**Задача 3: Проверка лицензии**

**Исходный код с комментариями**

python

Copy

Download

import os *# Импорт модуля для работы с файловой системой*

def check\_license():

*# Проверка существования файла лицензии*

if not os.path.exists("license.key"):

print("Файл лицензии не найден!")

return False *# Возвращаем False, если файла нет*

*# Чтение файла лицензии*

with open("license.key") as f:

key = f.read().strip() *# Удаляем лишние пробелы и переносы строк*

return key == "ABCD-1234-EFGH" *# Сравнение с эталонным ключом*

def main():

*# Проверка лицензии*

if not check\_license():

print("Программа заблокирована!")

return *# Выход, если лицензия недействительна*

*# Основной функционал программы*

print("Программа запущена!")

*# Здесь мог бы быть основной код приложения*

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

main() *# Запуск программы*

**Тестирование программы**

| **Сценарий** | **Результат** | **Вывод** |
| --- | --- | --- |
| **Нет файла license.key** | Файл лицензии не найден! Программа заблокирована! | Лицензия не обнаружена, доступ запрещён |
| **Файл есть, ключ верный** (ABCD-1234-EFGH) | Программа запущена! | Лицензия действительна |
| **Файл есть, ключ неверный** (XXXX-0000-YYYY) | Программа заблокирована! | Лицензия недействительна |

**Задача 4: Вопросы по защите ПО**

**1. Основные цели защиты ПО**

* Затруднение реверс-инжиниринга
* Предотвращение нелегального копирования
* Защита интеллектуальной собственности

**2. Важность защиты для бизнеса**

* Сохранение конкурентных преимуществ
* Предотвращение утечек данных
* Соблюдение лицензионных соглашений

**3. Риски при отсутствии защиты**

* Утечки алгоритмов и данных
* Внедрение вредоносного кода
* Финансовые потери из-за пиратства

**4. Реверс-инжиниринг и угрозы**

* **Что это?** Анализ бинарного кода для восстановления логики.
* **Угрозы:** Кража технологий, создание эксплойтов.

**5. Методы реверс-инжиниринга**

* Статический анализ (Ghidra, IDA Pro)
* Динамический анализ (отладчики: x64dbg, OllyDbg)

**6. Уязвимости в незащищённом ПО**

* Жёстко закодированные пароли (password = "12345")
* Отсутствие проверки лицензий
* Уязвимости переполнения буфера

**7. Популярные методы защиты**

* Обфускация кода (PyArmor, Obfuscator-LLVM)
* Шифрование строк и алгоритмов
* Лицензирование (HWID, онлайн-активация)

**8. Шифрование данных**

* **Симметричное** (AES, ключ один)
* **Асимметричное** (RSA, пара ключей)

**9. Обфускация кода**

* Преобразует код в трудночитаемый вид.
* **Примеры инструментов:** PyArmor (Python), ProGuard (Java).

**10. Лицензирование для защиты**

* **Типы лицензий:**
  + Оффлайн (файл license.key)
  + Онлайн (серверная проверка)
  + Привязка к железу (HWID)

**11. Тестирование защиты**

* Попытка декомпиляции (uncompyle6, Ghidra)
* Фаззинг-тесты (ввод неверных данных)

**12. Аудит безопасности**

* Выявление уязвимостей до релиза.
* **Инструменты:** SonarQube, Checkmarx.

**13. Автоматизированное тестирование**

* **Пример:** CI/CD-пайплайны с проверкой уязвимостей.

**14. Успешные практики защиты**

* **Denuvo** (защита игр от пиратства)
* **Spotify** (онлайн-лицензирование)

**15. Адаптация защиты под тип ПО**

* **Мобильные приложения:** Защита от модификации (Frida, Xposed)
* **Десктоп:** Антиотладочные техники

**16. Примеры взломов**

* Взлом игры **Cyberpunk 2077** (незащищённые скрипты)
* Утечка исходников **Windows XP** (2004 г.)

**17. Машинное обучение в защите**

* Детекция аномалий в работе ПО.

**18. Юридические аспекты**

* DMCA (США) – запрет на обход защиты.
* GDPR (ЕС) – защита пользовательских данных.

**19. Новые тренды**

* **RASP** (Runtime Application Self-Protection)
* **Контейнеризация** (изоляция кода)

**20. Влияние IoT и AI**

* **Уязвимости IoT:** Слабые стандарты шифрования.
* **AI-риски:** Подделка лицензионных ключей нейросетями.

**Итог:**

* Базовая проверка лицензии уязвима к модификации файлов.
* Для надёжной защиты нужен комплексный подход (обфускация + шифрование + серверная проверка).
* Регулярный аудит безопасности критически важен.