#### Задание 1. Обфускация исходного кода

**Исходный скрипт (auth.py):**

python

Copy

Download

def verify\_password(entered\_pwd):

secret = "Admin@2024" *# Пароль в открытом виде - уязвимость*

return entered\_pwd == secret

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

pwd\_attempt = input("Введите пароль: ")

print("Доступ разрешен!" if verify\_password(pwd\_attempt) else "Отказано!")

**Практические шаги:**

1. Установите инструмент обфускации:

bash

Copy

Download

pip install pyarmor

1. Выполните обфускацию:

bash

Copy

Download

pyarmor gen -O protected auth.py

1. Проверьте результат:
   * В папке protected находится преобразованный код
   * Исходная логика скрыта, но строковые константы могут сохраниться

**Эффективность защиты:**  
✓ Усложняет анализ алгоритма  
✓ Затрудняет прямое копирование кода  
× Критичные данные (пароли) могут остаться доступными

#### Задание 2. Комплексная защита приложения

**Этап 1. Создание исполняемого файла**

1. Установите упаковщик:

bash

Copy

Download

pip install pyinstaller

1. Соберите защищенный exe-файл:

bash

Copy

Download

pyinstaller --onefile --key=SecureKey123 auth.py

**Этап 2. Дополнительное шифрование**

python

Copy

Download

from cryptography.fernet import Fernet

*# Генерация и сохранение ключа должно быть отдельно!*

encryption\_key = Fernet.generate\_key()

crypto\_handler = Fernet(encryption\_key)

protected\_password = crypto\_handler.encrypt(b"Admin@2024")

def verify\_password(user\_input):

actual\_pwd = crypto\_handler.decrypt(protected\_password).decode()

return user\_input == actual\_pwd

**Проверка устойчивости:**

1. Попытка декомпиляции:

bash

Copy

Download

uncompyle6 auth.exe → Вероятно не удастся

1. Анализ в Ghidra:
   * Байт-код зашифрован
   * Прямое чтение строк затруднено
   * Логику можно восстановить при достаточном усилии

**Рекомендации по усилению защиты:**

1. Комбинируйте методы:
   * Обфускация + упаковка + шифрование
2. Безопасное хранение ключей:
   * Используйте переменные окружения
   * Применяйте аппаратные ключи (HSM)
3. Дополнительные меры:
   * Проверка на отладчик
   * Детектирование виртуальных сред
   * Контроль целостности кода

**Важное замечание:**  
Абсолютной защиты не существует, но комплексный подход значительно повышает уровень безопасности и затрудняет взлом. Для критически важных систем рекомендуется использовать языки со статической компиляцией (C++, Rust) и дополнительные системы защиты.