Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Информационные сети. Основы безопасности

**ОТЧЕТ**

К лабораторной работе № 4,5

на тему «Разработка защищенных приложений»

Студент: гр.153504

Тиханенок И. А.

Проверил: Лещенко Е. А.

Минск 2024

**СОДЕРЖАНИЕ**

[1 Цель работы 3](#_Toc157412482)

[2 Краткие теоретические сведения 4](#_Toc157412483)

[3 Демонстрация программного средства 13](#_Toc157412484)

[Приложение А Исходный код программы 14](#_Toc157412485)

# 1 ЦЕЛЬ РАБОТЫ

Изучить концепции ролевого управления доступом и способами защиты программного обеспечения от существующих угроз. Реализовать программное средство, которое использует ролевое управление доступом для разграничения полномочий пользователей. Получить навыки защиты разработанной программы от несанкционированного копирования и других угроз, которым может подвергаться программное обеспечение.

# 2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

При рассмотрении вопросов безопасности информационных систем практически все авторы выделяют три вида угроз безопасности:

1 Угрозы конфиденциальности информации;

2 Угрозы целостности информации;

3 Угрозы отказа в обслуживании.

Нарушение конфиденциальности возникает тогда, когда к какой-либо информации получает доступ лицо, не имеющее на это права. Этот вид угроз, пожалуй, наиболее часто встречается в реальном мире. Именно для уменьшения подобных угроз рекомендуется хранить в сейфах документы, содержащие секретные сведения, а при работе с такими документами вводить специальные защитные процедуры (допуски, журналы регистрации и т. п.).

Нарушение целостности происходит при внесении умышленных или неумышленных изменений в информацию. В реальном мире примером нарушения целостности может являться, например, подделка документов. Чтобы избежать этого, используются специальная бумага (с водяными знаками, голограммами и т. д.), печати и подписи. Для заверки подлинности документов существуют нотариальные службы.

Отказ в обслуживании угрожает не самой информации, а автоматизированной системе, в которой эта информация обрабатывается. При возникновении отказа в обслуживании уполномоченные пользователи системы не могут получить своевременный доступ к необходимой информации, хотя имеют на это полное право.

Коммерческие программы обычно защищают от несанкционированного тиражирования.

Наличие доступа только к носителю информации с дистрибутивом (набором инсталляционных файлов) программного продукта не должно давать возможности установить работоспособную копию программы. То есть данных дистрибутива, который можно скопировать или незаметно взять на несколько дней, не должно хватать для создания работоспособной копии программы. Подобные ограничения могут быть реализованы разными способами. Например, очень многие коммерческие программы при инсталляции требуют ввести серийный номер, напечатанный на коробке или указанный в одном из прилагаемых к программному продукту документов (у Microsoft — в сертификате аутентичности).

Также часто возникает потребность ограничить число пользователей, одновременно работающих с программой. То есть человек, который приобрел лицензию на одно рабочее место, не должен иметь возможности создать 2 рабочих места, функционирующих одновременно. Это достигается за счет использования аппаратных ключей, менеджеров лицензий и процедуры активации.

Для некоторых программных продуктов (в частности игр) часто используется привязка к носителю информации, например компакт-диску. То есть для запуска игры требуется наличие в приводе оригинального компакт-диска, который защищен от копирования стандартными средствами.

Для оценочных версий, ограниченных по времени или числу запусков, необходимо правильно реализовать хранение счетчиков, чтобы злоумышленник не смог заставить работать программу, просто переведя часы или удалив файл, в который записывается количество запусков программы или число обработанных файлов.

Условно бесплатные продукты, в отличие от ограниченных по функциональности оценочных версий коммерческих программ, после ввода регистрационного кода должны предоставлять доступ ко всем функциям, предусмотренным в полной версии программы. То есть в бесплатно распространяемой версии программы должны быть реализованы все функции полной версии. Следовательно, желательно так организовать защиту, чтобы злоумышленник не смог добраться до функций, присущих только полной версии, пока в его распоряжении не будет правильного регистрационного кода.

Процедуры проверки правильности серийных номеров, а также регистрационных кодов и кодов активации на самых разных приборах - от защиты информации до электромагниной диагностики deta-elis - должны строиться таким образом, чтобы злоумышленник не мог самостоятельно генерировать правильные коды и, в то же время, длина кодовой строки не была очень большой.

Также может возникнуть потребность защищать любые исполняемые файлы от внесения изменений, дизассемблирования, исследования под отладчиком и т.д.

# 3 ДЕМОНСТРАЦИЯ ПРОГРАММНОГО СРЕДСТВА

В результате выполнения лабораторной работы было получено программное средство, которое использует ролевое управление доступом для разграничения полномочий пользователей и реализует способы защиты программного обеспечения от существующих угроз.

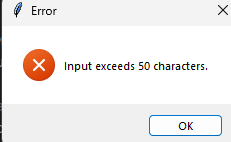
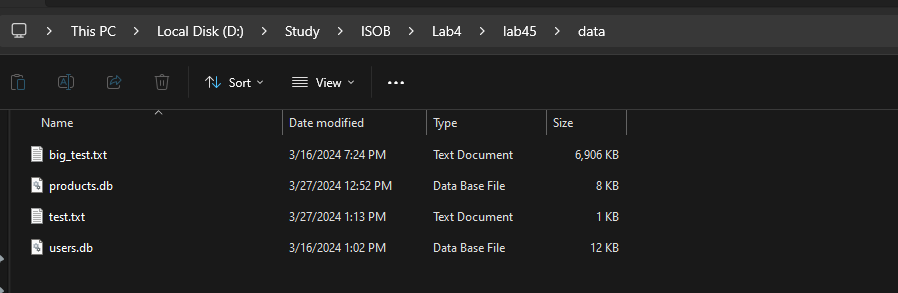


Рисунок 4.1. Способ защиты от атаки «переполнение буфера»



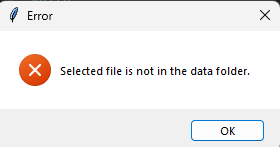


Рисунок 4.2. Способ защиты от атаки, эксплуатирующей ошибки канонизации

Также были реализованы такие защиты от атак как SQL-инъекция и разделение привилегий.

Для реализации приложения-инсталятора использовался PyInstaller. PyInstaller это инструмент для упаковки приложений Python в исполняемые файлы для различных операционных систем.

Исходный код программы представлен в приложении А.

# ПРИЛОЖЕНИЕ А

**Исходный код программы**

import tkinter as tk  
from tkinter import messagebox  
from authentication\_system.admin\_window import AdminWindow  
from authentication\_system.user\_window import UserWindow  
from authentication\_system.database import authenticate, create\_users\_table, create\_products\_table  
  
class MainWindow(tk.Tk):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.title("Authentication System")  
 self.geometry("500x250") # Увеличиваем размер окна  
  
 # Кастомные цвета  
 bg\_color = "#f0f0f0"  
 fg\_color = "#333333"  
 button\_bg = "#4CAF50"  
 button\_fg = "white"  
  
 # Создание и размещение элементов интерфейса  
 self.configure(bg=bg\_color) # Нейтральный фон  
  
 # Центрирование полей ввода  
 self.columnconfigure(1, weight=1)  
  
 self.username\_label = tk.Label(self, text="Username:", bg=bg\_color, fg=fg\_color)  
 self.username\_label.grid(row=0, column=0, padx=(50, 10), pady=10, sticky="e")  
 self.username\_entry = tk.Entry(self, bg=bg\_color, fg=fg\_color, font=("Arial", 12))  
 self.username\_entry.grid(row=0, column=1, padx=(0, 50), pady=10, sticky="ew")  
  
 self.password\_label = tk.Label(self, text="Password:", bg=bg\_color, fg=fg\_color)  
 self.password\_label.grid(row=1, column=0, padx=(50, 10), pady=10, sticky="e")  
 self.password\_entry = tk.Entry(self, show="\*", bg=bg\_color, fg=fg\_color, font=("Arial", 12))  
 self.password\_entry.grid(row=1, column=1, padx=(0, 50), pady=10, sticky="ew")  
  
 self.login\_button = tk.Button(self, text="Login", command=self.login, bg=button\_bg, fg=button\_fg, font=("Arial", 12))  
 self.login\_button.grid(row=2, column=0, columnspan=2, padx=10, pady=10, sticky="ew")  
  
 def login(self):  
 # create\_users\_table()  
 # create\_products\_table()  
 username = self.validate\_input(self.username\_entry.get())  
 password = self.validate\_input(self.password\_entry.get())  
 user = authenticate(username, password)  
 if user:  
 if user[2] == "admin":  
 AdminWindow(self)  
 else:  
 UserWindow(self)  
 # self.destroy() # Закрываем главное окно после успешной аутентификации  
 else:  
 messagebox.showerror("Error", "Invalid username or password.")  
  
 # defense from buffer overflow  
 def validate\_input(self, value):  
 if value is not None and len(value) > 50:  
 return None  
 return value  
  
def main():  
 app = MainWindow()  
 app.mainloop()  
  
main()