Written by: Lim Chee Yeong
ID : (INTI) J14016414, (UOW) 4933643
CONTACT: me@cylim.info

Table of Content

1. Introduction

The author/student expressed that it is a better way of learning system security, besides studying all those theory, the author have the chances to actual try to built a console program with the theory which lead the author to have a better understanding of related theories with actual skills.

2. Instructions

2.1. Python version

The python version using in this assignment is v2.7(more information: docs.python.org/2.7/). Even though the Python v3.4 is stable release, but most Linux Operating System come with v2.7 by default, that's why in the author chose to code in v2.7 in order to prevent incompatible issues. As there is a few changes which might affect the whole program. For examples, print statement is v3.4 behave as a function print().

2.2. Running the program

As python is higher level programming language, compilation is not needed. But if compilation is part of the requirement, it is totally okay to compile it.

"python py_compile main.py"
"mv main.pyc FileSystem"

There are certain way to run the program, below are the best practices,

without compilation, "python main.py" or "python main.py -i"
after compilation, "chmod +x FileSystem"
"./FileSystem" or "./FileSystem -i"

*with "-i" system argument, will execute the registration() for register new user. Without system argument will go to login() to login to the system.

2.3. File storing method

2.3.1. salt.txt and shadow.txt

This files will be store in ~/etc . The usage of this files are to store username, clearance, salt and hashed password.

2.3.2. Files.store

This file will be hidden file stored inside ~/FileServer . The usage of this file is to track all files which is available in the system and the security level.

2.3.3. Files created with the system

For all files created by the system will be stored at ~/FileServer , if the program didn't save at a session, that particular session files will be destroyed automatically to eliminate wastes of resources.

3. Source Code

3.1. Library - Uses of Library makes programming easier

```
import sys #for direct contact with command line, passing arguments and exit
import getpass #for hiding the password field purpose
import hashlib #library of hashing method, in this case: md5
import random #randomly generate integers
import time #for system to pause a few seconds
import os #for system call
import re #regular expression for password policy
```

3.2. main() - Decide the system to be open

```
def main():
    #when system argument more than 1, perform the checking
    #if the argument in location 1 is '-i' go to register()
    #else terminate program
    if len(sys.argv) >= 2:
        if sys.argv[1] == '-i':
            register()
            sys.exit()
        else:
            print "Invalid arguments, SYSTEM TERMINATED !"
            sys.exit()
    #if not special argument, go to login()
    login()
```

3.3. register() - function to register new user

```
def register():
    saltFile = open("etc/salt.txt", "r+")
    saltUser = []
    #read the file without '\n' and separate by ':'
    for line in saltFile:
        saltUser.append(line.strip().split(':'))

    salt = random.randint(10000000, 99999999) #generate random number
    username = raw_input("Username: ")
    rows = len(saltUser)#find the length of saltUser Array
    #for each row in the total rows of saltUser,
    #perform checking whether the username is mateched
    #if matched, terminated the program
    for row in xrange(rows):
        if saltUser[row][0] == username:
            print "User is available in the system!"
            sys.exit()
```

```
#get password from user input
#if password not matched 3times, terminate the program
#Password policy: atleast 6characters with one lower, upper and digit,
#id and password can't be the same
count = 0
while count < 4:
        password = getpass.getpass() #password field
        confirmPassword = getpass.getpass("Confirm Password: ")
        if len(password) < 6:
                print "Password must be atleast 6 characters."
        elif not re.search('[a-z]', password):
                print "Password must have atleast 1 LOWERCASE characters."
        elif not re.search('[A-Z]', password):
                print "Password must have atleast 1 UPPERCASE characters."
        elif not re.search('[0-9]', password):
                print "Password must have atleast 1 DIGIT characters."
        elif password!=confirmPassword:
                print "Password and confirm password not matched."
        elif username == password:
                print "Username and Password can not be same."
        elif confirmPassword == password:
                break
        count += 1
        if count == 3:
                print "Sorry, password incorrect three times, program terminated."
                sys.exit()
        else:
                print "Please try again,"

hash = hashlib.md5(password+ str(salt)).hexdigest() #hash the password and salt
#get user clearance, if not 0,1,2, retry
userClearance = raw_input("User clearance (0 or 1 or 2): ")
while userClearance != '0' and userClearance != '1' and userClearance != '2':
        print "Wrong clearance setting, please try again,"
        userClearance = raw_input("User clearance (0 or 1 or 2): ")
#store into files
saltFile.write(username + ':' + str(salt) + '\n')
saltFile.close()
shadow = open("etc/shadow.txt", "a")
shadow.write(username + ':' + hash + ':' + userClearance + '\n')
shadow.close()
```

3.4. login() - function for user to key in ID and password

```
def login():
    #ask for username and password,
    #then pass the value to verifyPass() for verification
    username = raw_input("Username: ")
    password = getpass.getpass()
    verifyPass(username, password)
```

3.5. verifyPass() - to verify username and password

```
def verifyPass(username, password):
    saltFile = open("etc/salt.txt", "r")
    saltUser = []
    for line in saltFile:
        saltUser.append(line.strip().split(':'))
    rows = len(saltUser) #find the total row the file contain
    for row in xrange(rows):
        if saltUser[row][0] == username:
            salt = saltUser[row][1]
            print saltUser[row][0], "found in system."
            print "Salt retrieved: ", salt
            break
    else:
        print "Username not found!"
        sys.exit()
    saltFile.close()

    print "hashing..."
    time.sleep(0.5) #pause system for 0.5sec
    #hash user enter password and check with hash in shadow.txt
    hash = hashlib.md5(password + salt).hexdigest()
    shadowFile = open("etc/shadow.txt", "r")
    shadow = []
    for line in shadowFile:
        shadow.append(line.strip().split(":"))
    rows = len(shadow) #find the total row the file contain
    for row in xrange(rows):
        if shadow[row][0] == username:
            check = shadow[row][1]
            userClearance = shadow[row][2]
    #check whether the user enter password same as the hash value
    #if same, show the username and clearance.
    #else terminate the system
    if check == hash:
```

```
            print "hash value:", hash
            time.sleep(0.5) #pause system for 0.5sec
            print "Authentication for user", username, "complete."
            print "The clearance for", username, "is", userClearance
    else:
            print "Password incorrect!"
            sys.exit()
    #pass the userClearance to Filesystem menu,
    #in order to examine the permission
    fileSystem(userClearance)
```

3.6. fileSystem() - handling all function about File management

```
def fileSystem(userClearance):
    #open file with read permission only
    fileList = open("FileServer/.Files.store", "r")
    list = []
    #write file data line by line to fileList(array)
    for line in fileList:
            list.append(line.strip().split(':'))
    fileList.close() #close file
    newFile = [] #for save and clear purpose
    while True: #infinite loop for the menu
            rows = len(list)
            print "(C)reate, (R)ead, (W)rite, (L)ist, (S)ave or (E)xit."
            input = raw_input("Option: ")
            #for creating new file, if the file exist, back to menu
            #if not, create the file and save it into list and newFile
            if input == 'C' or input == 'c':
                    filename = raw_input("Filename: ")
                    for row in xrange(rows):
                            if list[row][0] == filename:
                                    print filename, "is exist in the system!\n"
                                    break;
                    else:
                            security = raw_input("Security level(0 or 1 or 2): ")
                            while security != '0' and security != '1' and security != '2':
                                    print "Wrong security level setting, please try again,"
                                    security = raw_input("Security level(0 or 1 or 2): ")
                            list.append([filename, security])
                            os.system('touch FileServer/'+ filename)
                            print "File", filename, "created\n"
                            newFile.append(filename)
            #for reading a file, check file exist and security level
```

```
		#if exist and have access rights, then "cat <filename>", else back to menu
		elif input == 'R' or input == 'r':
			filename = raw_input("Filename: ")
			for row in xrange(rows):
				if list[row][0] == filename:
					if list[row][1] <= userClearance:
						os.system("cat FileServer/"+ filename)
						print '\n'
						break
					else:
						print "User clearance is lower than the file security
level.\n"
						break
			else:
				print filename, "is not exist in the system.\n"
		#for writing to a file, perform exactly same as read, but with different command
		elif input == 'W' or input == 'w':
			filename = raw_input("Filename: ")
			for row in xrange(rows):
				if list[row][0] == filename:
					if list[row][1] <= userClearance:
						os.system("vi FileServer/"+ filename)
						break;
					else:
						print "User clearance is lower than the file security
level.\n"
						break;
			else:
				print filename, "is not exist in the system.\n"
		#for listing the file, list the security level and file name.
		elif input == 'L' or input == 'l':
			print "Listing the files..."
			time.sleep(0.2)
			print "Security\tName"
			for row in xrange(rows):
				if userClearance >= list[row][1]:
					print list[row][1] + '\t\t' + list[row][0]
			print '\n'
		#for saving, clear the newFile[] and write list into fileList
		elif input == 'S' or input == 's':
			del newFile[:]
			fileList = open("FileServer/.Files.store", "w")
			for row in xrange(rows):
```

```
                fileList.write(list[row][0] + ':' + list[row][1] + '\n')
        time.sleep(0.3)
        print "Files had successfully saved to the FileServer.\n"
        fileList.close()
#for exiting the program, if the file is not saved, remove everthing
elif input == 'E' or input == 'e':
        print "Shut down the FileServer? (Y)es or (N)o"
        choice = raw_input("Options: ")
        if choice == 'Y' or choice == 'y':
                rows = len(newFile)
                for num in xrange(rows):
                        os.system("rm FileServer/" + newFile[num])
                sys.exit()
#any other input consider as wrong input
else:
        print "Wrong input, please try again."
```

4. Samples of Execution

4.1. Register new user

There is a lot of validation in register new user. For examples, username availability, password policy and user clearance setting.

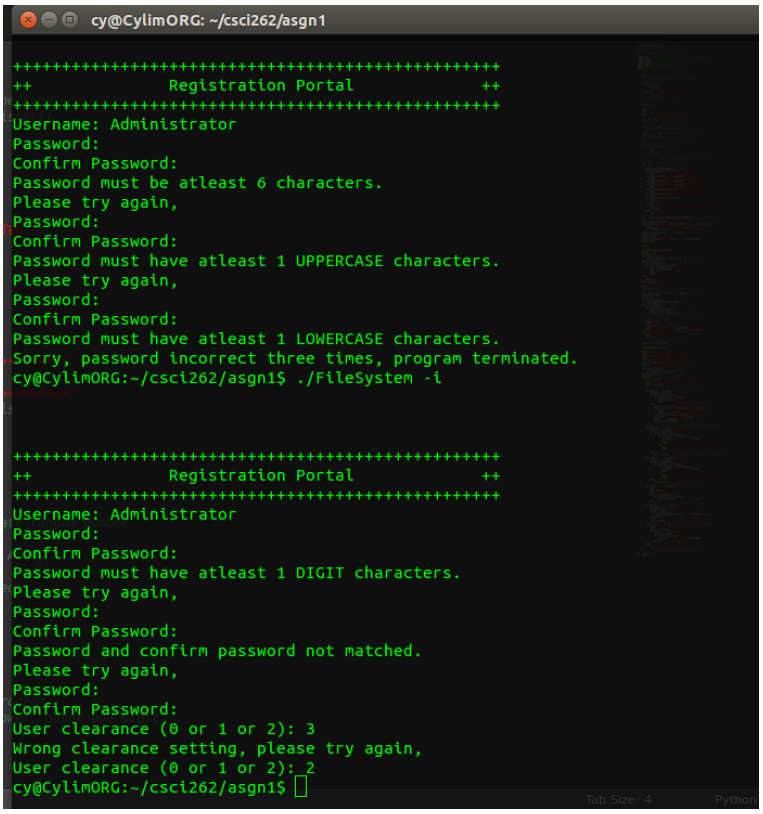


Figure 4.1.1 Registration Screenshot

4.2. Login user

Login user function check the username correctness and the password matched with the salt + hash encrypted password.

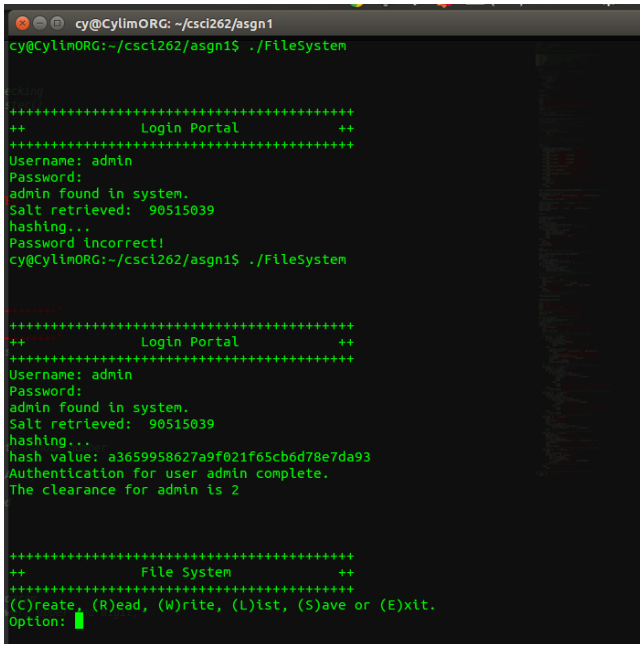
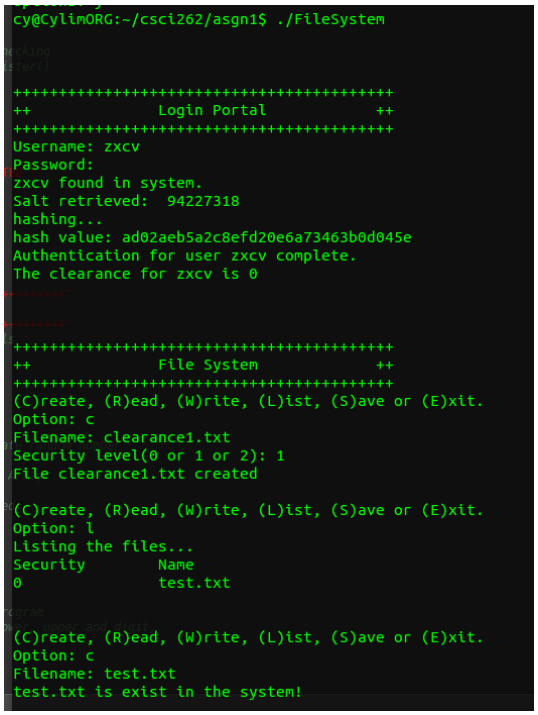
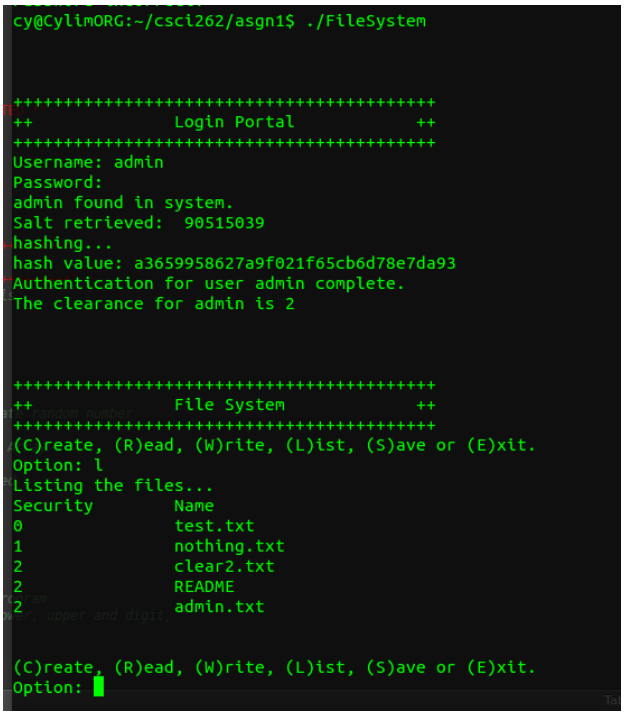


Figure 4.2.1 Login Validation Screenshot

4.3. Create and List

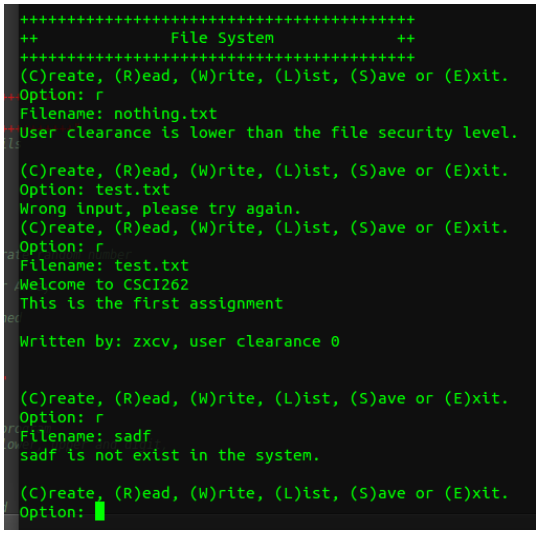
Lower clearance can't list higher security files, but can create higher security files. Filename validation included.

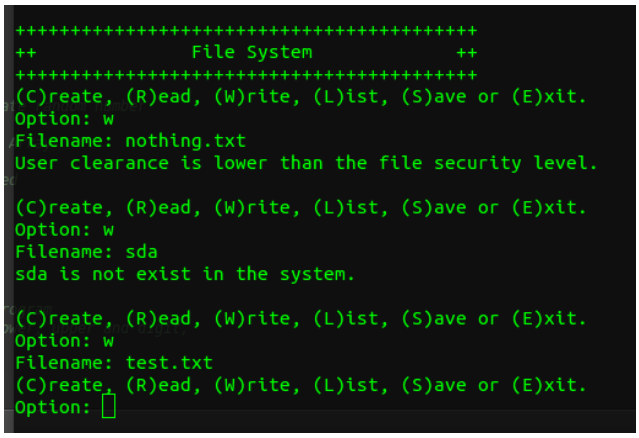




4.4. Read and Write

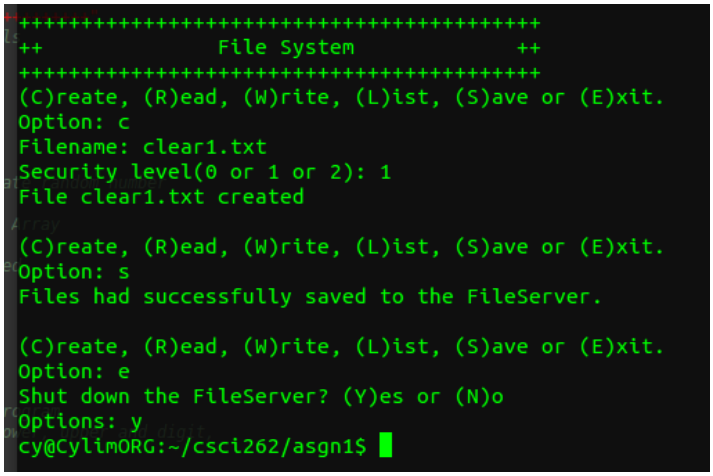
Clearance must be higher or equal to file security in order to read and write the files. File validation also included.





4.5. Save and Exit

If the system terminated without saving, the files create at that particular session will automatically remove from the session. Save function is used to save the files into files.store list.



5. Conclusion

The author appreciated that the lecturer used this method of teaching, and not to specific the programming language to be used for building the console program, as the purpose of this assessment is to enhance students understanding in certain topics and how Operating System works, in this case: Salt and Hashing. Beside that, the author also have better understanding of why Linux operating system separated user id, encrypted password and a lot other information in different files.