Distributed File System

Dhruv Patel
dhruvjp1@umbc.edu

Payodhi Mandloi
payodhi.m@umbc.edu

Dharani Madireddy
dmadire1@umbc.edu

Surya Kiran CH
RU17699@umbc.edu

SSSD SAICHAND CH
schebol1@umbc.edu

AIM:

The aim of this project is to build a Distributed File System with security features. The code was implemented in two phases

Phase 1: Developed a Simple Distributed File System

Phase 2: Implemented the security functionality of the Distributed File System

ARCHITECTURE:

We have designed a Distributed File System with three-level architecture:

Level 1: All Clients (Authentic/Random users)

Level 2: Master Node and Scheduler

Level 3: Three Server Node.

Level 1: Client

The client is connected to only the master (semi-server) node. The client in the system needs to authenticate themselves in order to access the functionality provided by the server. The client implements the encryption on the data (like the content of the file, file name, and directory name), and validates themselves in order to access the server at the master node.

Level 2: Master and Scheduler

The master node is responsible for establishing the connection between the client and the servers. This acts as an intermediate node between the client and the replica servers. The task of the master node is to provide a link between the client and the server. Before establishing the connection, it will validate the client. The master node will also handle which client will access which file as per the permission provided by the owner. The scheduler will keep track of the files in the directory if any file is physically deleted from the server, it will show the pop-up message.

Level 3: Servers

All the client requests are processed by the servers that are directed by the master node. The response of each operation is again sent to the master server to be communicated back to the client. The server will have the encrypted data of the client and will not be able to get any information from it.

The master node is an important part of our architecture which will have the record of the actions made by the client and records of all permission and validation data.

METHODOLOGY:

- The Distributed file system was implemented in Python and in order to implement the security features we have used various encryption packages available in Python
- The connection between the client and server is done using the socket programming that is implemented using the **socket** package
- Once the connection between the client and the master(server) node is established, the user needs to authenticate themselves to access the functionality of the system
- The system supports the following functionalities:
    - Create Directory
    - Create File
    - Read File
    - Write File
    - Delete File
    - Rename File
    - Get the list of files
    - Change Permission

- The system supports data replication across all the servers regardless of the function being performed. We have implemented 3 replica servers in our architecture
- All the files created by authorized users are given "owner" rights and by default, all the other users in the system are given null privileges. When a user wants to update the permission, they can make use of the "Change Permission" option to update the file permission for a specific user
- Content of the file are hidden from the servers which is achieved by using the **pyaes** package
- The name of the file and directory are treated as confidential and are not readable by any of the servers, this functionality is also achieved by using **pyaes** package
- All the operations performed by the user in the system are logged using the **logging** package. If a user tries to access the files they aren't authorized to, the system will return an appropriate error message
- We have implemented a scheduler that would keep track of all the files and directories of the system, if a malicious server tries to delete a file then it can be detected by the scheduler and the replica servers can be used to restore the most recent state of the system

HOW TO IMPLEMENT IN YOUR SYSTEM:

Requirements:

Software:

Python IDE(Any): Jupyter notebook, spyder, PyCharm, Atom, and many more with python setup. You can run on IDE or may also run on cmd or anaconda prompt

Microsoft Excel: To access the excel sheets

Libraries:

Socket: To provide the socket values like IP address and port number to create no overlapping connection between clients and server

Socketserver: To implement a server-client environment (the connection between server and client)

ThreadingTCPServer: To implement the multithreading for multiple client connections at the same time. Also, to implement thread-locking for no overlapping write or edit operation

OS and SYS: OS to get access to the directory and file path to implement all the changes

Pandas: To access and update the excel sheet for permission and validation of clients

Pickle and Jason: These libraries are used as the temporary storage unit while sending or fetching the data to the server or from the server.

Logging: To configure the logger file to keep track of all the changes made by the client on the server

Traceback: To print or retrieve the stack created to keep track of any update made by the client in the logger text file

Base64: This library is used to implement encryption on data at the client-side to keep data secure at the server end

Pyaes, pbkdf2, binascii and secret: pyaes to implement AES encryption method, pbkdf2 is a library to implement simple cryptography key derivation function, binascii to convert between binary and ASCII and secret to generate secure random numbers for managing secrets

Tkinter: To implement the pop-up notification which will indicate whether any file is deleted physically or not

Time: To implement the time loop for executing the scheduler function to search for a malicious attack

Getpass: To implement the hide input feature for passwords

If you lack any single library from above, then install it before performing or implementing this project.

SET UP THIS PROJECT:

Step 1: Install any IDE in which you want to perform this project. (Jupiter is more recommended). Here is the link to download Jupyter lab: *https://jupyter.org/*

Step 2: Install all the libraries mentioned above which are not installed in your system and also download their dependencies

The simple way to install these libraries in your system is to write "**pip install _____(library name)"** on the anaconda prompt or cmd

Note: Change the absolute Path in all server files and the scheduler file to the location of the files directory of each server in your system

Step 3: Clone this project from the Github repository in your system

Step 4: Open the project in your IDE or Jupyter notebook and run all three server files. You will get the IP address of that server. Copy that server IP address and paste it into the second cell of the master file in the IP array sequentially. Enter the IP address in sequence like: IP = ["Server1 IP", "Server2 IP", "Server3 IP"]. Then run the master node

Step 5: Before running the "client-dhruv.ipynb" file edit the validation excel sheet in a master folder by adding your username and password

Step 6: After making changes in the validation file, run the "scheduler.ipynb" file present in the master folder. This file will execute continuously until you stop executing this file externally

Step 7: Now you can execute the "client-dhruv.ipynb" file. Initially, it will ask for a username and password. Enter the username and password that you have entered in the validation sheet.

Step 8: After that, you will be able to perform all the operations like creating a directory and file, writing a file, updating the file, deleting a file, renaming a file, listing file names, and a few more operations. All data will be stored in an encrypted format at the server end.

Step 9: If you want to see the log of the operations performed by the client. You will find the logger text file in the master folder.

Step 10: When you delete any file physically from any of the servers. The pop-up will be displayed automatically and indicates the file which is deleted and the server from which it is deleted.

MEMBERS CONTRIBUTION:

Dhruv: Constructed the architecture client-master-server and created master file, Integrating individual modules to the client-server architecture, User validation, Scheduler file, Report
Payodhi: Server-Client connection multi-threading, logger**,** Implementation of user permission, Integration of encryption module, Report
Dharani: Implementation of Encryption module, Integrating encryption module at client-side and Implementing CRUD operations modules.
Surya: Implementation of Encryption module, Implementing CRUD operations modules, Report.
Chandu: Implementation of Encryption module, Implementing CRUD operations modules,Report.

The code can be found on: ***https://github.com/Dhruv68/Distributed_File_System***

CONCLUSION:

By developing this Distributed File System, we learned the implementation of networking and how to enable security features on a file system. We implemented the complete distributed file system with the implementation of encryption. We also implemented the logger function which will keep track of all the operations performed by an authenticated user and store it in the logger text file for future reference. We have created a scheduler that will continuously keep track of physical actions performed by any random user at the server end (Malicious detection) and notify the client that the particular file is physically deleted at the particular server.