**Задача 1: Обфускация кода с PyArmor**

**Исходный код (auth.py)**

python

Copy

Download

def check\_password(input\_pwd):

password = "Admin@2024" *# Уязвимая строка (хранится в открытом виде)*

return input\_pwd == password

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

user\_input = input("Password: ")

print("Access granted!" if check\_password(user\_input) else "Denied!")

**Задание**

1. **Установите PyArmor** (инструмент для обфускации Python-кода):

bash

Copy

Download

pip install pyarmor

1. **Обфусцируйте код** (преобразует в нечитаемый вид):

bash

Copy

Download

pyarmor gen -O dist auth.py

1. **Проверьте результат** в папке dist:
   * Исходный код заменён на зашифрованный байт-код.
   * Основная логика скрыта, но строки могут остаться в бинарном виде.

**Что это даёт?**

Затрудняет анализ кода (но не делает невозможным).  
 Скрывает логику, но **строки могут быть извлечены** (например, "Admin@2024").

**Задача 2: Упаковка + Шифрование**

**Шаг 1: Создание запакованного EXE-файла (PyInstaller)**

1. **Установите PyInstaller**:

bash

Copy

Download

pip install pyinstaller

1. **Соберите исполняемый файл с ключом шифрования**:

bash

Copy

Download

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

* + --onefile – создаёт один EXE-файл.
  + --key=... – шифрует байт-код (затрудняет декомпиляцию).

**Проверка защиты**

1. **Попробуйте извлечь код с uncompyle6**:

bash

Copy

Download

pip install uncompyle6

uncompyle6 auth.exe *# Вероятно, не сработает из-за шифрования*

1. **Проанализируйте в Ghidra/IDA Pro**:
   * Байт-код зашифрован, но строки могут быть видны.
   * Логику восстановить сложнее, но **не невозможно**.

**Шаг 2: Дополнительное шифрование строк**

python

Copy

Download

from cryptography.fernet import Fernet

*# Генерация ключа (лучше хранить отдельно!)*

key = Fernet.generate\_key()

cipher = Fernet(key)

enc\_password = cipher.encrypt(b"Admin@2024") *# Шифруем пароль*

def check\_password(input\_pwd):

real\_pwd = cipher.decrypt(enc\_password).decode() *# Расшифровка при проверке*

return input\_pwd == real\_pwd

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

user\_input = input("Password: ")

print("Access granted!" if check\_password(user\_input) else "Denied!")

**Что это даёт?**

**Пароль не хранится в открытом виде** (только в зашифрованном).  
 Даже если злоумышленник извлечёт auth.exe, он не увидит пароль напрямую.  
 **Но:**

* Ключ шифрования (key) всё равно где-то должен храниться.
* Если злоумышленник найдёт ключ, он сможет расшифровать пароль.

**Вывод: Как улучшить защиту?**

1. **Используйте обфускацию + упаковку** (PyArmor + PyInstaller).
2. **Шифруйте критичные строки** (пароли, API-ключи).
3. **Храните ключи отдельно** (например, в переменных окружения).
4. **Добавьте антиотладочные техники** (проверка на debugger, виртуальную машину).

**Важно:** 100% защиты не существует, но можно сильно усложнить анализ!