МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Информационной безопасности

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Модели безопасности компьютерных систем»
Тема: Информационный поток по памяти

Алексеев М. Д.
Студенты гр. 1362 Чернодоля П. И.
Преподаватель Шкляр Е. В.

Санкт-Петербург
2024

Теоретическая справка.

Сущность в произвольный момент времени может быть однозначно представлена словом некоторого языка (набором данных), которое может рассматриваться как состояние сущности.

Субъект – сущность, инициирующая выполнение операций над сущностями.

Объект (object), или контейнер (container) – сущность, содержащая или получающая информацию, и над которой субъекты выполняют операции.

Контейнеры могут состоять из объектов или других контейнеров.

Субъекты могут получать доступ к объектам целиком, но не к их части.

Субъекты могут получать доступ к контейнеру и сущностям, из которых он состоит.

Для выполнения операций над сущностями субъекты осуществляют доступы к ним.

Основные виды доступов:

read – на чтение из сущности;

write – на запись в сущность;

append – на запись в конец слова, описывающего состояние сущности;

execute – на активацию субъекта из сущности.

Информационный поток по памяти – информационный поток, при реализации которого фактор времени не является существенным.

Информационный поток по времени – информационный поток, при реализации которого фактор времени является существенным (например, передача данных осуществляется путем изменения продолжительности интервалов времени между событиями в КС или путем изменения последовательности событий).

Дискреционная политика управления доступом – политика, соответствующая следующим требованиям:

- Все сущности идентифицированы (т.е. каждой сущности присвоен уникальный идентификатор);

- Задана матрица доступов, каждая строка которой соответствует субъекту, а столбец – сущности КС, ячейка содержит список прав доступа субъекта к сущности;
- Субъект обладает правом доступа к сущности КС тогда и только тогда, когда в соответствующей ячейке матрицы доступов содержится данное право доступа.

Мандатная политика управления доступом – политика, соответствующая следующим требованиям:

- Все сущности идентифицированы;
- Задана решетка уровней конфиденциальности информации;
- Каждой сущности присвоен уровень конфиденциальности, задающий установленные ограничения на доступ к данной сущности;
- Каждому субъекту присвоен уровень доступа, задающий уровень полномочий данного субъекта в КС;
- Субъект обладает правом доступа к сущности КС тогда, когда уровень доступа субъекта позволяет предоставить ему данный доступ к сущности с заданным уровнем конфиденциальности, и реализация доступа не приведет к возникновению информационных потоков от сущностей с высоким уровнем конфиденциальности к сущностям с низким уровнем.

Политика ролевого управления доступом - политика, соответствующая следующим требованиям:

- Все сущности идентифицированы;
- Задано множество ролей, каждой из которых ставится в соответствие некоторое множество прав доступа к сущностям;
- Каждый субъект обладает некоторым множеством ролей;
- Субъект обладает правом доступа к сущности КС тогда, когда он обладает ролью, которой соответствует множество прав доступа, содержащее право доступа к данной сущности.

Является развитием дискреционного управления доступом, при этом позволяет определить более четкие и понятные для пользователей правила. Позволяет реализовывать гибкие правила управления доступом.

Политика безопасности информационных потоков основана на разделении всех возможных информационных потоков (ИП) между сущностями КС на два непересекающихся множества: множество разрешенных ИП и множество запрещенных ИП.

Цель – обеспечить невозможность возникновения в КС запрещенных ИП.

Как правило, реализуется в сочетании с политикой другого вида (дискреционной, мандатной или ролевой). Реализация крайне сложна, т.к. требует защиту от возникновения запрещенных потоков по времени.

Политика изолированной программной среды реализуется путем изоляции субъектов КС друг от друга и путем контроля порождения новых субъектов так, чтобы в системе могли активи-зироваться только субъекты из определенного списка.

Цель – задание порядка безопасного взаимодей-ствия субъектов КС, обеспечивающего невоз-можность воздействия на систему защиты КС и модификации ее параметров или конфигурации, результатом которого могло бы стать изменение политики управления доступом.

Задача.

Реализовать программу, реализующую создание файла, хранящего строку текста с заданным названием в приватную папку, а также копирование этого файла по запросу пользователя в общедоступную папку.

Реализовать программу нарушителя, которая позволяет просматривать содержимое публичной папки и копировать новые файлы с информацией в папку нарушителя.

Ход работы.

1. В первую очередь, была реализована программа нарушитель.

- Был создан интерфейс программы с использованием библиотеки python tkinter. Конечный интерфейс программы представлен на рисунке 1.

Рисунок 1 – Интерфейс программы

- На рисунке мы можем видеть 2 списка – содержание публичной папки и папки злоумышленника. При нажатии кнопки “Запустить сканнер” содержимое списка “Папка злоумышленника” будет дополнятсья недостающими файлами.

- После реализации интерфейса и функционала программы, была проведена проверка на корректную работу.

- Запустим программу. Не включая сканнер добавим новый файл в публичную директорию. Содержимое директории и списка файлов “Публичная папка” представлено на рисунках 2 и 3.

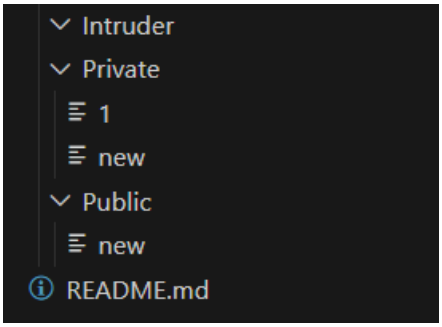

Рисунок 2 – Содержимое директорий.

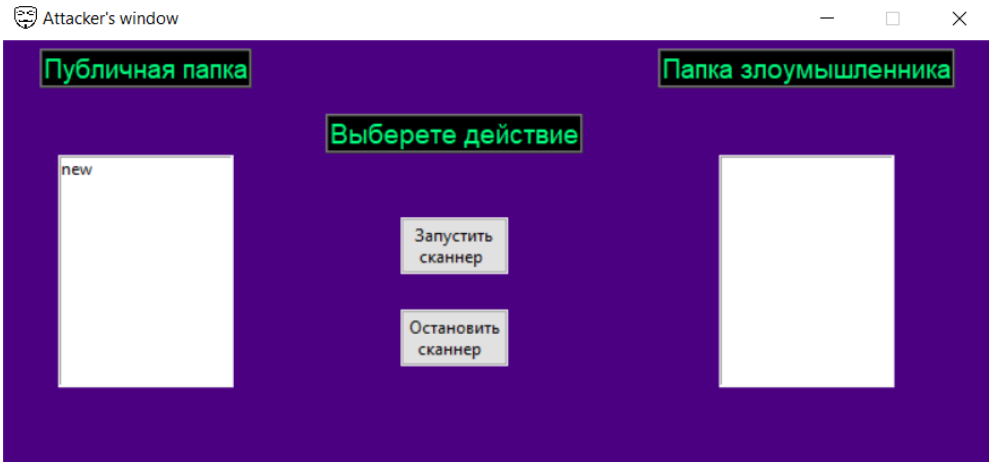

Рисунок 3 – Содержимое списков файлов директорий в программе.

- Нажмем кнопку запустить сканнер. После нажатия недостающий файл “new” будет перенесен в папку злоумышленника. Результат рабоыт сканнера изображен на рисунке 4.

Рисунок 4 – Результат работы сканнера.

2) Далее была реализована программа пользователя, которая может создавать файлы с заданным именем и содержанием в приватной папке, перемещать файлы из приватной папки в общедоступную. Интерфейс программы представлен на рисунке 5.

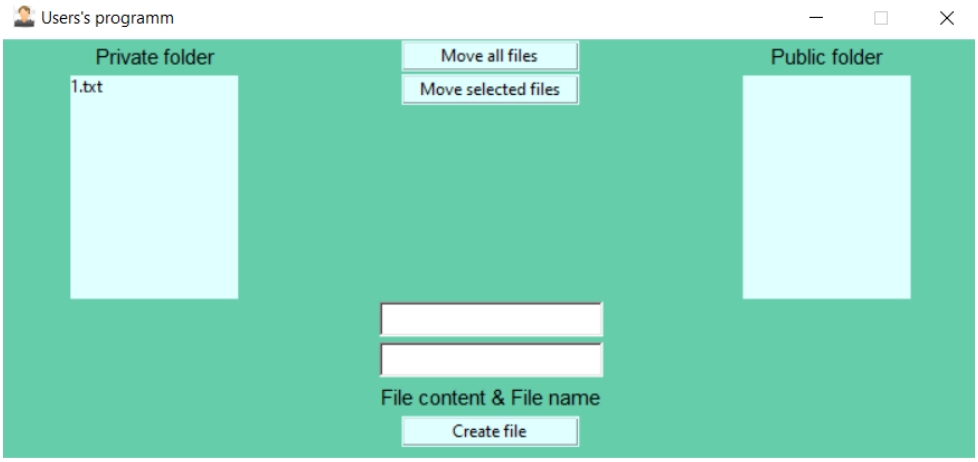

Рисунок 5 – Интерфейс программы пользователя

Создадим файл, записав в первое поле ввода “Hello World!” а во второе поле – название файла test.txt. Нажмем кнопку “Create file”. Результат представлен на рисунке 6.

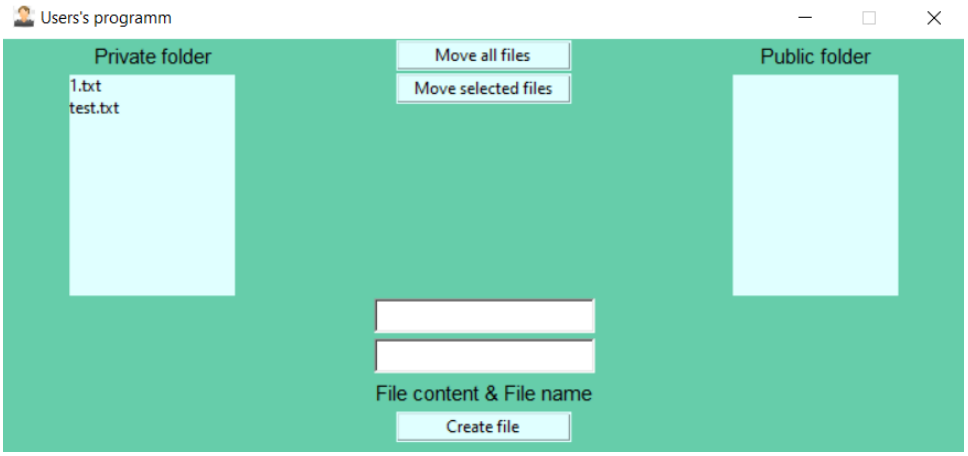

Рисунок 6 – Создание файла

Как видно из рисунка 6, после создания файла, список приватной папки автоматически обновился и в нем появился новый файл. Откроем этот файл через проводник и проверим его содержимое. Результат на рисунке 7.

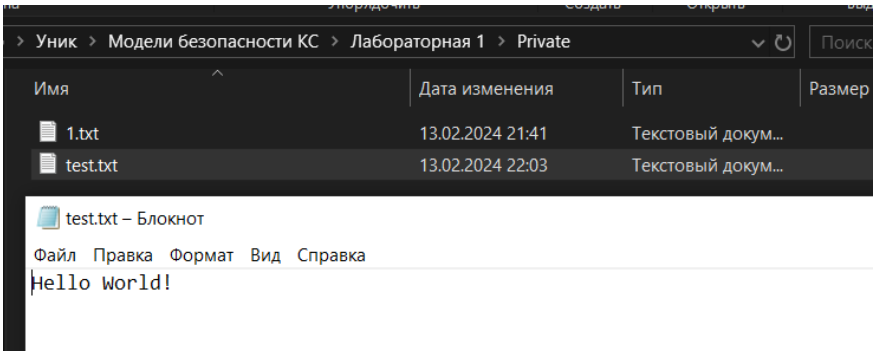

Рисунок 7 – Созданный файл

Как видно из рисунка 7 – файл создался корректно. Теперь выберем его и нажмем кнопку “Move selected files”. Результат представлен на рисунке 8.

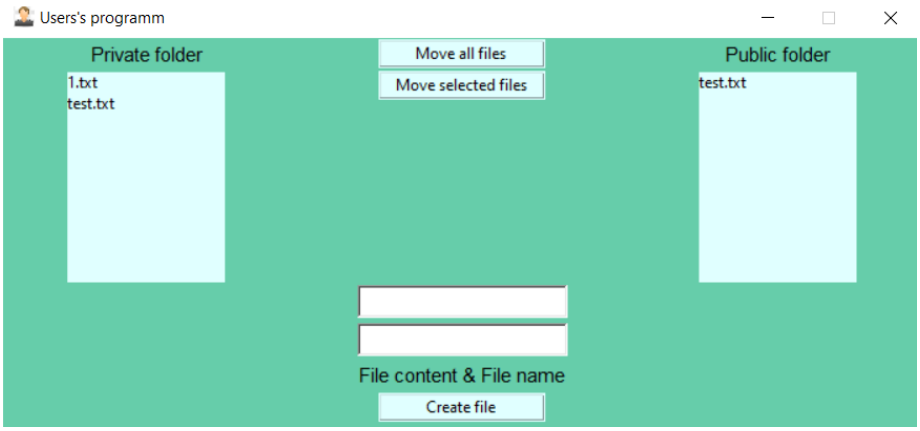

Рисунок 8 – Перенос файла

Копия файла появилась в публичной папке.

Создадим много новых файлов. Результат на рисунке 9.

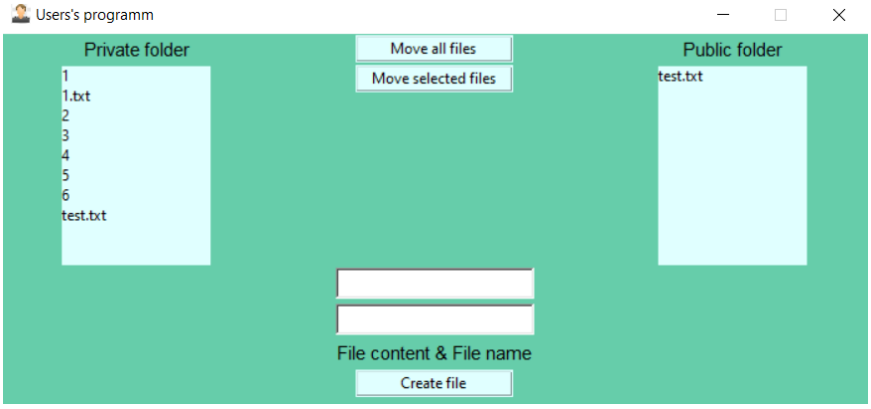

Рисунок 9 – Много новых файлов

Нажмем на кнопку “Move all files”. Результат после нажатия на рисунке 10.

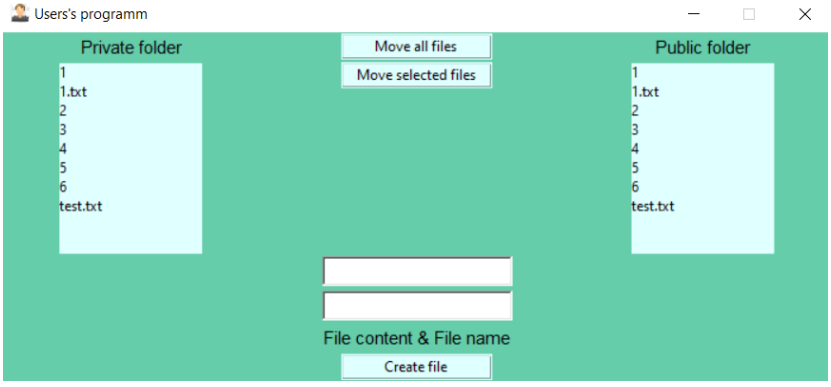

Рисунок 10 – Перенос всех файлов

Все файлы перенеслись в публичную папку.

Выводы.

В результате выполнения данной лабораторной работы было реализовано два различных программных средства:

1. Программное средство создающее уязвимость путем создания и копирования содержимого файла в директорию, позволяющую субъекту, не обладающему доступом на чтение прочитать его содержимое и использовать в угодных себе целях.

2. Программное средство эксплуатирующее уязвимость по копированию содержимого приватного файла, содержащего в себе секретную информацию.

Обе программы показывают важность распределения доступа к файлам для различных групп субъектов и ограничения доступа информационного потока по памяти.

Код приложения.

1. Программа пользователя.

```
import os
import shutil
from tkinter import *
from tkinter import ttk
def create_file(entry, listPr):
    direct = "..//Private"
    file_name = entry.get()
    file_content = entryContent.get()
    if    file_name    ==    ""    or    not    set(",:;!_*-
+()/#¤%&)").isdisjoint(file_name) or file_name[0]=="." :
        labelError.config(fg="#FF0000")
        return 1
    labelError.config(fg="#66CDAA")
    file = open(direct+"//"+file_name, 'w+')
    file.write(file_content)
    file.close()
    update_file_list(listPr, "..//Private")
    entry.delete(0, END)
    entryContent.delete(0, END)
    listPr.selection_clear(0, END)
    return 0


def move_file(listPr, listPb, mode):
    if mode == 2:
        shutil.copytree("..//Private",           "..//Public",
dirs_exist_ok = True)
    elif mode == 1:
        for file in listPr.curselection():
            shutil.copy("..//Private//"+listPr.get(file),
"..//Public")
    update_file_list(listPb, "..//Public")
    listPr.selection_clear(0, END)
    return 0


def update_file_list(listbox, directory_path):
    file_list = os.listdir(directory_path)
    listbox.delete(0, END)
    for file in file_list:
        listbox.insert(END, file)
   #      mw.after(1000,      update_file_list,      listbox,
directory_path)  # Проверяем каждую секунду

def hello(list):
    l = list.curselection()
    print(list.get(l))
    return 0
if __name__ == '__main__':
```

```
    dirPr = "..//Private"
    dirPb = "..//Public"

    mw = Tk()
    mw.title("Users's programm")
    mw.geometry('700x300+400+200')
    mw.configure(background="#66CDAA")
    mw.resizable(False, False)
    mw.iconbitmap(default="userIco.ico")

    for i in range(3): mw.columnconfigure(i,weight=1)
    for i in range(8): mw.rowconfigure(i, weight=1)

    # Создание надписей
    labelPrivate = Label(text = "Private folder", background=
"#66CDAA",font =("Arial", 11))
    labelPrivate.grid(row=0,column=0)
    labelPublic      =      Label(text="Public      folder",
background="#66CDAA", font=("Arial", 11))
    labelPublic.grid(row=0, column=2)
    # Создание списка файлов публичной папки
    listPublic         =         Listbox(selectmode=EXTENDED,
yscrollcommand="True",borderwidth=0,     highlightthickness=0,
background="#E0FFFF")
    listPublic.grid(row =1, column =2)
    listPublic.bindtags(("listPublic", "mw", "all"))
    update_file_list(listPublic, dirPb)
    # Создание списка файлов приватной папки
    listPrivate         =         Listbox(selectmode=EXTENDED,
yscrollcommand="True",   borderwidth=0,   highlightthickness=0,
background="#E0FFFF")
    listPrivate.grid(row=1, column=0)
    update_file_list(listPrivate, dirPr)


    # Настройка стиля для кнопок
    style = ttk.Style()
    style.theme_use('alt')
    style.configure('TButton',
                    background="#E0FFFF",
                    width=20,
                    borderwidth=1,
                    focusthickness=3,
                    focuscolor='none')
    style.map('TButton', background=[('active', "#AFEEEE")])

    # Создание кнопки копирования всех файлов
    btCopyAll   =   ttk.Button(text   =   "Move   all   files",
command=lambda: move_file(listPrivate, listPublic,2))
    btCopyAll.grid(row= 0,column = 1)
    # Создание кнопки копирвоания выделенного файла
```

```
    btCopySelect = ttk.Button(text="Move selected files",
command=lambda: move_file(listPrivate, listPublic, 1))
    btCopySelect.grid(row=1, column=1, sticky="n")

    # Создание кнопки создания файла
    btCreateFile = ttk.Button(text="Create file",
command=lambda: create_file(entry, listPrivate))
    btCreateFile.grid(row=5, column=1)
    # Создание поля для ввода имени файла
    entry = ttk.Entry(width = 25)
    entry.grid(row = 3, column=1,ipadx=2, ipady=2 )
    # Создание надписи к полю ввода имени
    labelNewName = Label(text="File content & File name",
background="#66CDAA", font=("Arial", 11))
    labelNewName.grid(row=4, column=1)
    # Создание поля для ввода содержания
    entryContent = ttk.Entry(width = 25)
    entryContent.grid(row = 2, column=1,ipadx=2, ipady=2,
sticky="n")

     # Создание надписи при неправильном вводе имени файла
    labelError = Label(text="Incorrect name",
background="#66CDAA", foreground="#66CDAA", font=("Arial",
11))
    #FF0000
    labelError.grid(row=4, column=0)

    mw.mainloop()
```

2. Программа нарушителя.

```
import os
import shutil
from tkinter import *
from tkinter import ttk

init_scanning = False
def copying(diff_list):

    for file in diff_list:
        buffer = ""
        copy_stream = open("../Public/"+str(file), "r")
        buff = copy_stream.read()
        create_stream = open("../Intruder/"+str(file), "w")
        create_stream.write(buff)
        copy_stream.close()
        create_stream.close()

def timed_checker(status=None):
    global init_scanning

    if status == "start":
```

```
            init_scanning = True
            st_but["text"] = "Сканнер\nзапущен"
        if status == "stop":
            init_scanning = False
            st_but["text"] = "Запустить\n сканнер"
        if init_scanning == True:
            pub_list = os.listdir("../Public")
            int_list = os.listdir("../Intruder")
            pub_list_set=set(pub_list)
            int_list_set=set(int_list)

            diff = pub_list_set.difference(int_list_set)
            diff_list = list(diff)
            copying(diff_list)
            # for file in diff_list:
            #     shutil.copy("..//Public//"+file,
"..//Intruder")
            public_listbox.after(1000, timed_checker)
        return 0


    def update_file_list(listbox, directory_path):
        file_list = os.listdir(directory_path)
        listbox.delete(0, END)
        for file in file_list:
            listbox.insert(END, file)
        window.after(1000, update_file_list, listbox,
directory_path)  # Проверяем каждую секунду

    if __name__ == '__main__':

        window = Tk()
        window.title("Attacker's window")
        window.geometry('700x300')
        window.configure(bg='#4B0082')
        window.iconphoto(False, PhotoImage(file="anon.png"))
        window.resizable(width=False, height=False)

        for i in range(3): window.columnconfigure(index=i,
weight=1)
        for i in range(8): window.rowconfigure(index=i, weight=1)

        label_public = Label(text = "Публичная
папка",background="#000000", foreground="#00FF7F",font=("Arial",
14),relief="ridge")
        label_public.grid(row=0, column=0)
        label_button = Label(text = "Выберете
действие",background="#000000",
foreground="#00FF7F",font=("Arial", 14),relief="ridge")
        label_button.grid(row=1, column=1)
        label_intr = Label(text = "Папка
злоумышленника",background="#000000",
foreground="#00FF7F",font=("Arial", 14),relief="ridge")
```

```
        label_intr.grid(row=0, column=2)

        public_listbox = Listbox(yscrollcommand=True)
        public_listbox.grid(row=1, column=0, rowspan=6)
        intr_listbox = Listbox(yscrollcommand=True)
        intr_listbox.grid(row=1, column=2, rowspan=6)
        update_file_list(public_listbox, "../Public")
        update_file_list(intr_listbox, "../Intruder")

        #ttk.Button(text="Перенести\n      все",command
=move_all).grid(row=2, column=1, padx=2, pady=2)

        st_but = ttk.Button(text="Запустить\n сканнер",\
                    command =lambda: timed_checker("start"))
        st_but.grid(row=3, column=1, padx=2, pady=2)
        ttk.Button(text="Остановить\n  сканнер",\
                    command =lambda:
timed_checker("stop")).grid(row=4, column=1, padx=2, pady=2)


        mainloop()
```