Министерство науки И ВЫСШЕГО ОБРАЗОВАНИЯ   
Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Национальный исследовательский Томский политехнический Университет»



Инженерная школа автоматизации и робототехники

Отделение автоматизации и робототехники

15.04.04 «Автоматизация технологических процессов и производств»

Вариант 1

Лабораторная работа №2

**«Реализация политики безопасности»**

по дисциплине:

**«Информационная безопасность автоматизированных систем»**

**Исполнитель:**

студент группы 8ТМ22 Гао Аоцзе \_\_\_\_\_\_\_\_\_\_

**Руководитель:**

К.т.н., доцент ОАР Суходоев Михаил Сергеевич \_\_\_\_\_\_\_\_\_\_

Томск – 2023

## Цель работы

Изучить способы конфигурации политик безопасности в компьютерных системах и программно реализовать дискреционную модель политики безопасности.

## Задачи

Пусть множество S возможных операций над объектами компьютерной сис­темы задано следующим образом: S = {«Доступ на чтение», «Доступ на запись», «Доступ на модификацию»}.

1. Получить данные о количестве пользователей и объектов компьютерной системы из табл. 1, соответственно варианту.

2. Реализовать программный модуль, создающий матрицу доступа пользова­телей к объектам компьютерной системы. Реализация данного модуля подразуме­вает следующее:

2.1. Необходимо выбрать идентификаторы пользователей, которые будут использоваться при их входе в компьютерную систему (по одному идентификатору для каждого пользователя, количество пользователей указано для варианта). Например, множество из трёх идентификаторов пользователей {Ivan, Sergey, Boris}. Один из данных идентификаторов должен соответствовать администратору компьютерной системы (пользователю, обладающему полными правами доступа ко всем объектам).

2.2. Реализовать программное заполнение матрицы доступа, содержащей ко­личество пользователей и объектов, соответственно варианту.

2.2.1. При заполнении матрицы доступа необходимо учитывать, что один из пользователей должен являться администратором системы (допустим, Ivan). Для него права доступа ко всем объектам должны быть выставлены как полные.

2.2.2. Права остальных пользователей для доступа к объектам компьютерной системы должны заполняться случайным образом с помощью датчика случайных чисел (либо произвольно вручную при создании матрицы). При заполнении матрицы доступа необходимо учитывать, что пользователь может иметь несколько прав доступа к некоторому объекту компьютерной сис­темы, иметь полные права, либо совсем не иметь прав.

2.2.3. Реализовать программный модуль, демонстрирующий работу в дискре­ционной модели политики безопасности.

3. Модуль должен выпол­нять следующие функции:

3.1. При запуске модуля должен запрашиваться идентификатор пользователя: при успешной идентифика­ции должен осуществляться вход в систему, при неуспешной – выводиться соответствующее сообщение.

3.2. При входе в систему после успешной идентификации пользователя на экране должен распечатываться список всех объектов системы с указанием пе­речня всех доступных прав доступа идентифицированного пользователя к данным объектам. Вывод можно осуществить, например, следующим образом:

User: Boris

Идентификация прошла успешно, добро пожаловать в систему

Перечень Ваших прав:

Объект1: Чтение

Объект2: Запрет

Объект3: Чтение, Запись

Объект4: Полные права

Жду ваших указаний >

3.3. После вывода на экран перечня прав доступа пользователя к объектам компьютерной системы, необходимо организовать ожидание указаний пользователя на осу­ществление действий над объектами в компьютерной системе. После получения команды от пользователя, на экран необходимо вывести сообщение об успешности либо не успешности операции. Программа должна поддерживать операцию выхода из системы (quit). Диалог можно организовать, например, так:

Жду ваших указаний > read

Над каким объектом производится операция? 1

Операция прошла успешно

Жду ваших указаний > write

Над каким объектом производится операция? 2

Отказ в выполнении операции. У Вас нет прав для ее осуществления

Жду ваших указаний > quit

Работа пользователя Boris завершена. До свидания.

4. Выполнить тестирование разработанной программы, продемонстрировав реализованную модель дискреционной политики безопасности.

5. Оформить отчет по лабораторной работе.

**Таблица 1. Варианты заданий**

|  |  |  |
| --- | --- | --- |
| Вариант | Количество субъектов доступа (пользователей) | Количество объектов доступа |
| 1 | 3 | 6 |

## Ход работы

В соответствии с требованием, мы реализуем доступ пользователя по логину. Согласно вариантам, у меня есть три пользователя и шесть файлов, каждый файл разделен на три разрешения: чтение, запись и выполнение. Один из трех пользователей является администратором и имеет все права на все шесть файлов. Остальные два пользователя имеют случайным образом сгенерированные разрешения на шесть файлов, которые включают в себя несколько разрешений, все разрешения и ни одного. Запустите модуль, введите идентификатор пользователя и после успешного распознавания отобразите информацию о пользователе, его разрешения и типы для всех файлов. Пользователь должен иметь возможность вводить команды чтения, записи и выполнения для манипулирования файлами и выводить сообщение об успешности или неуспешности манипулирования файлами. Программа также должна поддерживать функцию выхода. Когда пользователь вводит команду quit, программа подтверждает получение команды и закрывает программу. Также должна быть добавлена возможность переключения пользователей.

Дополнительно: В программе также должна быть реализована функция предоставления прав доступа, только руководство может предоставлять права другим пользователям, а после предоставления прав доступа предоставлять другим пользователям указанные права на указанный файл. После завершения операции матрица доступа должна быть изменена.

Передача прав доступа, после передачи прав доступа передающий сохраняет права доступа.

Выше приведена функциональная реализация, ядро программирования представляет собой математическую модель матрицы доступа. Столбцы этой матрицы соответствуют объектам системы, а строки - субъектам, поэтому я матрица имеет три строки и шесть столбцов. Элементами матрицы являются типы доступа (например, чтение, запись, выполнение).

Потоковый анализ интерфейса программы

Шаг 1: Введите имя пользователя;

Шаг 2: Отображение файлов и разрешений.

Пожалуйста, выберите файл, с которым вы хотите работать, или выйдите из программы, или измените имя пользователя;

Шаг 3: Отображение операции с правами доступа;

Шаг 4: Обратная связь; (дождаться инструкций, инструкции включают продолжение использования разрешений, возврат к выбранному файлу, выход из программы)

Шаг 5: Дождаться инструкций (продолжить работу или выйти).

Для предоставления разрешений предположим, что A предоставляет B доступ на чтение к файлу 1, и после предоставления A сохраняет доступ на чтение к файлу 1, а B также имеет доступ на чтение к файлу 1. При реализации кода важно отметить, что перед предоставлением прав проверяется матрица математической модели на наличие прав у B. Если B уже имеет эти права, то необходимо напомнить об этом A. Если B уже имеет разрешение, то следует напомнить об этом A. Если B не имеет разрешения, то пусть A подтвердит операцию предоставления разрешения. По завершении операции предоставления прав дайте обратную связь.

Сначала мы импортируем модуль и определяем структуры данных.

Модуль random используется для генерации случайных чисел.

resources - список ресурсов.

permissions - список разрешений.

users - словарь, содержащий пользователей и соответствующие им ресурсы и разрешения.

import random

# 资源和权限

resources = ['Resource1', 'Resource2', 'Resource3', 'Resource4', 'Resource5', 'Resource6']

permissions = ['read', 'write', 'execute']

# 用户信息

users = {

'admin': {resource: permissions for resource in resources},

'user1': {resources[random.randint(0, 5)]: random.sample(permissions, random.randint(0, 3)) for \_ in range(3)},

'user2': {resources[random.randint(0, 5)]: [random.choice(permissions)] for \_ in range(2)}

}

# 当前用户

current\_user = None

Далее мы определяем три функции:

Функция display\_user\_info(user) используется для отображения информации о пользователе и его правах доступа.

Функция switch\_user() используется для переключения пользователей.

Функция perform\_permission\_operation() используется для выполнения операции разрешения.

# 函数：显示用户信息和权限

def display\_user\_info(user):

print(f"User: {user}")

print("Permissions:")

for resource, perms in users[user].items():

print(f"{resource}: {', '.join(perms)}")

# 函数：切换用户

def switch\_user():

global current\_user

username = input("Enter username (or 'quit' to exit): ")

if username == 'quit':

return False

elif username in users:

current\_user = username

print(f"Switched to user: {username}")

display\_user\_info(username)

else:

print("Invalid username. Please try again.")

return True

# 函数：执行权限操作

def perform\_permission\_operation():

global current\_user

resource = input("Enter the resource you want to operate on: ")

permission = input("Enter the permission you want to check: ").strip().lower()

if current\_user in users and resource in users[current\_user]:

if permission in users[current\_user][resource]:

print(f"{current\_user} has {permission} permission for {resource}.")

else:

print(f"{current\_user} does not have {permission} permission for {resource}.")

else:

print(f"{current\_user} does not have access to {resource}.")

Далее следует определить основную программу.

Функция main() - это основная логика программы, которая перебирает команды, вводимые пользователем, и вызывает соответствующие функции.

Если пользователь вводит команду 'switch', то для переключения пользователей вызывается функция switch\_user().

Если пользователь вводит команду 'operate', то вызывается функция perform\_permission\_operation().

Если пользователь вводит 'quit', происходит выход из приложения.

\_\_name\_\_ == "\_\_main\_\_": Убедитесь, что при прямом запуске скрипта выполняется функция main().

def main():

global current\_user

print("Welcome to the resource management program!")

while True:

if not current\_user:

if not switch\_user():

break

command = input("Enter a command (switch/operate/quit): ").strip().lower()

if command == 'switch':

if not switch\_user():

break

elif command == 'operate':

perform\_permission\_operation()

elif command == 'quit':

break

else:

print("Invalid command. Please try again.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

тестирование кода

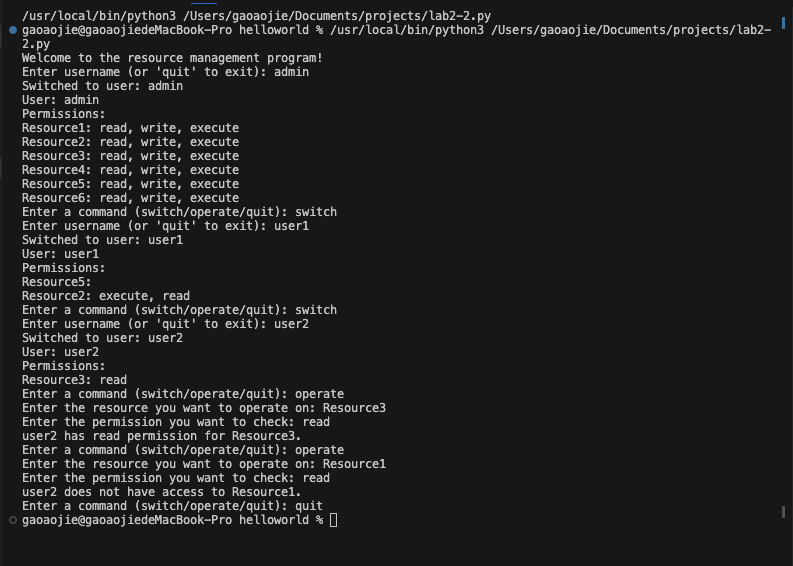


Рисунок 1 - результат теста перваого

В этом тесте мы сначала вошли в учетную запись администратора, которая показала все разрешения на шесть ресурсов. После этого мы переключились на пользователя 1, которому были выданы случайные разрешения. Переключившись на пользователя 2, мы снова получили случайные разрешения. Мы выполнили операцию разрешения на чтение ресурса 3, после чего получили сообщение о завершении операции. Система выдает подсказку для ресурсов, для работы с которыми нет разрешения. Наконец, мы реализовали ввод quit для выхода из программы.

Далее мы добавляем функцию передачи прав. Мы установим, что передавать права могут только администраторы. Администратор может выбрать передачу прав от одного пользователя к другому. После передачи прав передающий сохраняет их за собой.

Реализация кода выглядит следующим образом

import random

# 资源和权限

resources = ['Resource1', 'Resource2', 'Resource3', 'Resource4', 'Resource5', 'Resource6']

permissions = ['read', 'write', 'execute']

# 用户信息

users = {

'admin': {resource: permissions for resource in resources},

'user1': {resources[random.randint(0, 5)]: random.sample(permissions, random.randint(0, 3)) for \_ in range(3)},

'user2': {resources[random.randint(0, 5)]: [random.choice(permissions)] for \_ in range(2)}

}

# 当前用户

current\_user = None

# 函数：显示用户信息和权限

def display\_user\_info(user):

print(f"User: {user}")

print("Permissions:")

for resource, perms in users[user].items():

print(f"{resource}: {', '.join(perms)}")

# 函数：切换用户

def switch\_user():

global current\_user

username = input("Enter username (or 'quit' to exit): ")

if username == 'quit':

return False

elif username in users:

current\_user = username

print(f"Switched to user: {username}")

display\_user\_info(username)

else:

print("Invalid username. Please try again.")

return True

# 函数：执行权限操作

def perform\_permission\_operation():

global current\_user

resource = input("Enter the resource you want to operate on: ")

permission = input("Enter the permission you want to check: ").strip().lower()

if current\_user in users and resource in users[current\_user]:

if permission in users[current\_user][resource]:

print(f"{current\_user} has {permission} permission for {resource}.")

else:

print(f"{current\_user} does not have {permission} permission for {resource}.")

else:

print(f"{current\_user} does not have access to {resource}.")

# 函数：执行权限转让

def perform\_permission\_transfer():

global current\_user

if current\_user != 'admin':

print("Only admin can perform permission transfer.")

return

transfer\_from = input("Enter the username from which you want to transfer permissions: ")

transfer\_to = input("Enter the username to which you want to transfer permissions: ")

resource = input("Enter the resource for the permission transfer: ")

permission = input("Enter the permission for the transfer: ").strip().lower()

if transfer\_from in users and transfer\_to in users and resource in users[transfer\_from]:

if permission in users[transfer\_from][resource]:

users[transfer\_to].setdefault(resource, []).append(permission)

print(f"Permission {permission} for {resource} transferred from {transfer\_from} to {transfer\_to}.")

else:

print(f"{transfer\_from} does not have {permission} permission for {resource}.")

else:

print("Invalid usernames or resource.")

# 主程序

def main():

global current\_user

print("Welcome to the resource management program!")

while True:

if not current\_user:

if not switch\_user():

break

command = input("Enter a command (switch/operate/transfer/quit): ").strip().lower()

if command == 'switch':

if not switch\_user():

break

elif command == 'operate':

perform\_permission\_operation()

elif command == 'transfer':

perform\_permission\_transfer()

elif command == 'quit':

break

else:

print("Invalid command. Please try again.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

Тест кода с функцией передачи прав доступа

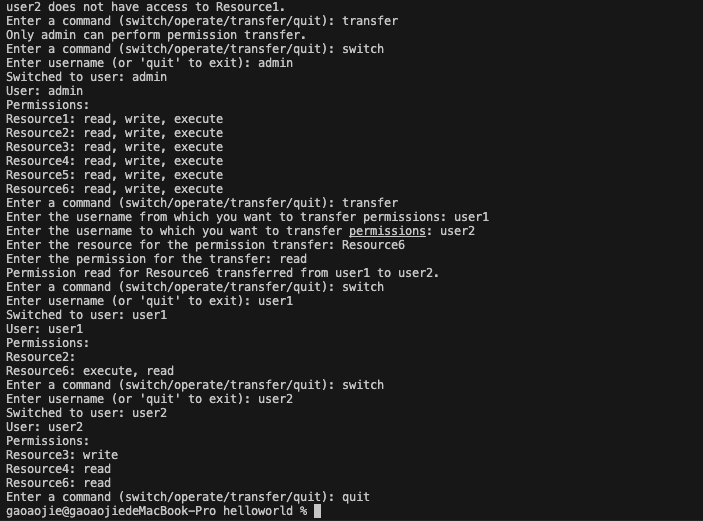


Рисунок 2 - результат теста второго

Начальные шаги тестирования такие же, как и раньше, на рисунке показана передача прав при входе в учетную запись под именем пользователя 2, система выдает сообщение о том, что передавать права могут только администраторы. После этого мы переходим в учетную запись администратора и передаем разрешение на чтение ресурса 6 от пользователя 1 пользователю 2. После завершения передачи мы входим в учетную запись пользователя 2 и видим добавленное разрешение.

## Вывод

В данном эксперименте с помощью языка python моделирована система управления правами, которая позволяет пользователям переключаться, просматривать права, выполнять операции с правами и осуществлять передачу прав. Администратор обладает специальными привилегиями, которые могут быть переданы, а другие пользователи могут просматривать только свои собственные привилегии.

Приложение

import random

# 资源和权限

resources = ['Resource1', 'Resource2', 'Resource3', 'Resource4', 'Resource5', 'Resource6']

permissions = ['read', 'write', 'execute']

# 用户信息

users = {

'admin': {resource: permissions for resource in resources},

'user1': {resources[random.randint(0, 5)]: random.sample(permissions, random.randint(0, 3)) for \_ in range(3)},

'user2': {resources[random.randint(0, 5)]: [random.choice(permissions)] for \_ in range(2)}

}

# 当前用户

current\_user = None

# 函数：显示用户信息和权限

def display\_user\_info(user):

print(f"User: {user}")

print("Permissions:")

for resource, perms in users[user].items():

print(f"{resource}: {', '.join(perms)}")

# 函数：切换用户

def switch\_user():

global current\_user

username = input("Enter username (or 'quit' to exit): ")

if username == 'quit':

return False

elif username in users:

current\_user = username

print(f"Switched to user: {username}")

display\_user\_info(username)

else:

print("Invalid username. Please try again.")

return True

# 函数：执行权限操作

def perform\_permission\_operation():

global current\_user

resource = input("Enter the resource you want to operate on: ")

permission = input("Enter the permission you want to check: ").strip().lower()

if current\_user in users and resource in users[current\_user]:

if permission in users[current\_user][resource]:

print(f"{current\_user} has {permission} permission for {resource}.")

else:

print(f"{current\_user} does not have {permission} permission for {resource}.")

else:

print(f"{current\_user} does not have access to {resource}.")

# 主程序

def main():

global current\_user

print("Welcome to the resource management program!")

while True:

if not current\_user:

if not switch\_user():

break

command = input("Enter a command (switch/operate/quit): ").strip().lower()

if command == 'switch':

if not switch\_user():

break

elif command == 'operate':

perform\_permission\_operation()

elif command == 'quit':

break

else:

print("Invalid command. Please try again.")

if \_\_name\_\_ == "\_\_main\_\_":

main()