**Задание 1. Сокрытие исходного кода (обфускация)**

**Исходный код (auth.py):**

python

Copy

Download

def verify\_password(entered\_pwd):

secret = "Admin@2024" *# Хранение пароля в открытом виде — небезопасно*

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. **Установка инструмента для обфускации:**

bash

Copy

Download

pip install pyarmor

1. **Запуск обфускации:**

bash

Copy

Download

pyarmor gen -O protected auth.py

1. **Результат обработки:**
   * В каталоге protected будет модифицированный код
   * Логика работы скрыта, но строки могут остаться читаемыми

**Эффективность защиты:**  
✔ Затрудняет анализ работы программы  
✔ Усложняет копирование функционала  
✖ Конфиденциальные данные (пароли) могут быть извлечены

**Задание 2. Многоуровневая защита приложения**

**Шаг 1. Генерация исполняемого файла**

1. **Инсталляция утилиты для сборки:**

bash

Copy

Download

pip install pyinstaller

1. **Создание защищенного исполняемого файла:**

bash

Copy

Download

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

**Шаг 2. Применение криптографии**

python

Copy

Download

from cryptography.fernet import Fernet

*# Ключ шифрования должен генерироваться отдельно!*

encryption\_key = Fernet.generate\_key()

cipher = Fernet(encryption\_key)

encrypted\_password = cipher.encrypt(b"Admin@2024")

def verify\_password(user\_input):

real\_password = cipher.decrypt(encrypted\_password).decode()

return user\_input == real\_password

**Тестирование защищенности:**

1. **Попытка обратной разработки:**

bash

Copy

Download

uncompyle6 auth.exe *# Скорее всего, завершится ошибкой*

1. **Исследование в Ghidra:**
   * Исполняемый код зашифрован
   * Извлечение строк требует значительных усилий
   * Основная логика может быть восстановлена при длительном анализе

**Советы по улучшению защиты:**

1. **Использование нескольких методов:**
   * Обфускация + компиляция + шифрование
2. **Надежное хранение ключей:**
   * Переменные окружения
   * Аппаратные модули безопасности (HSM)
3. **Дополнительные технологии защиты:**
   * Антиотладочные механизмы
   * Обнаружение виртуальных машин
   * Проверка целостности исполняемого файла

**Ключевой момент:**  
Полной защиты достичь невозможно, но комплекс мер существенно повышает безопасность и увеличивает сложность взлома. Для систем с повышенными требованиями к защите предпочтительнее использовать компилируемые языки (C++, Rust) с дополнительными средствами защиты на уровне исполняемого кода.