provide additional debug logging for a Kerberos secured web endpoint.

6 Perimeter Security using Apache Knox

Perimeter security refers to natural barriers or built fortifications to either keep intruders out or to keep captives contained within the area the boundary surrounds. Perimeter security helps secure Apache Hadoop cluster resources to users accessing from outside the cluster. It enables a single access point for all REST and HTTP interactions with Apache Hadoop clusters and simplifies client interaction with the cluster.

Perimeter security has following benefits:

- Hide service-specific URLs/Ports by acting as a Proxy.
- Simplify authentication of various Hadoop services and UIs
- Enable SSL termination at the perimeter
- Ease management of published endpoints across multiple clusters
- Provides detailed access logs

Typical perimeter security includes technologies like application proxies, firewalls, intrusion detection systems (IDS) and virtual private network (VPN) servers. A combination of these technologies will provide a hardened exterior to your deployment. Apache Knox as a type of application proxy within the perimeter layer which receives requests intended for another server and acts on the client's behalf to obtain the requested resource.

Application proxy servers are often used when the client and the server are incompatible for direct connection. If the client is unable to meet the security authentication requirements of the server but through the application proxy, it is permitted to access some of the services. An application proxy authenticates users and can assert the authenticated identity to the intended server, breaks the TCP/IP connection between a client and server, hides the internal cluster IP addresses; only the IP address of the proxy server is visible to clients, provides detailed access logs and caches information as well.

The Apache Knox Gateway is a system to extend the reach of Apache Hadoop services to users outside of a Hadoop cluster without reducing Hadoop security. Knox also simplifies Hadoop security for users who access the cluster data and execute jobs and it is designed as a reverse proxy. Knox integrates with Identity Management and SSO (Single Sign-On) systems used in enterprises and allows identity from these systems be used for access to Hadoop clusters. Considering Accessibility, Security and Integration, Apache Knox Gateways provides out of the box capabilities. They are given below –

- It extends Hadoop's REST/HTTP services by encapsulating Kerberos to within the Cluster. Hence, access is simplified.

International Journal of Scientific & Engineering Research, Volume 11, Issue 5, May-2020
ISSN 2229-5518

- It imposes REST API security centrally, routing requests to multiple Hadoop clusters. Hence, controlling the access is centralized.
- It exposes Hadoop's REST/HTTP services without revealing network details, providing SSL out of the box. Hence, security is enhanced.
- It supports Active Directory, LDAP, SSO, SAML and other authentication systems. Hence, enterprise integration is greatly achieved.

Apache Knox 1.4.0! released on 26th April 2020 delivers three groups of users facing services

Proxying Services	Provide access to Apache Hadoop via proxying of HTTP resources.
Authentication Services	Authentication for REST API access as well as WebSSO flow for UIs. LDAP/AD, Header based PreAuth, Kerberos, SAML, OAuth are all available options.
Client Services	Client development can be done with scripting through DSL or using the Knox Shell classes directly as SDK. The Knox-Shell interactive scripting environment combines the interactive shell of groovy shell with the Knox Shell SDK classes for interacting with data from your deployed Hadoop cluster.

Apache Knox deployment Architecture

Hadoop is accessed by the user externally through Knox, REST API and through the Hadoop CLI tools. The following diagram shows a typical Hadoop deployment using Apache Knox.

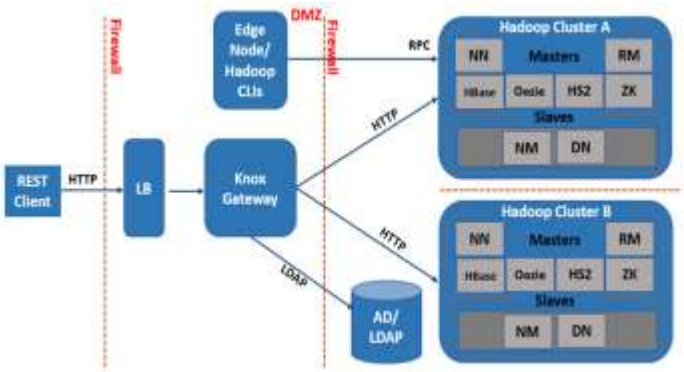


Figure 13: Apache Knox deployment Architecture

NN=NameNode, RM=Resource Manager, DN=DataNote, NM=NodeManager, HS2=HiveServer 2, LB=Load Balancer, CLI=Command Line Interface, RPC=Remote Procedure Call, DMZ=Demilitarized Zone

Typical Security Flow for retrieving data from Hive in a kerborized Hadoop cluster using firewall and request routed through Knox gateway along with authentication by Ranger and distributed directory information services LDAP.

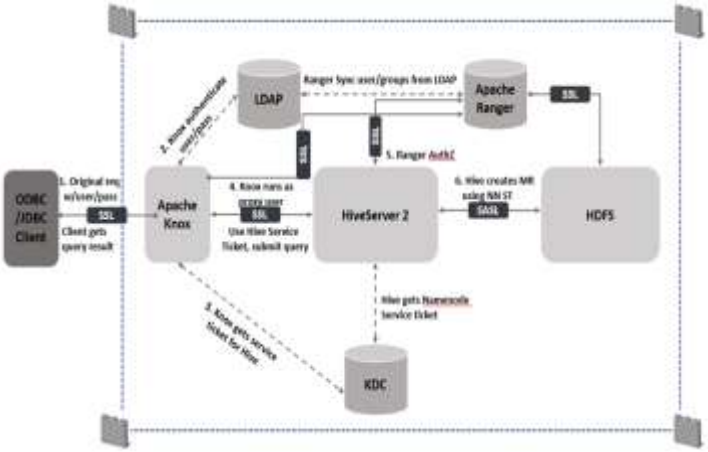


Figure 14: Security flow on data retrieval from Hive through Knox in a Kerborized Hadoop cluster

Supported Hadoop Services

Apache Knox Gateway supports the following Hadoop services versions in both Kerberized and Non-Kerberized clusters:

The following Apache Hadoop ecosystem services have integrations with the Knox Gateway

Ambari and Cloudera Manager	Apache Oozie	Apache Tinkerpop - Gremlin
WebHDFS (HDFS)	Apache Hive/JDBC	Apache Avatica/Phoenix
Yarn RM	Apache Hive WebHCat (Templeton)	Apache SOLR
Stargate (Apache HBase)	Apache Storm	Apache Livy (Spark REST Service)
Apache Ranger	Kafka REST Proxy	Apache Zeppelin

Supported Apache Hadoop ecosystem UIs

Name Node UI	Apache Ambari UI
Job History UI	Apache Ranger Admin Console
Yarn UI	Apache Zeppelin
Apache Oozie UI	Apache NiFi
Apache HBase UI	Hue
Apache Spark UI	Livy and Apache Impala

Mapping examples of WebHDFS, Oozie, Hive JDBC and HBase are as below –

WebHDFS	
Gateway	https://{gateway-host}:{gateway-port}/{gateway-path}/{cluster-name}/webhdfs
Cluster	http://{webhdfs-host}:50070/webhdfs

Oozie	
Gateway	https://{gateway-host}:{gateway-port}/{gateway-path}/{cluster-name}/oozie
Cluster	http://{oozie-host}:11000/oozie}

HBase	

International Journal of Scientific & Engineering Research, Volume 11, Issue 5, May-2020
ISSN 2229-5518

Gateway	https://{gateway-host}:{gateway-port}/{gateway-path}/{cluster-name}/hbase
Cluster	http://{hbase-host}:8080

Hive JDBC	
Gateway	jdbc:hive2://{gateway-host}:{gateway-port}/;ssl=true;sslTrustStore={gateway-trust-store-path};trustStorePassword ={gateway-trust-store-password}; transportMode =http;httpPath={gateway-path}/{cluster-name} /hive
Cluster	http://{hive-host}:10001/cliservice

These mapping are generated from the combination of the

a. Gateway configuration file:{GATEWAY_HOME}/conf/gateway-site.xml) and
b. cluster topology descriptors: {GATEWAY_HOME}/conf/topologies/{cluster-name}.xml).

Recommendations

Apache Knox integration with Apache Ranger is recommended to check the permissions of users who want to access cluster resources. Enabling SSL is highly recommended. In case Hive connections gets disconnected via Apache Knox, modify the connection timeout and number of max connections in gateway-site configuration file.

Improve response times via Apache Knox setting up below properties in gateway.site gateway.metrics.enabled=false, gateway.jmx.metrics. reporting.enabled=false, gateway.graphite.metrics.reporting.enabled=false.

7 Authorization using Apache Ranger

Apache Ranger is a framework that provides a centralized platform to define, administer and manage security policies consistently across Hadoop components. It provides comprehensive security across the Apache Hadoop ecosystem. Using the Apache Ranger console, administrators can easily manage different policies for access to files, databases, tables and columns. These policies can be set for individual users or groups and then enforced consistently across multiple Hadoop stacks.

Encryption of data-at-rest in HDFS is achieved by Ranger Key Management Service (KMS). The Ranger KMS provides a scalable cryptographic key management service for data encryption. Ranger KMS is developed by the Apache community which extends the native Hadoop KMS functionality by allowing system administrators to store keys in a secure database. Ranger also provides centralized audit capabilities that tracks all the access requests in real time and support multiple destination sources including HDFS and Solr.

Apache Ranger has the following goals:

a. Centralized security administration to manage all security related tasks in a central UI or using REST APIs.
b. Fine grained authorization to do a specific action and/or operation with Hadoop component/tool and managed through a central administration tool
c. Standardize authorization method across all Hadoop components.
d. Enhanced support for different authorization methods - Role based access control, attribute-based access control etc.
e. Centralize auditing of user access and administrative actions (security related) within all the components of Hadoop.
f. Dynamic column masking and row level filtering, dynamic policy conditions, classification or tag-based policies for Hadoop ecosystem components

Apache Ranger provides a centralized security framework to manage fine-grained access control across 9 different components.

Apache Hadoop HDFS	Apache Solr
Apache Hive	Apache Kafka
Apache HBase	Apache NiFi
Apache Storm	YARN
Apache Knox	-

Ranger plugin for HDFS checks for Ranger policies and access is granted to user if the policy is defined. If the access policy is not defined in Ranger, then Ranger would default to native permissions model in HDFS (POSIX or HDFS ACL). This federated model is applicable for HDFS and Yarn service in Ranger as well.

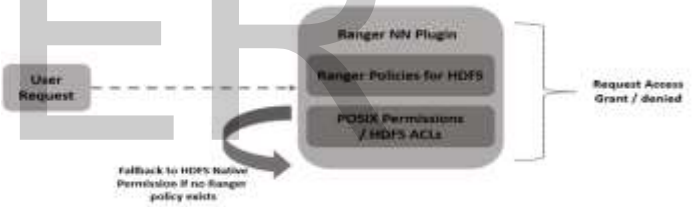


Figure 15: Federated permission model in Ranger

Simplified workflow for accessing Hive –

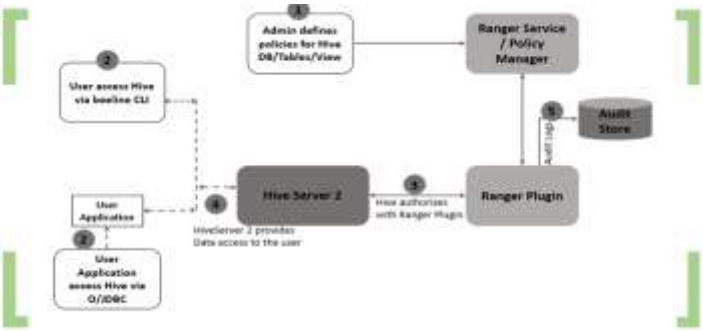


Figure 16: Simplified workflow for accessing Hive

7.1 Row-level filtering

Row-level filtering in Apache Ranger helps to set access policies for rows in Hive tables. It filters and simplifies the hive query by moving the access restriction logic down into the Hive layer. This access restrictions are applied every time when data access is attempted. It controls access to rows in Hive table based on the

International Journal of Scientific & Engineering Research, Volume 11, Issue 5, May-2020
ISSN 2229-5518

context. This feature provides seamless behind-the-scenes enforcement of row-level segmentation without having to add this logic to the predicate of the query.

Through this security features in Ranger, Hive data access can be restricted to specific rows based on user characteristics such as group membership and the runtime context in which this request is issued. Hence, it improves the reliability and robustness of Hadoop cluster. Few of the use cases where row-level filtering can be beneficial include:

a. In Healthcare service, a hospital can create a security policy which filters the patient's data and allows doctors to view data rows only for their own patients, and that allows insurance claims administrators to view only specific rows for their specific site.

b. In financial services, a bank can create a policy to restrict access to rows of financial data based on the employee's department and role. For example - only employees in the finance department are allowed to see customer invoices, payments, and accrual data; only Singapore HR employees can see Singapore employee data.

c. In information technology, a multi-tenant application can create logical separation of each tenant's data so that each tenant can see only their own data rows.

Along with row-level, filters can also be applied to specific users, groups, and conditions as well. Filters are evaluated in the order listed in the policy and an audit log entry is generated each time a row-level filter is applied to a table or view. For example: to restrict users to access only records of customers located in the same country where the user works. US users can only access US customer records and UK users can only access UK customer records. Users belong to one of the country-specific groups maintained in LDAP/AD, as shown in the example below.

Group name	Users
us-employees	NameA, NameB
uk-employees	NameC, NameD, NameE
de-employees	NameF, NameG

Sample records in customer table in Hive.

User Id	Username	Country	Date of Birth	Phone Number
1	NameA	US	1993-12-18	123-411-7890
2	NameB	US	1975-03-22	234-522-8901
3	NameC	UK	1985-01-11	12-3433-7890
4	NameD	UK	1976-05-19	23-4555-8901
5	NameE	UK	1981-07-23	34-5644-9012
6	NameF	DE	1995-09-07	23-477-78901
7	NameG	DE	1999-02-06	34-599-89012

On executing the select query, user 'NameE', a member of uk-employees group can only see the records of customers on UK.

User Id	Username	Country	Date of Birth	Phone Number
3	NameC	UK	1985-01-11	12-3433-7890
4	NameD	UK	1976-05-19	23-4555-8901
5	NameE	UK	1981-07-23	34-5644-9012

Specific query can be defined in row level filter to make the data available for public group.

7.2 Data Masking

Column masking in Hive with Ranger policies is of two types. They are dynamic resource-based and dynamic tag-based column masking. Data masking doesn't physically alter any data or make a copy of it. Sensitive data doesn't leave the data store rather obfuscated when presenting to the user. In addition, no changes are incorporated at the application or hive later for data masking.

Dynamic Resource-Based Column Masking in Hive with Ranger Policies

Apache Ranger dynamic resource-based column masking capabilities are used to protect sensitive data in Hive in near real-time. Policies can be set that anonymize sensitive data columns dynamically from Hive query output. For example, you can mask sensitive data within a column to show only the first or last four characters. With dynamic column-level masking, sensitive information never leaves Hive database and no changes are required at the consuming application or the Hive layer. Producing additional protected duplicate versions of datasets is also not needed.

Several masking options are available in Ranger such as show last 4 characters, show first 4 characters, Hash, Nullify and date masks (show only year). The masking type can be defined for specific users, groups, and conditions and Masks are evaluated in the order listed in the policy. In addition, an audit log entry is generated each time a masking policy is applied to a column.

Dynamic Tag-Based Column Masking in Hive with Ranger Policies

The tag-based masking policy anonymizes Hive column data based on tags and tag attribute values associated with Hive column usually specified as metadata classification in Atlas. If there are multiple tag masking policies applied to the same column in the Hive table, the masking policy with the lexico-graphically smallest policy-name is chosen for enforcement, e.g. policy "a" is enforced before policy "aa". Masking type can be defined for specific users, groups, and conditions and a variety of masking types such as show last 4 characters, show first 4 characters, Hash, Nullify, and date masks can be de-fined. In addition, masks are evaluated in the order listed in the policy and an audit log entry is generated each time a masking policy is applied to a column.

International Journal of Scientific & Engineering Research, Volume 11, Issue 5, May-2020
ISSN 2229-5518

A few use cases to enforce column level masking are (a) show only last 4 digit of the phone number (b) show only year value in date of birth column, (c) hide the data of a particular column and (d) applying a custom transformation to a particular col-umn. Given below is an example of showing the last 4 digits of the phone number of the customer for the public group.

A sample data retrieved from customer table is as below

User Id	Username	Country	Date of Birth	Phone Number
1	NameA	US	1993-12-18	xxx-xxx-7890
2	NameB	US	1975-03-22	xxx-xxx-8901
3	NameC	UK	1985-01-11	xxx-xxx-7890
4	NameD	UK	1976-05-19	xxx-xxx-8901
5	NameE	UK	1981-07-23	xxx-xxx-9012
6	NameF	DE	1995-09-07	xxx-xxx-8901
7	NameG	DE	1999-02-06	xxx-xxx-9012

Recommendations

For HDFS authorization, change HDFS umask from 022 to 077 to prevent any new files or folders to be accessed by anyone other than the owner. The HDFS native permissions for application data directories such as (/apps/hive, /apps/spark as well as any custom data folders need to be restrictive and managed through Apache Ranger.

Auditing in Apache Ranger can be controlled as a policy. Configure SSL for all the plugins enabled

A default policy is created when Apache Ranger is installed through Ambari for all files and directories in HDFS and with auditing option enabled. Ambari uses this policy to do smoke test by "ambari-qa" test user to verify HDFS services. A similar policy for enabling audit across all files and folders must be created if the administrators disable this default policy.

8 Authorization using Apache Sentry

Apache Sentry is a granular, role-based authorization module for Hadoop. It's enforces fine grained role-based authorization to data and metadata stored on a Hadoop cluster. It has the ability to control and enforce precise levels of privileges on data for authenticated users and applications on a Hadoop cluster.

Sentry is designed to be a pluggable authorization engine for Hadoop components. It currently works out of the box with Apache Hive, Hive Metastore/HCatalog, Apache Solr, Impala and HDFS (limited to Hive table data). It allows to define authorization rules to validate a user or application's access requests for Hadoop resources. Sentry is highly modular and can support authorization for a wide variety of data models in Hadoop.

Role-based access control (RBAC) is an important mechanism to manage authorization for a large set of users and data objects in an organization. New data objects get added or removed, users join, move, or leave organizations all the time. Managing objects or users through RBAC is easier. For example - if user A joins the finance department, simply adding the user to the active directory (AD) group i.e. finance-department group will suffice. This will give use A the access to data from the finance department tables i.e. Sales and Customer tables.

Sentry components involved in the authorization process –

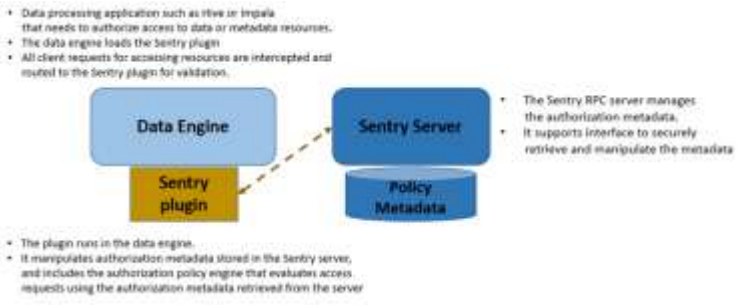


Figure 17: Components of Apache Sentry

Core Capabilities of Sentry:

Fine-grained Authorization	Permissions on object hierarchy - DB, Tables and Columns
Role-based Authorization	Support for role templates to manage authorization for a large set of users and data objects
Multi-Tenant Administration	Ability to delegate admin responsibilities or a subset of resources

Key Concepts in Sentry:

Authentication	Verifying credentials to reliably identify a user
Authorization	Limiting the user's access to a given resource
User	Individual identified by underlying authentication system
Group	A set of users, maintained by the authentication system
Privilege	An instruction or rule that allows access to an object
Role	A set of privileges; a template to combine multiple access rules
Authorization models	Defines the objects to be subject to authorization rules and the granularity of actions allowed. For example: In SQL model, the objects can be databases or tables, and the actions are SELECT, INSERT, CREATE and so on. In Search model, the objects are indexes, collections and documents; the access modes are query, update and so on.

Apache Sentry Architecture

The basic goal of Apache Sentry is to implement authorization for Hadoop ecosystem components in a unified way so that security administrators can easily control what users and groups have access to without needing to know the ins and outs of every single component in the Hadoop stack. The high-level architecture is as below.

International Journal of Scientific & Engineering Research, Volume 11, Issue 5, May-2020
ISSN 2229-5518

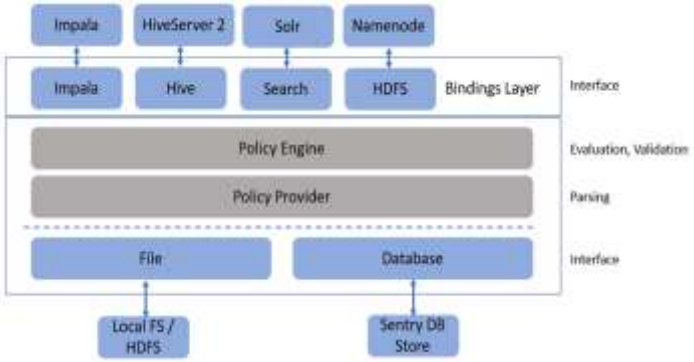


Figure 18: Apache Sentry Architecture

Recommendations

To figure out why Sentry has denied access in a specific instance, enable Sentry's debug level logging temporarily. This will help to address other service failures such as queries to the hive warehouse fail with an authentication error, Impala queries continue to work even if Sentry is down because they are authorized against the cached copy of the metadata but authorization DDLs such as CREATE ROLE or GRANT ROLE will fail.

For HDFS/Sentry Synchronized Permissions, the default timeout value is 60 seconds. Its recommended to increase the timeout value to few minutes as needed to accommodate large clusters.

9 A COMPARATIVE STUDY OF APACHE SENTRY AND APACHE RANGER

Although both Apache Ranger and Apache Sentry are used for authorization, the usage depends upon what Hadoop distribution tool that you are using like Apache, Cloudera or Hortonworks (powered by Apache Hadoop)

Apache Sentry is owned by Cloudera and it supports HDFS, Hive, Solr and Impala. Ranger 2.0.0, the latest version doesn't support Impala. Apache Ranger is owned by Hortonworks (now Cloudera owned) and it offers a centralized security framework to manage fine-grained access control across: HDFS, Hive, HBase, Nifi, Storm, Knox, Solr, Kafka, and YARN with Auditing. Apache Sentry v2.1.0, the latest version of Sentry doesn't support Rest API and audit feature. Apache Ranger supports more features and integrated with more other Hadoop components.

Below are the outlines / a comparative study of both the projects.

Features & Supported Components	Apache Sentry	Apache Ranger
Authorization	Yes	Yes
Row Level Filtering	No	Yes
Column Masking	No	Yes
Support Tag Based	No	Yes
Denial Support	No	Yes
Rest API	No	Yes
Command Line	Yes	No
Audit	No	Yes
Nifi	No	Yes
Impala	Yes	No
Hive	Yes	Yes
HDFS	Yes	Yes
Solr	Yes	Yes
Kafka	Yes	Yes
HBase	No	Yes
Knox	No	Yes
YARN	No	Yes
Storm	No	Yes

10 DATA PROTECTION

Data protection through data encryption is required by a number of different governments, financial, and regulatory entities. For example, the health-care industry has HIPAA (Health Insurance Portability and Accountability Act) regulations, the card payment industry has PCI DSS (The Payment Card Industry Data Security Standard) regulations, and the US government has FISMA (The Federal Information Security Management Act) regulations. Having transparent encryption built into HDFS makes it easier for organizations to comply with these regulations. Encryption can be performed at the application-level but if it is integrated through HDFS, the existing applications can operate on encrypted data without any changes. This integrated architecture implies stronger encrypted file semantics and better coordination with other HDFS functions. Through transparent, end-to-end HDFS encryption, data read from and written to special HDFS directories is transparently encrypted and decrypted without requiring changes to user application code.

Data encryption is broadly categorized into encryption of data-at-rest i.e. encrypting data on persistent storage and encryption of data in-motion / transit i.e. encrypting data travelling over the network.

10.1 Encryption of Data at Rest

HDFS data-at-rest encryption implements end-to-end encryption of data read from and written to HDFS. End-to-end encryption means that data is encrypted and decrypted only by the client. HDFS does not have access to unencrypted data or keys. HDFS encryption involves elements such as encryption key, encryption zone and Ranger KMS.

An Encryption key is a new level of permission-based access protection, in addition to standard HDFS permissions. HDFS encryption zone which is a special HDFS directory within which all data is encrypted upon write and decrypted upon read. Each encryption zone is associated with an encryption key that is specified when the zone is created. Each file within an encryption zone has

International Journal of Scientific & Engineering Research, Volume 11, Issue 5, May-2020
ISSN 2229-5518

a unique encryption key, called the "data encryption key" (DEK). HDFS does not have access to DEKs. HDFS DataNodes only see a stream of encrypted bytes. HDFS stores "encrypted data encryption keys" (EDEKs) as part of the file's metadata on the NameNode. Clients decrypt an EDEK and use the associated DEK to encrypt and decrypt data during write and read operations.

Ranger Key Management Service (Ranger KMS) is an open source key management service based on Hadoop's KeyProvider API. It provides a scalable cryptographic key management service for HDFS "data at rest" encryption. It extends the native Hadoop KMS functionality by allowing system administrators to store keys in a secure database. For HDFS encryption, the Ranger KMS has three basic responsibilities: (a) Provide access to stored encryption zone keys. (b) Generate and manage encryption zone keys and create encrypted data keys to be stored in Hadoop. (c) Audit all access events in Ranger KMS.

Transparent Data Encryption flow -

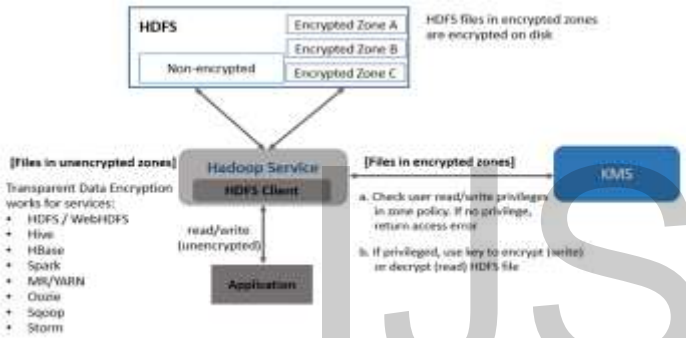


Figure 19: Transparent Data Encryption flow

The overall workflow is as follows:

a. Create an HDFS encryption zone key that will be used to encrypt the file-level data encryption key for every file in the encryption zone. This key is stored and managed by Ranger KMS.
b. Create a new HDFS folder. Specify required permissions, owner, and group for the folder.
c. Using the new encryption zone key, designate the folder as an encryption zone.
d. Configure client access. The user associated with the client application needs enough permission to access encrypted data. In an encryption zone, the user needs file/directory access (through POSIX permissions or Ranger access control) as well as access for certain key operations.
e. After permissions are set, Java API clients and HDFS applications with enough HDFS and Ranger KMS access privileges can write and read to/from files in the encryption zone.

10.2 Encryption of data in-motion/in-transit

There are several ecosystems in Hadoop from Spark and Sqoop to Kafka and Kudu. These services communicate with each other through different protocols including HTTP, RPC or TCP/IP and each protocol has a different method to encrypt data. To achieve secure communications in Hadoop we need to enable the secure version of protocols used to protect data in-transit. As data and credentials i.e. username and password go in and out of the cluster, and network configuration (firewall) can mitigate some risk, encrypting data in transit is indeed required.

RPC/SASL

The most common way for a client to interact with a Hadoop cluster is through Remote Procedure Call (RPC). A client connects to a NameNode over RPC protocol to read or write a file. RPC connections in Hadoop use the Java Simple Authentication and Security Layer (SASL) which supports encryption. When the hadoop.rpc.protection property is set to privacy in core-site.xml file, the data over RPC is encrypted with symmetric keys. Authentication provides authentication between the two parties. Integrity provides authentication between the two parties and message integrity and privacy provides confidentiality. RPC encryption covers not only the channel between a client and a Hadoop cluster but also the inter-cluster communication among Hadoop services.

TCP/IP

TCP/IP communication protocol is mainly used for communication between Datanodes. TCP/IP doesn't have encryption built in or directly supported by default. To address this deficiency, the existing data transfer protocol is wrapped with SASL handshake. SASL support encryption.

All HDFS communication protocols build on the TCP/IP protocol. HDFS clients connect to a Transmission Control Protocol (TCP) port opened on the name node, and then communicate with the name node using a proprietary Remote Procedure Call (RPC)-based protocol. Data nodes talk to the name node using a proprietary block-based protocol. To enable this TCP/IP encrypted data stream, set the dfs.encrypt.data.transfer property to "true" in the hdfs-site.xml configuration file. This configuration change must be made on both the NameNode and DataNodes.

TLS/HTTPS

HTTPS is a proven and widely adopted standard for encrypting HTTP communications. Hadoop uses HTTP for its web Interfaces, MapReduce shuffle phase and for FSimage operations between the NameNode and Secondary NameNode. Java and browsers support HTTPS and many libraries and tolls in most operating systems have built-in support for HTTPS. Different Apache Hadoop components have been developed in different coding languages – for example, MapReduce in Java or Hue with Python. Hence, there are different ways and locations where the SSL/TLS is configured.

For example - To enable the encrypted WebUI for MapReduce v2, edit the core-site.xml file setting the hadoop.ssl.enabled property to "true". To enable encrypted shuffle for MapReduce v2, edit the mapred-site.xml file and set the mapreduce.shuffle.ssl.enabled property to "true".

The Hadoop SSL Keystore Factory manages SSL for core services that communicate with other cluster services over HTTP, such as MapReduce, YARN, and HDFS. Other components that have services that are typically not distributed, or only receive HTTP connections directly from clients, use built-in Java JDK SSL tools. Examples include HBase and Oozie. Services that are co-located on a host must configure the server certificate and keys, and in some cases the client truststore. The certificate must be managed by Hadoop SSL Keystore management Factory and by JDK. When using CA signed certificates, configure the Hadoop SSL Keystore Factory to use the Java keystore and truststore locations. Hadoop SSL Keystore Management Factory provides

below features.

a. Supports only JKS formatted keys.
b. Supports toggling the shuffle between HTTP and HTTPS.
c. Supports two-way certificate and name validation.
d. Uses a common location for both the keystore and truststore that is available to other Hadoop core services.
e. Allows you to manage SSL in a central location and propagate changes to all cluster nodes.
f. Automatically reloads the keystore and truststore without restarting services.

SSL Management with JDK supports below features.

a. Allows either HTTP or HTTPS.
b. Uses hard-coded locations for truststores and keystores that may vary between hosts. Typically, this requires you to generate key pairs and import certificates on each host.
c. Requires the service to be restarted to reload the keystores and truststores.
d. Requires certificates to be installed in the client CA truststore.

Its recommended to use LDAP over SSL (LDAPS) rather than LDAP when communicating with the corporate enterprise directories to prevent sniffing attacks.

Recommendations

Periodic key rolling policies are recommended to protect the data from rogue users who collect keys to which they have access and use them later to decrypt encrypted data. Enable trash to prevent accidental deletion of files and directories.

The attacker may gain physical access to swap files of processes containing data encryption keys. This can be mitigated by disabling swap, using encrypted swap, or using mlock to prevent keys from being swapped out.

11 Access Control List (ACL)

Access Control Lists (ACLs) are useful for implementing permission requirements that differ from the natural organizational hierarchy of users and groups. An ACL provides a way to set different permissions for specific named users or named groups, not only the file's owner and the file's group. Along with traditional POSIX permissions model, HDFS supports POSIX Access Control Lists (ACLs) as well. The HDFS ACLs provide a fine-grained file permissions model that is appropriate for a large enterprise where the data stored on the Hadoop cluster and the data is accessible to some groups but inaccessible to many others. An ACL consists of a set of ACL entries. Each ACL entry names a specific user or group and grants or denies read, write and execute permissions for that specific user or group. There are two types of ACL rules:

a. access ACLs: Specify access information for a single file or directory
b. default ACLs: Pertain to a directory only. It specifies default access information for any file within the directory that does not have an access ACL.

Different permissions are assigned to different users and group through HDFS extended ACLs.

Figure 20: HDFS extended ACLs permissions

setfacl (Syntax: -setfacl --set <acl_spec> <path>) and getfacl (Syntax: -getfacl [-R] <path>) are utilities to create and list ACLs on HDFS.

Option	Description
-b	Remove all entries but retain the base ACL entries. The entries for User, Group, and Others are retained for compatibility with Permission Bits.
-k	Remove the default ACL.
-R	Apply operations to all files and directories recursively.
-m	Modify the ACL. New entries are added to the ACL, and existing entries are retained.
-x	Remove the specified ACL entries. All other ACL entries are retained.
--set	Fully replace the ACL and discard all existing entries. The acl_spec must include entries for User, Group, and Others for compatibility with Permission bits.
<acl_spec>	A comma-separated list of ACL entries.
<path>	The path to the file or directory to modify.

```
hdfs dfs -setfacl -m user:hadoop:rw- /file
hdfs dfs -setfacl -x user:hadoop /file
hdfs dfs -setfacl -b /file
hdfs dfs -setfacl -k /dir
hdfs dfs -setfacl --set user::rw-,user:hadoop:rw-,group::r--,other::r-- /file
hdfs dfs -setfacl -R -m user:hadoop:r-x /dir
hdfs dfs -setfacl -m default:user:hadoop:r-x /dir
```

Displays the ACLs of files and directories. If a directory has a default ACL, getfacl also displays the default ACL.

Option	Description
-R	List the ACLs of all files and directories recursively.
<path>	The path to the file or directory to list.

Example:

```
hdfs dfs -getfacl /file
hdfs dfs -getfacl -R /dir
```

Recommendations

Rely on traditional permission bits to implement most permission requirements and define a smaller number of ACLs to augment the permission bits with a few exceptional rules. A file with an ACL incurs an additional cost in memory in the NameNode compared to a file that has only permission bits.

12 Hadoop Group Mapping for LDAP AD with SSSD

The System Security Services Daemon (SSSD) provides a set of daemons to manage access to local or remote identity and authentication resources through a common framework that can

International Journal of Scientific & Engineering Research, Volume 11, Issue 5, May-2020
ISSN 2229-5518

provide caching and offline support to the system. It provides Name Service Switch (NSS) and Pluggable Authentication Modules (PAM) interfaces toward the system and a pluggable back end system to connect to multiple different account sources. SSSD provides interfaces towards several system services. Most notably:

c. NSS (Name Service Switch) which includes passwd (Password), shadow (User Group), groups (Groups).
d. PAM provider service
e. SSH Provider service
f. autofs
g. sudo proivder service

To ensure that LDAP/AD group level authorization is enforced in Hadoop, Hadoop group mapping setup for LDAP/AD must be performed. There are three ways to set up Hadoop group mapping. They are as below.

a. Configure Hadoop Group Mapping for LDAP/AD using SSSD
b. Configure Hadoop Group Mapping in core-site.xml and
c. Manually create the users and groups in the Linux environment.

Setting up Hadoop group mapping for AD/LDAP using SSSD over SSL is highly recommended.

Configure Hadoop Group Mapping for LDAP/AD Using SSSD

The recommended method for group mapping is to use SSSD or one of the following services to connect the Linux OS with LDAP over SSL. The services given below allow to not only look up a user and enumerate their groups, but also allow you to perform other actions on the host. NSS is required because it performs user/group resolution. PAM module performs user authentication and may represent a security risk.

a. Centrify
b. NSLCD
c. Winbind
d. SAMBA

In core-site.xml, configure Hadoop to use LDAP-based group mapping. You will need to provide the value for the bind user, the bind password, and other properties specific to you LDAP instance, and make sure that object class, user, and group filters match the values specified in your LDAP instance. For LDAP over SSL, the keystore and truststore have to be configured.

Depending on configuration, execute refresh user and group mappings using the following HDFS and YARN commands.

hdfs dfsadmin -refreshUserToGroupsMappings

yarn rmadmin -refreshUserToGroupsMappings

Verify LDAP group mapping to fetch groups from LDAP for the current user by running the “hdfs groups” command. With LDAP group mapping configured, the HDFS permissions can leverage groups defined in LDAP for access control. Both nss_ldap and pam_krb5 are a common configuration to implement LDAP identification and Kerberos authentication in enterprise environments. RedHat deprecated pam_krb5 with the RedHat Enterprise Linux 7.4 (RHEL) release and does not plan to include pam_krb5 with RHEL 8.

The target replacement for pam_krb5 is SSSD that is already available in both RHEL 6 and RHEL 7. SSSD implements features which are not available in a standard nss_ldap/pam_krb5 configuration, such as authentication when AD is offline, better performance and LDAP ID-mapping. Its architecture is to reside between NSS/PAM and the LDAP/Kerberos systems, and to make requests to LDAP/Kerberos on behalf of NSS/PAM as shown below.

Figure 21: System Security Services Daemon Connectivity

Recommendations

Use SSL (hadoop.security.group.mapping.ldap.ssl) to make the communication between Hadoop cluster and directory server secure. Its recommended not to use bind password (hadoop.security.group.mapping.ldap.bind.password) instead put the password in a separate file (hadoop.security.group.mapping.ldap.bind.password.file)

13 SSL FOR HADOOP WEB COMPONENTS

SSL is a security protocol that allows encrypted communication between a web server and internet browser. It encrypts all data transmitted between the server and the user using an encryption key on the server. An SSL certificate ensures that the information being transmitted between the web server and browser is only visible to the user and the website. HTTPS uses TLS (transport layer security) or SSL (secure sockets layer) for encryption. These unique encryption keys are transmitted between the web browser and server to keep the communication safe. The goal of SSL/TLS is to make it safe and secure to transmit sensitive information including personal data, payment or login information. It makes it difficult for hackers to spy on the connection and steal the data. Given below are some of the Hadoop web components where HTTPS can be configured.

Ambari Web UI
Namenode UI
Resource Manager UI
Ranger Admin UI
Zeppelin Notebook UI
Oozie Web Console

International Journal of Scientific & Engineering Research, Volume 11, Issue 5, May-2020
ISSN 2229-5518

HBase Master UI
Hive Server 2/Interactive UI
Kafka Web UI
Falcon Web UI
Knox Gateway Web UI
Accumulo Web Monitor UI
Atlas Web UI

Recommendations

Enable TLS/SSL for all web components in Hadoop cluster to provide encrypted communication and secure identification of the server. In addition, use robust security certificates as a part of enabling HTTPS for Hadoop web UI

14 Data Governance

Data Governance is a system of decision rights and accountabilities for information-related processes, executed according to agreed-upon models which describe who can take what actions with what information, and when, under what circumstances, using what methods.

Apache Atlas provides governance capabilities for Hadoop and it uses both prescriptive and forensic models enriched by business taxonomical metadata. Atlas is designed to exchange metadata with other tools and processes within and outside of the Hadoop stack, thereby enabling platform-agnostic governance controls that effectively address compliance requirements. The core governance services include search and prescriptive lineage which facilitates pre-defined and ad hoc exploration of data and metadata, while maintaining a history of data sources and how specific data was generated. The services also include flexible modeling of both business and operational data, metadata-driven data access control and classification of data to help understand the nature of data i.e. internal or external sources.

It provides an intuitive UI to view lineage of data as it moves through various processes and a REST APIs to access and update lineage. From search and discovery standpoint, it provides rich REST APIs to search by complex criteria such as search entities by type, classification, attribute value or free-text along with SQL like query language to search entities - Domain Specific Language (DSL). Atlas provides metadata services for the following components:

a. Hive
b. Ranger
c. Sqoop
b. Storm/Kafka (limited support)
c. Falcon (limited support)

Features	Description
Knowledge store	Leverages existing Hadoop metastores. Categorized into a business-oriented taxonomy of data sets, objects, tables, and columns. Supports the exchange of metadata between Hadoop components and third-party applications or governance tools.
Data lifecycle management	Leverages Apache Falcon with a focus on provenance, multi-cluster replication, data set retention and eviction, late data handling, and automation.
Security	Leverages Apache Ranger plug-in architecture for security policy enforcement.
Audit store	Historical repository for all governance events, including security events (access, grant, deny), operational events related to data provenance and metrics. The Atlas audit store is indexed and searchable for access to governance events.
RESTful interface	Supports extensibility by way of REST APIs to third-party applications
Policy engine	Fully extensible policy engine that supports metadata-based, geo-based, and time-based rules that rationalize at runtime

Recommendations

Data governance principles apply to both used and unused data. Applying governance policies to unused data should not be discarded because Big Data Governance requires an adjustment of conventional policies and practices on both used and unused data in the data lake so that the untapped potential of Big Data is not constrained by governance.

15 Auditing and Reporting

Audit logging is an accounting process that logs all operations happening in Hadoop cluster. It is important to understand where the data in the cluster is coming from and how it's being used. The goal of auditing is to capture a complete and immutable record of all activity within a system. Auditing plays a central role in three key activities within the enterprise.

a. Auditing is part of a system's security regime and can explain what happened, when, and to whom or what in case of a breach or other malicious intent. For example, if an administrator deletes a user's data set, auditing provides the details of this action, and the correct data may be retrieved from backup.
b. In terms of compliance, auditing participates in satisfying the core requirements of regulations associated with sensitive or personally identifiable data (PII), such as the Health Insurance Portability and Accountability Act (HIPAA) or the Payment Card Industry (PCI) Data Security Standard. Auditing provides the touchpoints necessary to construct the trail of who, how, when, and how often data is produced, viewed, and manipulated.
c. Auditing provides the historical data and context for data forensics. Audit information leads to the understanding of how various populations use different data sets and can help establish the access patterns of these data sets.

The risks facing auditing are the reliable, timely, and tamper-proof capture of all activity, including administrative actions. Until recently, the native Hadoop ecosystem has relied primarily on using log files. Log files are unacceptable for most audit use cases in the enterprise as real-time monitoring is impossible, and log mechanics can be unreliable - a system crash be-

International Journal of Scientific & Engineering Research, Volume 11, Issue 5, May-2020
ISSN 2229-5518

fore or during a write commit can compromise integrity and lead to data loss.

Apache Ranger provides the capability of centralized auditing for Apache Hadoop. Ranger auditing data can be stored in multiple locations including Apache Solr and HDFS. With HDFS storing audit logs for compliance purposes, we needed a way to query these logs. Apache Hive provides the ability to query HDFS data without a lot of effort. Apache Ranger provides comprehensive scalable audit logging for resource access events with user context, policy edits/creation/deletion, user session information and component plugin policy sync operations.

Apache Ranger v2.1.0 supports fine grained authorization and auditing for following Apache projects.

d. Apache Hadoop
e. Apache Hive
f. Apache HBase
g. Apache Storm
h. Apache Knox
i. Apache Solr
j. Apache Kafka
k. YARN

Apache Ranger provides auditing capabilities at Access, Admin, Login Session and Plugins levels. The Access page provides service activity data for all policies that have audit set to on. The default service policy is configured to log all user activity within the Service. This default policy does not contain user and group access rules. The admin tab contains all events for the auditing security administration web UI, including Service, Service Manager, Log in, etc. and actions like create, update, delete, password change. The Login Sessions tab logs the information related to the sessions for each login. The plugins tab shows the upload history of the security agents. This module displays all the services exported from the system. The Plugin Status tab shows policies in effect for each plugin. That includes the relevant host info and when the plugin downloaded and started enforcing the policies. The user sync page provides service activity data for all usersync processes in Ranger. This creates a compliance/audit trail for users and groups synchronized with each run of userync.

Apache Ranger can use Apache Solr to store audit logs. Apache Solr is an open-source enterprise search platform. It provides a search capability of the audit logs through the Ranger Admin UI. Audits to Solr are primarily used to enable search queries from the Ranger Admin UI. HDFS is a long-term destination for audits - audits stored in HDFS can be exported to any SIEM system, or to another audit store. Ranger uses Apache Solr to store audit logs and provides UI searching through the audit logs.

Recommendations

Due to regulation and compliance, audit logs should be stored in a database for immediate availability and in HDFS for long-term storage.

16 Conclusion

As major data breaches are hitting well-known entities, implementation of data security cannot be ignored. The effects of a data security breach can be catastrophic. Enterprises putting sensitive data to good use must handle the inherent security challenges. Fake data generation is one of them. Without security measures, a financial company may get false flag in their data and may not be able to identify it as fraud which is caused by fake data generation. A manufacturing company is not an exception. It may get a false temperature report which results in slowdown in production and loss of revenue.

Granular access control helps to grant different user's different levels of access to database and applications. Coarse-grained permissions that give users all or nothing access is no longer enough for hundreds or thousands of employees who need access to data for many different usages. Instead, a scalable and fine-grained access control that prevents unnecessary access to sensitive information at every stage of processing is utmost needed. HDFS extended ACLs encompass the HDFS permission model to support more granular file access based on arbitrary combinations of users and groups.

Data provenance helps to identify where a breach comes from and it helps to apply a right technique to track the flow of data using metadata. As data which can be structured, semi-structured and unstructured flows from a different number of source systems and processed both batch and real-time, tracking it can be difficult without the right framework. Apache Atlas provides efficient metadata management and governance capabilities for organizations to build a catalog of their data assets, classify and govern these assets and provide collaboration capabilities around these data assets for data scientists, analysts and the data governance team.

Hadoop in the enterprise can no longer get by with simplistic means for identifying and trusting users and services. Implementing Kerberos in Hadoop data lake is utmost necessary for strong authentication and identity propagation for users, applications and services. Knox Gateway provides a single point of secure access for Hadoop clusters. It exposes access to Hadoop clusters through a Representational State Transfer (RESTful) API. The gateway provides a firewall between users and Hadoop clusters.

Standardize authorization method across all Hadoop components is efficiently achieved by Apache Ranger. Centralized security administration to manage all security related tasks in a central UI or using REST APIs make the administrative work easy. Implementing right set of tools which ensures confidentiality, availability and integrity for data security is necessary for data management especially when big data is concerned. This paper highlighted different aspects of data security and its challenges and how to leverage Apache Hadoop components to mitigate the threats. The paper outlined different data security solutions for Authentication, Authorization, Data Encryption, Data Masking, Row level filtering, High Availability, Disaster Recovery, Data Governance, Auditing and Reporting. Security component upgrade in the existing data lake is important as new security features are constantly being developed and support by the Hadoop vendors.

There is no all-purpose solution to the specific enterprise security requirements. I believe that this paper will provide insights on data security challenges and provide solutions to

International Journal of Scientific & Engineering Research, Volume 11, Issue 5, May-2020
ISSN 2229-5518

mitigate the threats by leveraging Apache Hadoop with the help of right security strategy.

Disclaimer:

The focus of this paper is to share the challenges in data security and how to mitigate the threats by leveraging Apache Hadoop. The information shared in the paper are based on my experience and research. The ideas and recommendations portrayed in the paper can be used as references for on-premises and/or cloud Big Data solutions.

REFERENCES

[1] https://ranger.apache.org/
[2] https://sentry.apache.org/
[3] http://atlas.apache.org/#/
[4] https://knox.apache.org/
[5] https://hadoop.apache.org/
[6] https://web.mit.edu/kerberos/
[7] https://en.wikipedia.org/wiki/SPNEGO
[8] https://docs.cloudera.com/
[9] https://docs.cloudera.com/HDPDocuments/HDP3/HDP-3.1.5/
[10] O'Reilly - Hadoop Security: Protecting Your Big Data Platform, Author- Ben Spivey, Joey Echeverria, ISBN-13: 978-1491900987, ISBN-10: 1491900989
[11] Security, Privacy, and Forensics Issues in Big Data (Advances in Information Security, Privacy, and Ethics), Author - Ramesh C. Joshi and Brij B. Gupta, ISBN-13: 978-1522597421, ISBN-10: 1522597425.
[12] Data Infrastructure for Next-Gen Finance, Author - Jane Roberts, Released June 2016, Publisher(s): O'Reilly Media, Inc. ISBN: 9781491959664.
[13] Hadoop: Data Processing and Modelling, Author - Garry Turkington, Tanmay Deshpande, Sandeep Karanth, ASIN: B01LD8K59S
[14] Big Data Analytics with Hadoop 3.0, Author - Sridhar Alla, ISBN: 9781788628846
[15] Expert Hadoop Administration: Managing, Tuning, and Securing Spark, YARN, and HDFS (Addison-Wesley Data & Analytics Series), Author - Sam R. Alapati, ISBN: 0134597192
[16] Architecting Modern Data Platforms: A Guide to Enterprise Hadoop at Scale, Author - Jan Kunigk, Ian Buss, Paul Wilkinson, Lars George, Published at: 03/01/2019, ISBN: 149196927X
[17] Big Data Governance: An Emerging Imperative, Author - Sunil Soares, 1583473777, 9781583473771
[18] Data Governance: How to Design, Deploy and Sustain an Effective Data Governance Program (The Morgan Kaufmann Series on Business Intelligence), Author - John Ladley, ISBN-13: 978-0124158290, ISBN-10: 0124158293
[19] Hortonworks Data Platform, an open-architecture platform to manage data in motion and at rest, IBM Analytics, https://www.ibm.com/downloads/cas/DKWR4KZB
[20] Apache Hadoop. 2019. Apache Hadoop 3.x HDFS Federation Features. http://hadoop.apache.org/docs/r3.2.0/hadoop-project-dist/hadoop-hdfs/Federation.html
[21] Hortonworks Security White paper – http://hortonworks.com/wp-content/uploads/2015/07/Security_White_Paper.pdf
[22] Ready Solutions for Data Analytics: Hortonworks Hadoop 3.0 white papers
[23] https://www.dellemc.com/resources/ja-jp/asset/white-papers/solutions/h17561-hortonworks-hadoop-v3-ra.pdf
[24] Big Data Management and Security: Audit Concerns and Business Risks
[25] https://chapters.theiia.org/Orange%20County/IIA%20OC%20Presentation%20Downloads/2015-03-%20IAA%20Big%20Data%20Security.pdf
[26] Simple and Protected GSSAPI Negotiation Mechanism (SPNEGO): https://en.wikipedia.org/wiki/SPNEGO
[27] New Approaches Required for Comprehensive Hadoop Security: https://www.dataguise.com/new-approaches-required-for-comprehensive-hadoop-security-3/
[28] Unprotected Database Exposed 5 Billion Previously Leaked Records: https://www.securityweek.com/unprotected-database-exposed-5-billion-previously-leaked-records
[29] 6 Big Data Security Issues for 2019 and Beyond: https://rtslabs.com/6-big-data-security-issues-for-2019-and-beyond/
[30] Marriott Data Breach: https://www.forbes.com/sites/kateoflahertyuk/2019/03/11/marriott-ceo-reveals-new-details-about-mega-breach/#12dbeae155c0
[31] 2019 on Track for Another 'Worst Year on Record': https://www.riskbasedsecurity.com/2019/08/15/2019-on-track-for-another-worst-year-on-record/
[32] Lenovo Confirms 36TB Data Leak Security Vulnerability: https://www.forbes.com/sites/daveywinder/2019/07/17/lenovo-confirms-36tb-data-leak-security-vulnerability/#7df19f9e62b9
[33] Facebook Security Breach Exposes Accounts of 50 Million Users: https://www.nytimes.com/2018/09/28/technology/facebook-hack-data-breach.html
[34] Australia data breach: 90,000 staff, students, suppliers impacted at Melbourne Polytechnic: https://www.databreaches.net/australia-data-breach-90000-staff-students-suppliers-impacted-at-melbourne-polytechnic/
[35] Health Data Breach Tally Spikes in Recent Weeks: https://www.databreachtoday.com/health-data-breach-tally-spikes-in-recent-weeks-a-14031