Project Title: Unveiling Unauthorized Access in Hadoop Distributed File Systems

1. ABSTRACT

In the rapidly evolving digital landscape, managing large datasets securely and efficiently has become a critical challenge. Our project aims to address this challenge by developing a system that leverages Hadoop's Distributed File System (HDFS) with Role-Based Access Control (RBAC) for secure and efficient file operations. This system is crucial for scenarios where data security and user access management are paramount. The crux of our project lies in balancing high levels of security with user accessibility in HDFS. Existing systems often struggle with this balance, either sacrificing security for accessibility or vice versa. Our solution integrates a robust RBAC mechanism to control user access without complicating the user experience, ensuring that authorized users can access data efficiently and securely. Our methodology involves setting up a Hadoop environment, configuring it for RBAC, and integrating these components in a Java environment for added security. We also used Secure Shell (SSH) for secure remote access and operation. This comprehensive system aims to enhance the security and efficiency of file operations in HDFS. The "Hadoop RBAC Manager" outlines a scenario focused on managing RBAC within a Hadoop environment. In distributed computing frameworks like Hadoop, securing data and managing user permissions are critical challenges. The RBAC model simplifies the administration of security policies by assigning permissions to roles rather than individual users, ensuring that only authorized users can access specific data sets or perform certain operations. Overall, the project aims to improve the security and efficiency of managing large datasets in HDFS by implementing a balanced approach to security and user accessibility through RBAC.

2. INTRODUCTION

1.1 Background and Importance

Hadoop is an open-source framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. However, managing access to data and resources in such an environment can be complex due to the scale and distributed nature of Hadoop.

Role-Based Access Control (RBAC) is a method of regulating access to computer or network resources based on the roles of individual users within an enterprise. RBAC helps in securing data within a Hadoop cluster by ensuring that only authorized users can access specific data sets or perform certain operations.

1.2 Objective & Scope of the Project

Objective:

To develop a system that facilitates the management of access permissions for files stored on HDFS, leveraging the principles of Role-Based Access Control (RBAC). The system aims to ensure that only authorized users can perform specific actions (e.g., reading, writing, deleting) on the files, based on their assigned roles or permissions. This is intended to enhance security, compliance, and data management efficiency within Hadoop environments.

Scope:

1. **Hadoop Setup and Configuration:**

   Installation and configuration of Hadoop, SSH (for secure communication), and Java dependencies necessary for Hadoop's operation. This includes setting up environmental variables, Hadoop's core and HDFS configurations, and ensuring that Java is correctly configured for Hadoop's use.

2. **Secure SSH Configuration:**

   Configuration of SSH to facilitate secure communication without the need for manual intervention during authentication. This includes generating SSH keys, configuring SSH to bypass strict host checking, and setting up authorized keys for seamless access.

3. **Hadoop File System (HDFS) Initialization and Management:**

   Initializing the HDFS name node and data nodes, formatting HDFS, and starting HDFS daemons. The operations include managing the HDFS directory structure and permissions, enabling basic file operations like uploading, deleting, and downloading files.

4. **RBAC Implementation for File Access on HDFS:**

   Implementing rudimentary RBAC functionalities to manage access to files stored on HDFS. This involves creating functions to upload, delete, and download files with checks to ensure that operations are performed only by users with the appropriate permissions. The system uses environment variables to manage user roles and permissions, ensuring that file operations respect the security constraints defined by the RBAC model.

5. **User Interaction and Automation:**

   Providing a user-friendly interface and automated processes for interacting with the HDFS and managing file access. This includes for setting up the environment, functions that abstract complex operations, and integration with the Google Colab interface for file uploads and downloads.

3. PROBLEM DEFINITION

3.1 Problem Statement

Despite the evolution of Hadoop, its initial oversight in prioritizing security left it vulnerable to unauthorized access and potential data compromise. Hadoop consists of prevalent security issues in Hadoop's Distributed File System (HDFS). The occurrence of a security problem in components can endanger the work of the entire system.

3.2 Problem Illustration

Hadoop, a cornerstone in the realm of big data processing, has undergone significant evolution to meet the escalating demands of large-scale data analytics. However, amidst its strides in performance and scalability, a critical oversight emerged – the initial neglect in prioritizing security. This lapse has rendered Hadoop susceptible to unauthorized access and potential compromise of valuable data. The structural weakness opens a gateway for unauthorized users to breach the system, jeopardizing the integrity and confidentiality of the stored data. This loophole not only compromises individual components but also poses a systemic threat, as a security breach in one part can reverberate across the entire Hadoop ecosystem. The current Hadoop system lacks a robust interface with secure login functionalities and rolebased access control (RBAC). Without such features, there is a risk of unauthorized access to sensitive data stored in the Hadoop Distributed File System (HDFS). Additionally, there's a need to enforce permissions such that only the creator of a file has the ability to modify it, while other users are restricted to read-only access. Failure to address these issues may result in data breaches, unauthorized modifications, and compromised data integrity within the Hadoop ecosystem.

4. LITERATURE SURVEY

Existing Technologies on Hadoop RBAC Manager

In a sophisticated Hadoop RBAC Manager, operations are part of a comprehensive system that includes detailed role definitions, policy management, and audit and compliance features. Specific roles, such as administrator, data scientist, and data engineer, are defined with corresponding sets of permissions. Policies are established to dictate the actions each role can or cannot perform within the Hadoop Distributed File System (HDFS), including access to directories and specific operations.

Additionally, the system includes robust audit and compliance functionalities to track user actions and changes to roles or permissions. This tracking is essential for security auditing and ensuring compliance with regulatory requirements. To enhance security and RBAC capabilities, tools and extensions like Apache Ranger and Apache Sentry are commonly used with Hadoop. These tools provide fine-grained access control, advanced policy management, and auditing capabilities that are seamlessly integrated with Hadoop's ecosystem, further enhancing security and compliance measures.

Role-Based Access Control (RBAC) in Hadoop is a fundamental security framework that regulates access to resources based on the roles assigned to users. This approach is essential for efficiently managing permissions across a potentially large number of users and resources within a Hadoop cluster. The RBAC model simplifies permission management in several key ways. First, it involves defining roles and associating specific data access permissions and actions (such as read, write, and execute) with these roles rather than with individual users. Second, users are assigned roles that automatically grant them the associated permissions, making it easier to manage permissions by modifying roles rather than individual user permissions. Lastly, RBAC facilitates auditing and compliance efforts by allowing for easier tracking of data access and manipulation through role assignments and permissions, which is crucial for ensuring compliance with data protection regulations.

[1] Big Data Security Challenges: Hadoop Perspective. International Journal of Pure and Applied Mathematics

Authors: Saraladevi, B., Pazhaniraja, N., Paul, P.V., Basha, M.S.S. and Dhavachelvan, P.

The paper presents three approaches to enhance security within the Hadoop Distributed File System (HDFS):

Kerberos Mechanism: This approach utilizes the Kerberos network authentication protocol to secure connections within HDFS. It involves using tickets to authenticate nodes and clients, ensuring that only authorized users can access data blocks. The Ticket Granting Ticket (TGT) and Service Ticket play crucial roles in authenticating connections between clients and the NameNode.

Bull Eye Algorithm Approach: This novel approach focuses on ensuring the security of sensitive information stored within Hadoop. The Bull Eye Algorithm scans data nodes in a 360° view to identify any risks or vulnerabilities. By implementing this approach, it aims to provide comprehensive security coverage for data stored within HDFS, minimizing the chances of data breaches or unauthorized access.

Namenode Approach: This approach addresses the issue of data availability in the event of NameNode failures. By deploying redundant NameNodes within the HDFS cluster, it aims to ensure continuous data availability and mitigate the risk of data loss. The NameNode Security Enhance (NNSE) system manages these redundant NameNodes, allowing for seamless failover and ensuring that data remains accessible even during NameNode failures. This approach enhances the overall reliability and availability of data within HDFS, contributing to a more robust security posture.

[2] Innovative Secure Authentication Interface for Hadoop Cluster Using DNA Cryptography: A Practical Study

Authors: J. Balaraju and P. V. R. D. Prasada Rao

The paper discusses the significance of handling big data in today's distributed networks and the widespread use of Hadoop for this purpose due to its support for parallel processing and distributed computing. However, it highlights the lack of built-in data security in Hadoop clusters (HC), which poses a risk, especially for sensitive information. Existing security mechanisms rely on third-party providers, leading to computational overhead. To address this, the paper proposes a novel security mechanism called Secure Authentication Interface (SAI) layered over HC. SAI provides user authentication, metadata security, and access control, reducing computational overhead compared to existing mechanisms. The paper includes a comprehensive literature review of related work, outlining various security challenges and proposed solutions in the field. It concludes by highlighting the importance of SAI in creating a trusted environment within Hadoop clusters and suggests future research directions for securing both data and cluster management.

[3] Dynamic Node Identification Management in Hadoop Cluster Using DNA

Authors: J. Balaraju and P. V. R. D. Prasada Rao

The paper discusses Distributed Computing (DC) and its application in bioinformatics, particularly in DNA sequence analysis. It proposes a Distributed Bioinformatics Computing System to analyze DNA sequences efficiently across multiple computers. Additionally, it introduces Hadoop Cluster (HC) as a distributed processing framework and Dynamic Host Configuration Protocol (DHCP) for managing IP addresses within a network. The paper addresses the problem of dynamic node identification management in HC and presents a solution using a security layer implemented within DHCP. This security layer utilizes DNA cryptography to generate unique keys for each node, enhancing the security of the distributed system. The proposed method aims to prevent unauthorized access and ensure data security in distributed computing environments.

Author(s)	Strategies	Advantages	Disadvantages
Saraladevi, B., Pazhaniraja, N., Paul, P.V., Basha, M.S.S. and Dhavachelvan.	Kerberos Approach, Bull Eye Algorithm Approach, Name node Approach.	Secure Authentication, 360-Degree View, Redundancy for Availability.	Complexity, Resource Intensive, Resource Overhead.
Balaraju.J., PVRD Prasada Rao.	DNA Algorithm, BABA.	24x7 Security, Reduced Dependence on External Data Centers.	Performance Overhead, Complexity, Security issues.
Balaraju.J., PVRD Prasada Rao.	Secure Authentication Interface (SAI), DNA Cryptography.	Improved Performance, Reduced Overhead	Complex Implementation, Limited Compatibility.
Balaraju.J., PVRD Prasada Rao.	Integration of HDFS with Secure-Node.	Enhanced Security, Reduced Management Overhead.	Single Point of Failure, Resource Utilization.

5. PROPOSED METHOD

5.1 Motivation for the Proposed System

A precise Hadoop RBAC manager holds immense importance for various critical reasons. Initially, concerning security, RBAC plays a central role in upholding data security within the Hadoop cluster. Errors or vulnerabilities in the RBAC setup might lead to unauthorized entry into sensitive data, potentially causing data breaches and undermining system integrity.

Moreover, adherence to industry regulations serves as a significant catalyst for maintaining high precision in RBAC management. Numerous sectors, including healthcare (HIPAA) and finance (GDPR), impose stringent regulatory frameworks regarding data access and privacy. An accurately functioning RBAC manager is indispensable for ensuring compliance with these regulations and averting penalties.Furthermore, alongside accuracy, the RBAC manager should exhibit efficiency in handling access control lists (ACLs) to prevent performance hindrances, particularly in expansive Hadoop clusters.

5.2 PROPOSED METHOD:

- The proposed method aims to address the challenge of implementing a secure interface in Hadoop, incorporating Role-Based Access Control (RBAC) and file-level permissions within the Hadoop Distributed File System (HDFS).
- Users can upload files to the Hadoop Distributed File System (HDFS) using standard upload methods or tools. When a file is uploaded, the system records the user who uploaded it as the original owner.
- Only the original owner of a file has permission to delete it. Other users, including intruders, do not have permission to delete the file.
- If the original owner wants to grant deletion permission to another user, they can do so through the system. Once permission is granted, the new user can delete the file.
- Similarly, only the original owner has permission to view or read the file. Other users cannot access the file unless the owner grants them permission.

- If an unauthorized user or intruder tries to access the file without permission, the system presents fake or false data instead of the actual file content. This ensures that unauthorized access attempts are thwarted.
- If the original owner grants viewing permission to another user, that user can then view or read the file. The system ensures that permissions are enforced correctly.
- A log file is generated by the system to track all actions performed on files in HDFS. This log includes information such as file uploads, deletions, permission grants, and access attempts. The log helps in monitoring and auditing file operations in the system.

5.3 System Architecture:

Initial Setup and Configurations:

1. Java and SSH Configuration: This includes installing Java and configuring SSH, both vital components for Hadoop's functioning. Java serves as the programming language for Hadoop, while SSH enables secure communication between cluster nodes

2. Hadoop Installation and Environment Setup: This step involves downloading and configuring Hadoop, including setting up environment variables like JAVA_HOME and HADOOP_HOME. These variables are essential for Hadoop to locate Java and for system commands to recognize Hadoop binaries.

3. HDFS Configuration: Here, properties of HDFS such as the default filesystem URL, replication factor, and directories for NameNode and DataNode are configured. This step determines how data will be stored across the cluster.

4. SSH Key Generation and Configuration: Generating SSH keys and configuring SSH to bypass strict host checking are preparatory measures. These ensure smooth communication between master and worker nodes without manual intervention.

5.4 HDFS File Operations:

Uploading Files to HDFS:

The function 'upload_to_hdfs' facilitates file uploads to the Hadoop Distributed File System (HDFS) in a controlled manner, ensuring only authorized users can contribute data. It begins by checking for necessary directories within HDFS, creating them if absent, to maintain a structured data organization. After directory verification, the function proceeds to upload the specified file to the designated HDFS location. By strictly validating directories and restricting uploads to authorized users, this function strengthens data governance, mitigates data fragmentation risks, and enhances security.

Moreover, it supports uploads from local sources, enhancing flexibility and accessibility. This capability streamlines data ingestion, promoting collaboration and data sharing across distributed teams. Additionally, it improves data accessibility, enabling contributions from remote personnel regardless of network connectivity.

Deleting files from HDFS:

The 'delete_from_hdfs' function follows RBAC principles to reinforce data governance and security within HDFS. It verifies file ownership against the user before permitting deletion, preventing unauthorized data removal attempts and ensuring data integrity. By integrating RBAC principles, this function enhances the overall security framework of Hadoop, promoting accountability and transparency in data management.

Downloading Files from HDFS:

The download function in HDFS employs stringent access control measures, validating file ownership and presenting simulated files to unauthorized users to prevent data breaches discreetly. This approach maintains data security and confidentiality while providing insights into unauthorized access attempts. Overall, these functions play critical roles in enforcing access control policies, upholding data governance standards, and ensuring a secure data environment within HDFS.

6. COMPARITVE STUDY OF EXISTING SYSTEM AND PROPOSED SYSTEM

When contrasting the prototype Hadoop RBAC Manager with existing systems, several pivotal aspects come into focus:

Integration and User-Friendliness: Current Hadoop environments often incorporate integrated security solutions like Apache Ranger or Apache Sentry, which provide comprehensive security management, including RBAC functionalities. In comparison, the prototype, although functional within its designated scope, would require significant development efforts to match the seamless integration and user-friendliness of these established tools.

Security Capabilities: Established security management tools for Hadoop offer a wide array of advanced features, including auditing, policy management, data masking, and encryption. In contrast, the prototype primarily focuses on fundamental file operations with role-based access, lacking the broader security functionalities crucial for enterprise-grade applications.

Scalability and Performance: Existing systems are finely tuned for optimal performance and scalability, specifically engineered to handle the demands of large-scale Hadoop clusters efficiently. The scalability and performance of the prototype in a real-world scenario, particularly within expansive Hadoop environments, remain uncertain and would necessitate thorough testing and optimization.

Compliance and Auditability: Adhering to industry standards and regulations is paramount for ensuring data security. Tools such as Apache Ranger offer detailed auditing capabilities, vital for compliance purposes. Conversely, the prototype lacks auditing functionality, a critical component for enterprise-level applications requiring adherence to regulatory standards.

7. RESULT ANALYSIS

Initial Setup and Configuration

Setup Java Environment

```
[1] #!apt-get install openjdk-8-jdk-headless -qq > /dev/null
    print('Java Version:')
    !java -version

Java Version:
openjdk version "11.0.22" 2024-01-16
OpenJDK Runtime Environment (build 11.0.22+7-post-Ubuntu-0ubuntu222.04.1)
OpenJDK 64-Bit Server VM (build 11.0.22+7-post-Ubuntu-0ubuntu222.04.1, mixed mode, sharing)

[2] # Set JAVA_HOME environment variable for Java 11
    import os
    os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-11-openjdk-amd64"

[3] # Verify Java installation and JAVA_HOME path
    !echo $JAVA_HOME
    !java -version

/usr/lib/jvm/java-11-openjdk-amd64
openjdk version "11.0.22" 2024-01-16
OpenJDK Runtime Environment (build 11.0.22+7-post-Ubuntu-0ubuntu222.04.1)
OpenJDK 64-Bit Server VM (build 11.0.22+7-post-Ubuntu-0ubuntu222.04.1, mixed mode, sharing)
```

Fig. 6.1.1 Installation and Setting up Java

```
!apt-get install -qq -o=Dpkg::Use-Pty=0 openssh-client openssh-server > /dev/null
!ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
!cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
!chmod 0600 ~/.ssh/authorized_keys

!echo "Host *" > ~/.ssh/config
!echo "  StrictHostKeyChecking no" >> ~/.ssh/config
!echo "  UserKnownHostsFile=/dev/null" >> ~/.ssh/config
!chmod 400 ~/.ssh/config
```

```
[5] !apt-get install -qq openssh-server
    !service ssh start

 * Starting OpenBSD Secure Shell server sshd
   ...done.

!ssh

usage: ssh [-46AaCfGgKkMNnqsTtVvXxYy] [-B bind_interface]
           [-b bind_address] [-c cipher_spec] [-D [bind_address:]port]
           [-E log_file] [-e escape_char] [-F configfile] [-I pkcs11]
           [-i identity_file] [-J [user@]host[:port]] [-L address]
           [-l login_name] [-m mac_spec] [-O ctl_cmd] [-o option] [-p port]
           [-Q query_option] [-R address] [-S ctl_path] [-W host:port]
           [-w local_tun[:remote_tun]] destination [command [argument ...]]
```

Fig. 6.1.2 SSH Configuration

6.2 Hadoop Installation

Download and Unpack Hadoop

```
[7]  !wget -q https://archive.apache.org/dist/hadoop/common/hadoop-3.2.1/hadoop-3.2.1.tar.gz
     !tar -xzf hadoop-3.2.1.tar.gz
     !mv hadoop-3.2.1 /usr/local/hadoop
```

```
[8]  os.environ["HADOOP_HOME"] = "/usr/local/hadoop"
     os.environ["PATH"] += ":/usr/local/hadoop/bin:/usr/local/hadoop/sbin"
```

```
[9]  # Update hadoop-env.sh with JAVA_HOME for Java 11
     !echo "export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64" > /usr/local/hadoop/etc/hadoop/hadoop-env.sh
```

```
[10] import os
     os.environ["HDFS_NAMENODE_USER"] = "root"
     os.environ["HDFS_DATANODE_USER"] = "root"
     os.environ["HDFS_SECONDARYNAMENODE_USER"] = "root"
```

```
[18] !/usr/local/hadoop/sbin/stop-dfs.sh
     !/usr/local/hadoop/sbin/start-dfs.sh
```

```
Stopping namenodes on [localhost]
localhost: Warning: Permanently added 'localhost' (ED25519) to the list of known hosts.
Stopping datanodes
localhost: Warning: Permanently added 'localhost' (ED25519) to the list of known hosts.
Stopping secondary namenodes [1ff5adc6ae64]
1ff5adc6ae64: Warning: Permanently added '1ff5adc6ae64' (ED25519) to the list of known hosts.
Starting namenodes on [localhost]
localhost: Warning: Permanently added 'localhost' (ED25519) to the list of known hosts.
Starting datanodes
localhost: Warning: Permanently added 'localhost' (ED25519) to the list of known hosts.
Starting secondary namenodes [1ff5adc6ae64]
1ff5adc6ae64: Warning: Permanently added '1ff5adc6ae64' (ED25519) to the list of known hosts.
```

```
!jps

# Check what all nodes are running: Should list SecondaryNameNode, NameNode, DataNode, Jps
```

```
3412 NameNode
3925 Jps
3736 SecondaryNameNode
3533 DataNode
```

Fig. 6.2.1 Install and Unpack Hadoop

6.3 Helper Function Outputs

```
!hdfs dfs -ls /
```

Fig 6.3.1 Before uploading files

```
upload_to_hdfs('/content/sample_data/mnist_test.csv', '/mnist_test.csv', 'Samhitha')
```

```
INFO:HadoopRBACManager:Operation: Upload, Message: from: /content/sample_data/mnist_test.csv to /mnist_test.csv by user Samhitha
Successfully uploaded /content/sample_data/mnist_test.csv to /mnist_test.csv
```

Fig 6.3.2 User uploading file

```
upload_to_hdfs('/content/sample_data/california_housing_test.csv', '/california_housing_test.csv', 'Amitha')
```

```
INFO:HadoopRBACManager:Operation: Upload, Message: from: /content/sample_data/california_housing_test.csv to /california_housing_test.csv by user Amitha
Successfully uploaded /content/sample_data/california_housing_test.csv to /california_housing_test.csv
```

Fig 6.3.3 User uploading file

```
# Example usage:
upload_file_from_local_to_hdfs(hdfs_username='Rakshitha')
```

Choose Files No file chosen Cancel upload

Fig 6.3.4 Uploading local file to hdfs

```
# Example usage:
upload_file_from_local_to_hdfs(hdfs_username='Rakshitha')
```

Choose Files Memos.pdf

- **Memos.pdf**(application/pdf) - 1936882 bytes, last modified: 2/11/2024 - 100% done

```
INFO:HadoopRBACManager:Operation: Upload, Message: from: Memos.pdf to /Memos.pdf by user Rakshitha
Saving Memos.pdf to Memos.pdf
Memos.pdf /Memos.pdf
Successfully uploaded Memos.pdf to /Memos.pdf
```

Fig 6.3.5 Local file uploaded

```
!hdfs dfs -ls /

Found 3 items
-rw-r--r--   1 Nandini  supergroup       1697 2024-04-12 08:19 /anscombe.json
-rw-r--r--   1 Amitha   supergroup     301141 2024-04-12 08:19 /california_housing_test.csv
-rw-r--r--   1 Samhitha supergroup   18289443 2024-04-12 08:19 /mnist_test.csv
```

Fig 6.3.6 List of files inside hdfs after uploading

```
delete_from_hdfs('/Memos.pdf', 'Amitha')

INFO:HadoopRBACManager:Operation: Delete, Message: file: /Memos.pdf owner: Rakshitha user: Amitha
INFO:HadoopRBACManager:Operation: Delete, Message: User Amitha does not have permission to delete /Memos.pdf. Owner is Rakshitha.
User Amitha does not have permission to delete /Memos.pdf. Owner is Rakshitha.
```

Fig 6.3.7 Intruder do not have permission to delete the file

```
[37] grant_file_permission(hdfs_path='/Memos.pdf', owner='Rakshitha', user='Amitha', download_perm=True, delete_perm=True)

INFO:HadoopRBACManager:Operation: GrantPermission, Message: Permissions granted for /Memos.pdf to Amitha: download=True, delete=True by Rakshitha

delete_from_hdfs('/Memos.pdf', 'Amitha')

INFO:HadoopRBACManager:Operation: Delete, Message: file: /Memos.pdf owner: Rakshitha user: Amitha
INFO:HadoopRBACManager:Operation: Delete, Message: Successfully deleted /Memos.pdf
Successfully deleted /Memos.pdf
```

Fig 6.3.8 owner grants permission to the user to delete the file

```
# Example usage
download_hdfs_file('/california_housing_test.csv', 'Amitha')

INFO:HadoopRBACManager:Operation: Download, Message: file: /california_housing_test.csv owner: Amitha user: Amitha
Downloading the file...
Downloading "20240412082313_california_housing_test.csv":
```

Fig 6.3.9 original user reading the data

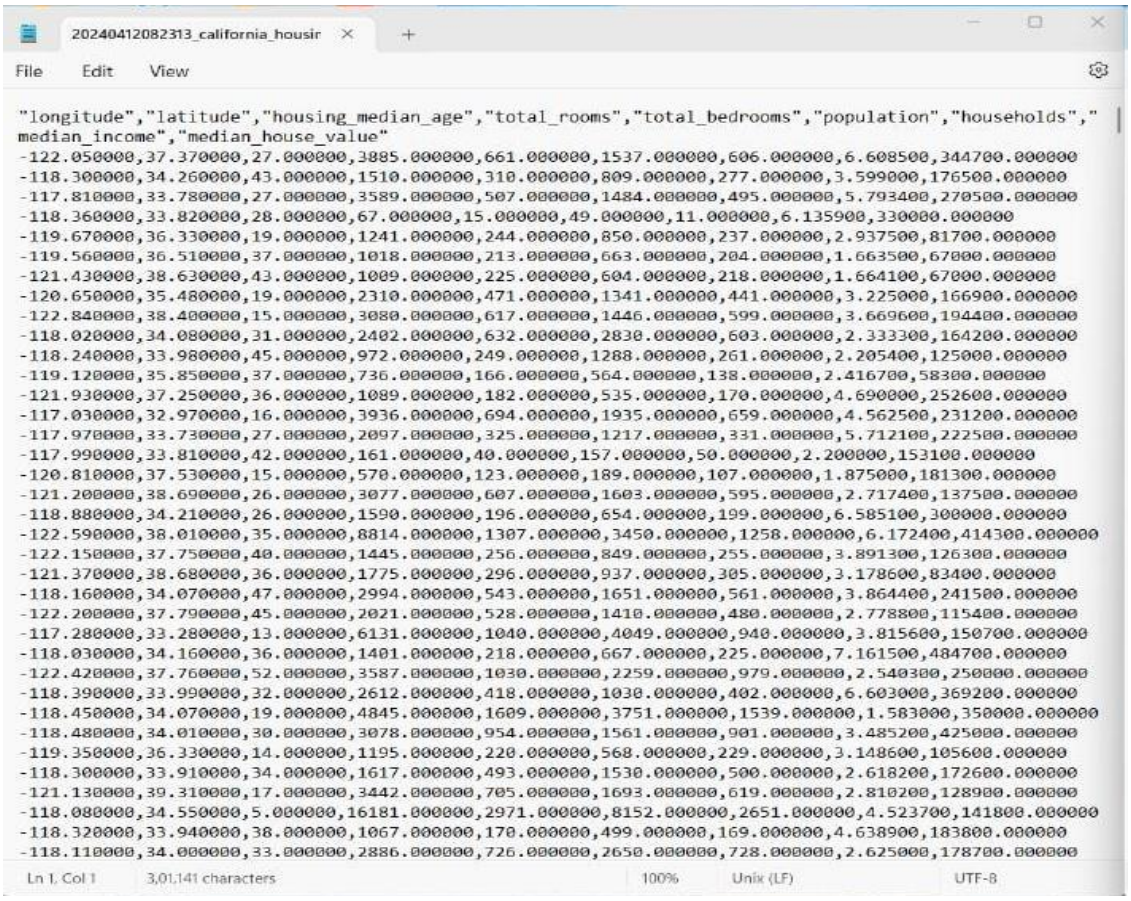

Fig 6.3.10 Original data

Fig 6.3.11 Intruder trying to read the data

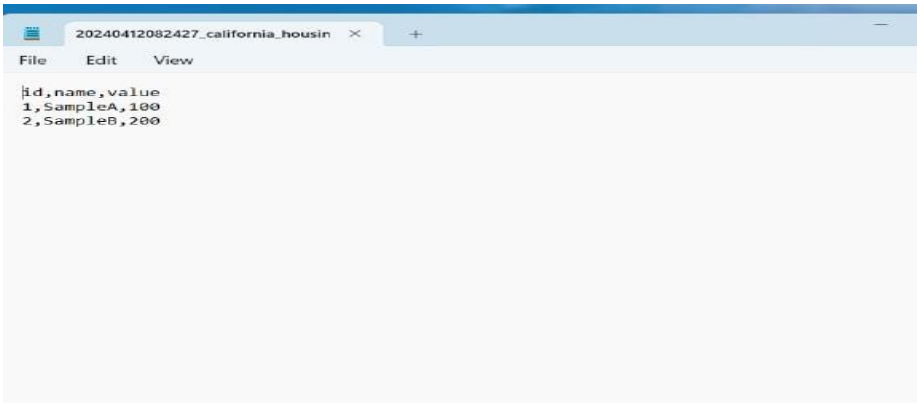

Fig 6.3.12 False data

Fig 6.3.13 Granting permission to the other user to read the files

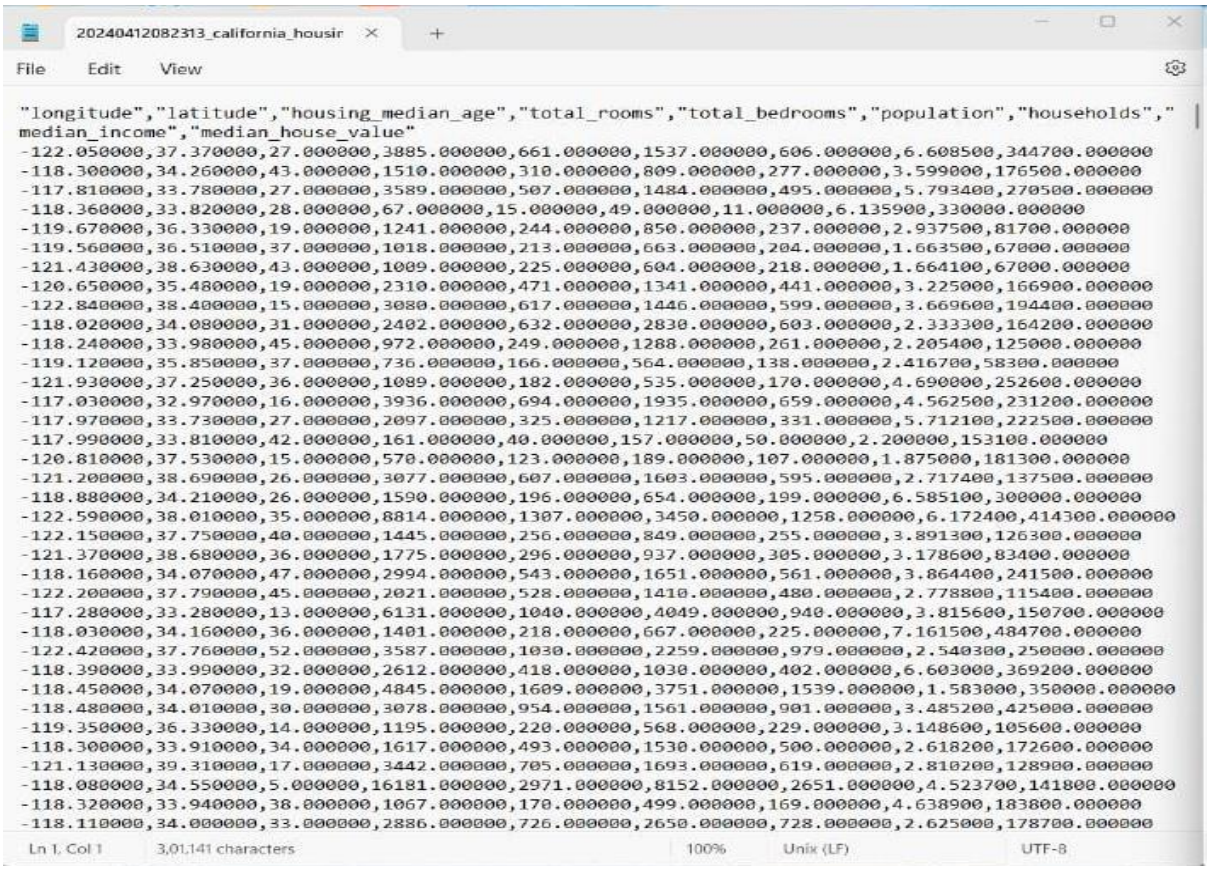

Fig 6.3.14 Original data

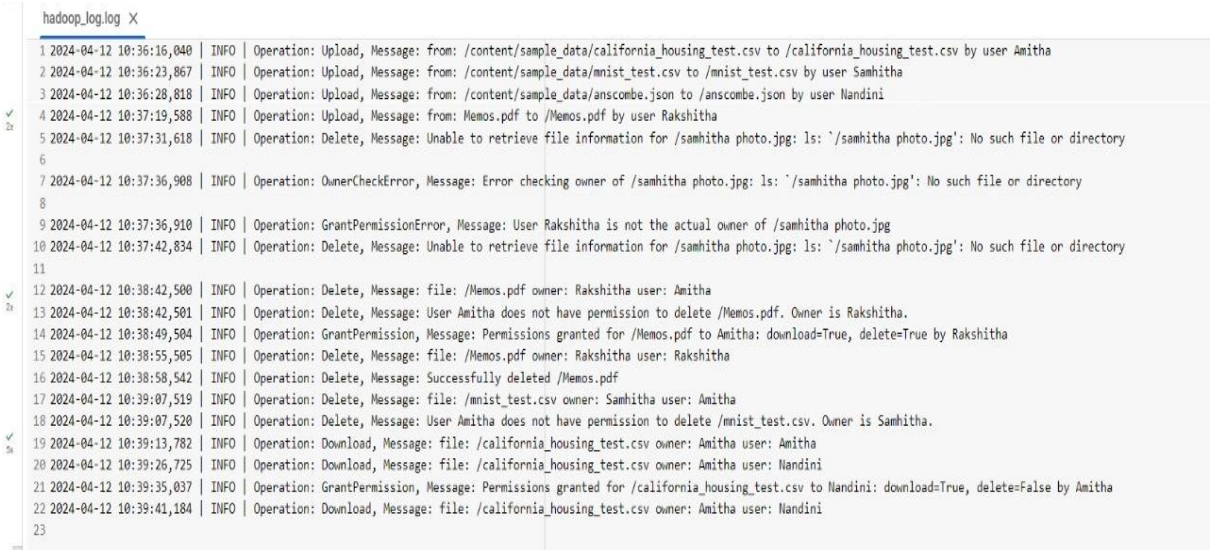

Fig 6.3.15 Logged file

The analysis of results encompasses key performance indicators such as file upload time, file deletion security, and scalability to evaluate the effectiveness and efficiency of the system. File upload time measures the duration taken to upload files onto the platform, indicating the system's responsiveness and user experience. File deletion security assesses the robustness of the system's security measures in ensuring safe and reliable removal of files, safeguarding sensitive data from unauthorized access or deletion. Scalability refers to the system's ability to handle increasing loads and user demands, demonstrating its capacity to accommodate growth and maintain optimal performance levels under varying usage conditions. Together, these metrics provide valuable insights into the system's performance and suitability for real-world deployment.

8. CONCLUSION AND FUTURE ENHANCEMENTS

Future enhancements to the Hadoop RBAC Manager should prioritize a comprehensive approach that integrates enhanced security measures, fine-grained access controls, efficient user and group management, policy compliance, scalability, usability improvements, integration, extensibility, robust documentation, and community engagement. Firstly, bolstering security through Kerberos integration and encryption protocols would fortify data protection both at rest and in transit within Hadoop clusters. Secondly, implementing Attribute-Based Access Control (ABAC) alongside detailed audit trails would offer nuanced access controls and comprehensive oversight over user actions, ensuring compliance and aiding forensic analysis. Thirdly, streamlining user and group management via LDAP integration and role delegation would align RBAC with organizational structures, fostering distributed management and reducing administrative burdens. Fourthly, centralizing policy management and ensuring compliance with regulatory frameworks would enhance governance and mitigate risks. Additionally, optimizing scalability and performance through cached access permissions and distributed administration would facilitate seamless operations as the system scales. Usability enhancements, such as a user-friendly GUI and policy simulation tools, would enhance accessibility and user experience. Integration and extensibility via robust APIs and plugin architectures would enable seamless interaction with external systems and accommodate future requirements. Finally, comprehensive documentation and community engagement initiatives would empower users, foster collaboration, and drive continuous improvement. Balancing these recommendations requires a nuanced approach that prioritizes security, usability, and performance, ensuring that the RBAC Manager evolves to meet the dynamic needs of users and organizations without compromising accessibility or efficiency.

The summary, conclusion, and recommendation for the project "Unveiling Unauthorized Access in Hadoop Distributed File Systems" encapsulate the key findings and insights gleaned from the research endeavor. The project aimed to address the critical challenge of managing access permissions for files stored on Hadoop Distributed File Systems (HDFS), with a focus on enhancing security and compliance through Role-Based Access Control (RBAC) principles. In summary, the project successfully developed a system that leverages RBAC to manage access permissions for files on HDFS effectively. By assigning roles or permissions to users, the system

ensures that only authorized individuals can perform specific actions such as reading, writing, or deleting files. This granular control over access rights enhances security within Hadoop environments, mitigating the risk of unauthorized access and potential data breaches. Moreover, the implementation of RBAC streamlines access management processes, leading to improved data management efficiency and compliance with regulatory requirements. In conclusion, the project underscores the importance of robust access control mechanisms in safeguarding sensitive data stored on Hadoop Distributed File Systems. By adopting RBAC principles, organizations can establish a hierarchical system of permissions that aligns with their security policies and regulatory obligations. The successful development and implementation of the access management system demonstrate its efficacy in enhancing security, compliance, and data management efficiency within Hadoop environments. Moving forward, it is recommended that organizations continue to prioritize access control measures and invest in technologies that bolster the security of their Hadoop environments. Additionally, ongoing monitoring and evaluation of access permissions should be conducted to identify and address any potential vulnerabilities or compliance gaps. Furthermore, regular training and awareness programs can help educate users about the importance of adhering to access control policies and best practices, fostering a culture of security awareness and accountability across the organization. By taking a proactive approach to access management, organizations can strengthen their defenses against unauthorized access and safeguard their critical data assets effectively.

9. REFERENCES

[1] Saraladevi, B., Pazhaniraja, N., Paul, P.V., Basha, M.S.S. and Dhavachelvan, P. (2018) Big Data Security Challenges: Hadoop Perspective. International Journal of Pure and Applied Mathematics, 120, 11767-11784.

[2] Balaraju J., PVRD. Prasada Rao (2020) Investigation and Finding A DNA Cryptography layer for Securing data in Hadoop Cluster

[3] J. Balaraju and P. V. R. D. Prasada Rao (2020) Innovative Secure Authentication Interface for Hadoop Cluster Using DNA Cryptography: A Practical Study

[4] J. Balaraju and P. V. R. D. Prasada Rao (2021) Dynamic Node Identification Management in Hadoop Cluster Using DNA

[5] Balaraju.J., PVRD. Prasada Rao (2019) Designing Authentication for Hadoop Cluster using DNA Algorithm

[6] Yang, H., Lee, J.: Secure distributed computing with straggling servers using polynomial codes. IEEE Trans. Inf. Forensics Secur. 14(1), 141–150 (2019). https://doi.org/10.1109/TIFS.2018. 2846601

[7] Khanan, A., Abdullah, S., Mohamed, A.H.H.M., Mehmood, A., Ariffin, K.A.Z.: Big data security and privacy concerns: a review. In: Al-Masri, A., Curran, K. (eds.) Smart technologies and innovation for a sustainable future. Advances in Science, Technology & Innovation (IEREK Interdisciplinary Series for Sustainable Development). Springer, Cham (2019). https://doi.org/ 10.1007/978-3-030-01659-3_8

[8] Roy, M., et al.: Data security techniques based on DNA encryption. In: Chakraborty, M., Chakrabarti, S., Balas, V. (eds.) Proceedings of International Ethical Hacking Conference 2019. eHaCON 2019. Advances in Intelligent Systems and Computing, vol. 1065. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-0361-0_19

[9] Samet, R., Aydın, A., Toy, F.: Big data security problem based on Hadoop framework. In: 2019 4th International Conference on Computer Science and Engineering (UBMK), Samsun, Turkey, pp. 1–6 (2019). https://doi.org/10.1109/UBMK.2019.8907074