CS419 System Security
CS 419 System Security Project
Please Run Using Python3

SYSTEM SECURITY
• A protected file system
• For a given folder and files inside, the system only allows the account Alice to use certain programs to create/read/edit/delete it
• You need to assign correct permissions
• Other accounts are not able to read the content
• Purely user level file system is fine

Questions to Answer:
• Please explain how you encrypt and decrypt files
• Please explain how do you deal with random access/edit
• Please explain the trade-off in the design
• Please explain how do you store keys
• Please explain how do you authenticate Alice
• Please explain the provided security features

1) We encrypt files by converting the file into a textfile.gpg by hashing it using MD5 hashing algorithm. After we hash it, we then apply a passphrase, remove the original file and then assign chmod of the file and set it to 700. Doing all of this adds a hashed file of rh encrypted file. Then in order to decrypt the file, we then reverse what we did to the encrypted file. We revert the file by converting it from .gpg back to.txt. After doing this we remove the hashed file and by doing this we then decrypt the file back to its original state.

2) We deal with random access and edit by using script2. We add the add to the md5 hash and the metadata from the os.stat method as well as a user inputted password to create a new hash. With this change, there are new variables that are calculated when creating a new hash in which we will be able to identify if there has been any edits or random access.

3) The trade-off in the design is that crypto can provide very strong controls over data confidentiality, but it is computationally expensive, and becomes difficult to manage when there are many users who need access to the same data and all begin to use it at the same time. Meanwhile, access control is much more flexible and can be easier to implement, especially in cases where complex relationships exist between users and data, but it is only enforced by the code that is running on the system.

4) The way we store the keys are when we create the hashed file using MD5 hashing algorithm, we include the key inside of the hashed file. So when we encrypt, the original file then becomes

a hashed file that includes both the contents and the key itself. When we wish decrypt the file, we are able to grab the key from the hashed file and use it.

5) We authenticate Alice by making the permissions owner only. If someone other than Alice tries to open it, they will not be able to. When we encrypt the file we set the chmod to 700 in order to limit access to the user only and that the root user can not access it.

6) Some of the provided security features are, we create a user- inputted passphrase which helps combat randomaccess/editing. We also include when we encrypt the file we change the chmod to 700 to enable that only the root user can't access it. By including auditing with script2.py we enable that if anyone attempts to change or modify the file without permission it would not be allowed.

file.py is our script which focuses on creating a new file, writing to a file, reading a file, or deleting a file. The way this script works is the user will enter one of the following options; new, read, write, delete, or exit. Based on the option our script will do the corresponding action.

script.py is our main script which is in charge of encrypting and decrypting the requested file. The way this file works is the user imports the file and chooses an option of either encrypting or decrypting the file. Once chosen, if we wish to encrypt then we take the file and using MD5 we hash the file, turning it into a .gpg file. If we wish to decrypt the file, we take the hashed file and revert it back into its original file, turning it back to a .txt.

script2.py is our script which focuses on auditing. It is is a modified version of script.py which creates a user-inputted passphrase. In this script we add to the md5 hash and the metadata from the os.stat method as well as a user inputted password to create a new hash. This helps because os.stat has change time and access time variables that are near impossible to fake, and these times being changed results in new hashes. Therefore, if anything was changed the hash will not equal the one in the file, which will be a red flag that it was viewed by someone that wasn't Alice.