System Programming – Project Description

Name: Niharika Srivastava

Student ID: 202012004

MSc. IT (Semester - 2)

Project Title – Secure My Media

Goal of The Project –

This project aims to enable the system to smoothly provide and handle multiple facilities of securing and managing media (files and folders) to a user on his/her computer. The system can provide multiple facilities at the same time as per the user's choice by opening the intended number of processes (one to many) in separate child terminals from the parent terminal, such that the outputs of concurrent processes do not get intermingled and are unambiguous. Complex functionalities like encryption and decryption of multiple media are handled by the system efficiently where it manages and removes the un-secure files (permanently deletes them from System Trash as well such that they are in-accessible) generated in the process and also provides extra security by providing the facility of moving encrypted media to a secure folder. The system also enables the user to hide/unhide media effectively and change their permissions effortlessly. A crucial feature of this project is that it automatically generates logs (log file stored in the system's default folder for all log files i.e., /var/log) for every parent and child process, and clearly elaborates when and which media was affected (successful/unsuccessful operations) by which child process. This project extensively uses the concept of shell scripting to provide desired functionalities of media security to the user by hiding background complexities which are handled by the system. In combination with concepts like Process Management, Pipes, File I/O and Signals, it creates a user-friendly environment and enables the system to achieve multiple tasks at the same time. This project thus, comprises multiple system programming concepts and solves the real-world problem of securing media on a small-scale.

Project Requirements (Program-wise) –

1. Program Name: 1_secureMyMedia.c

- This C Program is the main program which has the potential to execute every shell scripting file as per the user's requirements.
- Five functionalities are provided to the user where each functionality is provided by a separate shell script. Five child processes are created through appropriate nesting and their exit status are captured by the parent to prevent them from becoming orphan processes.
- The user's selection code fragment is prevented from CNTRL+C Attack by ignoring the interrupt signal.
- Once the user has selected all the functionalities that he/she wants, a separate child terminal will be opened by each child process where the corresponding shell script will execute for providing the desired functionality. Thus, One to Five Child Terminals can open and execute their instructions simultaneously by using the gnome-terminal command.
- Every chosen child process will send an acknowledgement of executing its functionality to the parent process by using unidirectional pipes.
- File I/O (Buffered I/O) is used to write log entries like 'Session Opened' when this program is executed and 'Positive Acknowledgement' from the selected child processes when their child terminals are opened.
- The log entry's prefix is created in this format: Timestamp UserName (UserID). Every Log statement is appended to this prefix after a colon (:)

2. Program Name: 2_protectFileFolder.sh

- This program intends to provide password protection and symmetric encryption to a single file or folder inside a directory path.
- The user needs to specify the directory path where the media requiring encryption is located.
- If the path exists, all the files and folders located inside the path will be shown from which the user can enter the name of the file/folder which he/she wishes to encrypt.

- If the user selects a file for encryption, the protectFile() function will be called where gpg command is executed. The gpg command stands for gnu private guard which is an inbuilt command provided by the Linux operating system for encrypting and decrypting media. The -c switch specifies that a symmetric encryption is being performed where both the keys (passwords) for encryption and decryption are same.
- If the encryption is successful, the extension '.gpg' is appended to the filename and the un-encrypted file is explicitly removed from the current folder as well as the System Trash because the gpg command does not remove the original file. Logs are generated (with clear specification of Child Process Number i.e., 1, the path and the name of the encrypted file) and redirected to the existing log file. If the encryption is unsuccessful appropriate logs are stored for the same.
- The user is further provided the facility of moving the encrypted file to a Secure Folder. If the user agrees, the moveToSecureFolder() function is called from within the protectFile() function. Access permissions from the encrypted target is removed to make it more secure and the encrypted target is moved from the current path to the Secure Folder using the mv command.
- If the user selects a directory for encryption, the protectDir() function is called which converts the directory to a file by compressing it using the tar command. It is necessary to convert the directory to a file as the gpg command only works for files. This tar file is now passed as a file target to the protectFile() function and the above steps are executed.

3. Program Name: 3_protectMultipleMedia.sh

- This program intends to provide password protection and symmetric encryption to multiple files and folder inside a directory path.
- The user needs to specify the directory path where all the media requiring encryption are located.
- If the path exists, all the files and folders located inside the path will be shown from which the user can enter the names of all the files and folders which he/she wishes to encrypt.
- Once all selections are made, the protectMedia() function is called. It asks the user to provide a directory name in which all the above Targets will be moved implicitly and this directory will be compressed to create a single tar file.
- This tar file will be further encrypted by the gpg command and it can be moved to the Secure Folder as well. Appropriate logs as per successful or unsuccessful encryption will be created and redirected to the log file with the Child Process Number i.e., 2 and the affected file target name.

4. Program Name: 4_decryptingMultipleMedia.sh

- This program intends to decrypt one to multiple files and folders inside a directory path which have been previously encrypted using the gpg command.
- The user needs to specify the directory path where all the encrypted media requiring decryption are located.
- If the directory path is same as the Secure Folder's path, the moveBackFromSecureFolder() function is called which asks the user to specify a new directory path where all the desired encrypted media from Secure Folder will be moved and only then the decryption process will be initiated. The selected media's permission is changed again to the default access permission as they were previously revoked when the encrypted media was moved to the secure folder. The decryption process is not performed in the secure folder as it violates the concept of fundamental privacy guaranteed by this folder.
- If the directory path is different than the secure folder's path, the decryption process will get initiated directly.
- All the files and folders located inside the path will be shown from which the user can enter the names of all the files and folders which he/she wishes to decrypt.
- The file targets are stored separately and will be passed to the decryptFiles() function which will decrypt every single file (using gpg command without any switch) and remove the encrypted file from the current folder as well as the system trash. As per successful/unsuccessful decryption, corresponding logs containing the Child Process Number i.e., 3 and the affected file target name will be redirected to the log file.
- The directory targets are stored separately and will be passed to the decryptDirectories() function which will first decrypt every tar file, extract all its contents and then the encrypted file and the tar file will be removed from the current folder as well as the system trash. Corresponding logs are stored for the same.

5. Program Name: 5_changeMediaPermission.sh

- This program intends to change the access permissions for one to multiple files and folder inside a directory path as per the user’s choice.
- The user needs to specify the directory path where all the media requiring permission change are located.
- If the path exists, all the files and folders located inside the path will be shown from which the user can enter the names of all the files and folders which he/she wishes to change the permissions for.
- Once the user selects a file/folder, its existing permission will be shown to the user and the user can then select the new permission (for all User, Group and Other users) efficiently by specifying 1 or 0 for read, write and execute access.
- The binary selections entered by the user will be converted to a decimal value which will be passed to the chmod command and it will change the permissions efficiently. The same process can be performed for multiple media and the corresponding logs for Child Process Number i.e., 4 and the affected file/folder target name with the old and new permission will be redirected to the log file.

6. Program Name: 6_hideUnhideMedia.sh

- This program intends to change the visibility (hidden or unhidden) for one to multiple files and folder inside a directory path as per the user’s choice.
- The user needs to specify the directory path where all the media requiring visibility change are located.
- If the path exists, all the files and folders including the hidden one (ls -a switch) located inside the path will be shown from which the user can enter the names of all the files and folders which he/she wishes to change the permissions for.
- The user would have to choose either Hide (choice=1) or Unhide (choice=2) and the corresponding functions hideMedia() and unhideMedia() will be called.
- If the user has chosen option 1 for hiding media, the target’s name will be prefixed with a dot(.) and will be renamed using the mv command. All hidden media are prefixed by dot in Linux operating system. Corresponding logs with Child Process Number i.e., 5 and the affected file/folder target name will be redirected to the log file.
- If the user has chosen option 2 for unhiding media, the target’s name prefix of dot(.) will be removed by pattern substitution and mv command will be used to rename the same. Corresponding logs are stored for the same.

Project Assumptions –

- Each shell scripts functionality is directory specific (directory path entered by the user) and works on all the media located in that particular directory. The Secure Folder is the only directory which is not a part of the directory path specified by the user (unless directory path is specified to be same as the Secure Folder path) and is still accessed by the shell script on user demand.
- The gpg command is updated and supported in the Linux operating system.
- All five Child processes are created even though their functionality might not be selected by the user. If they are not selected, they immediately exit. Otherwise, they open a new child terminal and execute their corresponding shell scripts.
- exec command is not used and instead of it, gnome-terminal command has been used. The exec command displays the output of every child process in the same terminal and due to multiple concurrently executing child processes, the outputs and inputs for all the child processes can get intermingled and create confusion.
- The gnome-terminal command and its switches are not deprecated, and are updated and supported in the Linux Operating System.
- The /var/log folder has been granted the access permission for storing our custom log file inside it.
- The shell files have been granted execute permission such that they can be executed by the C program.
- The log file is accessed and written to by the C program as well as all the shell script files.

Note – The screenshots of all functionalities provided by the project are located inside the Output folder.*