**A Simple Authentication and Authorization Framework**

*1. Source Code*

#author: Junbum Kim

from getpass import getpass

#Account List

accounts = []

#Login attempt counter

login\_count = 0

#Login success

login\_success = False

#username to keep track

username = ""

#---------------------------------------------------------------------#

# Account Object - SETUP FOR ACCOUNT

class account:

def \_\_init\_\_(self, user, pwd):

self.username = user

self.password = pwd

def get\_username(self):

return self.username

def get\_password(self):

return self.password

# Open user login database

login\_db = open("users.txt", "rw+")

login\_db\_string = login\_db.read()

login\_db.close()

# STORE ACCOUNTS

# Tokenize newline character

accounts\_list = login\_db\_string.split()

# Tokenize with delimiter : and store into a list of account object

for x in accounts\_list:

user\_pass\_list = x.split(":")

account\_to\_add = account(user\_pass\_list[0], user\_pass\_list[1])

accounts.append(account\_to\_add)

# LOG IN CHECK

def login\_correct(username, password):

result = False

for x in accounts:

if username == x.get\_username() and password == x.get\_password():

result = True

break

return result

#---------------------------------------------------------------------#

#ATTEMP LOGIN - LOGIN PROCEDURE

print "---------------------"

while login\_count < 3 and not login\_success:

#prompt user for username and password (echo off)

username = raw\_input("User name: ")

password = getpass()

#prompt username again if longer than 8 characters

if len(username) > 8:

print "\*\*\*User name cannot be longer than 8 alphanumerica characters\*\*\*"

#username or password does not match

elif login\_correct(username, password) == False:

print "\*\*\*Incorrect username or password.\*\*\*"

#username and password matches

else:

login\_success = True

print "Login Success!"

print "---------------------"

break

#LOGIN FAIL

#increment login count and indicate left login attemps

login\_count = login\_count + 1

print "---------------------"

print str(3 - login\_count) + " login attemp(s) left."

#if no login attempts left and login is unsuccessful, exit program

if login\_count == 3 and login\_success == False:

print "LOGIN FAIL. EXIT PROGRAM"

print "---------------------"

exit(0)

#---------------------------------------------------------------------#

#LOGGED IN - SETUP FOR AUTH

#Authentication Object

class auth:

def \_\_init\_\_(self, action, user, file\_name):

self.action = action

self.user = user

self.file = file\_name

def get\_action(self):

return self.action

def get\_user(self):

return self.user

def get\_file(self):

return self.file

#authentication list

auth\_list = []

# Open user login database

auth\_db = open("auth.txt", "rw+")

auth\_db\_string = auth\_db.read()

auth\_db.close()

# STORE ACTIONS

# Tokenize newline character

action\_list = auth\_db\_string.split()

# Tokenize with delimiter : and store into a list of auth object

for x in action\_list:

actions\_list = x.split(":")

auth\_to\_add = auth(actions\_list[0], actions\_list[1], actions\_list[2])

auth\_list.append(auth\_to\_add)

#list of files

files = []

for x in auth\_list:

if x.get\_file() not in files:

files.append(x.get\_file())

files.remove("")

files = sorted(files)

#list of files with denied files

global\_permit = []

permitted\_files = []

permit\_all = []

global\_deny = []

denied\_files = []

deny\_all = []

#----------------------------------------------------------------------#

#AUTHENTICATION

print "Authentication Information:"

for x in auth\_list:

if x.get\_action() == "PERMIT":

if x.get\_user() == '':

if x.get\_file() == '':

global\_permit = files

else:

global\_permit.append(x.get\_file())

files.remove(x.get\_file())

elif x.get\_user() == username:

if x.get\_file() == "":

permit\_all = files

else:

permitted\_files.append(x.get\_file())

if x.get\_action() == "DENY":

if x.get\_user() == '':

if x.get\_file() == '':

global\_deny = files

else:

global\_deny.append(x.get\_file())

files.remove(x.get\_file())

elif x.get\_user() == username:

if x.get\_file() == "":

deny\_all = files

else:

denied\_files.append(x.get\_file())

#Get Rid of files that are both permitted and denied from the permitted files

for x in denied\_files:

for y in permit\_all:

if x==y:

permit\_all.remove(y)

for y in global\_permit:

if x==y:

global\_permit.remove(y)

for x in permitted\_files:

for y in deny\_all:

if x == y:

deny\_all.remove(y)

for y in global\_deny:

if x==y:

global\_deny.remove(y)

#Add files

permitted\_files = global\_permit + permitted\_files + permit\_all

denied\_files = global\_deny + denied\_files + deny\_all

#sort files and get rid of duplicates

permitted\_files = list(set(permitted\_files))

denied\_files = list(set(denied\_files))

permitted\_files = sorted(permitted\_files)

denied\_files = sorted(denied\_files)

#echo permitted files

for x in permitted\_files:

fo = open(x,"r")

print fo.read()

fo.close()

#print denied files

for x in denied\_files:

print "Access to file <" + x + "> denied."

**File1. users.txt:**

userA:1

userB:2

userC:3

userD:4

userE:5

**File2. auth.txt:**

PERMIT::file1

PERMIT:userA:

DENY:userB:file4

PERMIT:userB:file2

PERMIT:userB:file3

DENY:userC:file1

PERMIT:userC:file3

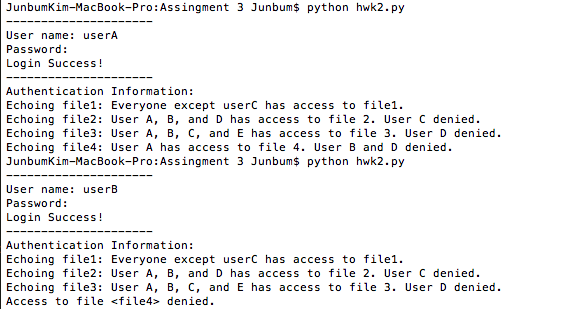
PERMIT:userD:

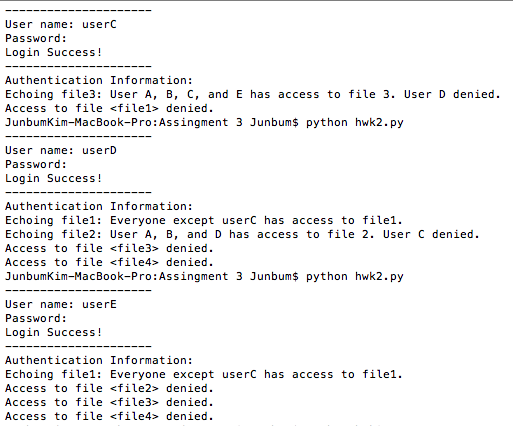
DENY:userD:file3

DENY:userD:file4

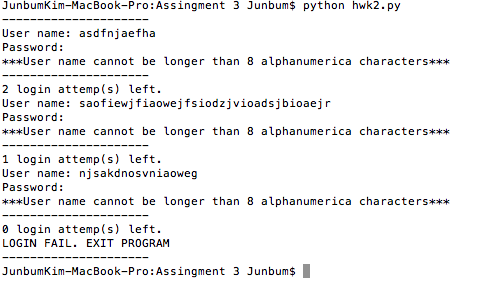
DENY:userE:

**Testing for authentication:**





**Testing for Login:**



*2. Explain how you would set up the following: “user1 may access file1 or file2. User2 may access file2 only. User3 may access any file.”*

PERMIT:user1:file1

PERMIT:user1:file2

DENY:user2:

PERMIT:user2:file2

PERMIT:user3:

*3. Consider how you would implement the following unusual request: “Authenticated users may have access to ANY files in the directory, except the user eve is forbidden to access the file named adam”*

PERMIT::

DENY:eve:adam

*4. What are the ramifications of the databases being in the same directory as the files?*

If the database is in the same directory as the files, the users will have access to the database, and will be able to edit the access control of files.