Отчёт №14

Задание 1:

1. import os#импорт библеотек
2. def check\_license(): #Создаётся функция check\_license(), которая будет проверять наличие и содержимое файла лицензии.
3. if not os.path.exists("license.key"):#Проверяется: существует ли файл с названием в текущем каталоге."license.key
4. print("Файл лицензии не найден!")
5. return False#Функция завершает работу и возвращает значение , которое означает: лицензия не прошла проверку.False
7. with open("license.key") as f:
8. key = f.read().strip()
9. return key == "ABCD-1234-EFGH"
10. def main():Определяется основная функция программы — . В ней реализована логика запуска или блокировки программы.main()
11. if not check\_license():
12. print("Программа заблокирована!")
13. return
15. print("Программа запущена!")#Программа проходит проверку и запускается
16. *# Основной код*
17. if \_\_name\_\_ == "\_\_main\_\_":
18. main()

№3

1.Строка 4: возвращает , так как файла нет. os.path.exists("license.key")False

Строка 5: Выводится сообщение об отсутствии файла.

Строка 6: возврат функции . check\_license()False

Строка 13: Условие становится истинным. if not check\_license()

Строка 14: Выводится сообщение о блокировке программы.

Строка 15: Программа завершает работу.

2.Строка 4: возвращает . os.path.exists("license.key")True

Строка 8: Файл открыт для чтения.

Строка 9: Соdе файл читается в переменную держимую (после сохранения ). key.strip()"ABCD-1234-EFGH"

Строка 10: Сравнение возвращает . key == "ABCD-1234-EFGH"True

Строка 13: Условие становится ложным. if not check\_license()

Строка 17: Программа выводит сообщение о запуске.

3.Строка 4: возвращает . os.path.exists("license.key")True

Строка 8: Файл открыт для чтения.

Строка 9: Соое файл читается в переменную держимую (после сохранения ). key.strip()"WRONG-KEY-1234"

Строка 10: Сравнение возвращает . key == "ABCD-1234-EFGH"False

Строка 13: Условие становится истинным. if not check\_license()

Строка 14: Выводится сообщение о блокировке программы.

Строка 15: Программа завершает работу.

Задание 2:

Ответы на вопросы:

1. Основные цели защиты программного обеспечения от исследования:

- Предотвращение несанкционированного доступа к исходному коду.

- Защита интеллектуальной собственности (алгоритмы, бизнес-логика).

- Предотвращение взлома, кражи данных или модификации программы.

2. Почему защита программного обеспечения важна для бизнеса?

- Предотвращает утечки коммерческих данных.

- Сохраняет конкурентоспособность продукта.

- Снижает риски финансовых потерь из-за кибератак.

3. Риски при отсутствии защиты программ:

- Кража исходного кода и его использование конкурентами.

- Внедрение вредоносного кода (например, троянов).

- Утечки персональных данных пользователей.

4. Что такое реверс-инжиниринг и его угрозы?

Реверс-инжиниринг — анализ программы для восстановления её исходного кода или логики.

Угрозы:

- Восстановление секретных алгоритмов.

- Обход защиты (например, кряки, патчи).

- Выявление уязвимостей для последующего взлома.

5. Основные методы реверс-инжиниринга:

- Дизассемблирование (преобразование машинного кода в ассемблер).

- Декомпиляция (восстановление исходного кода из байт-кода).

- Отладка (пошаговое выполнение программы для анализа поведения).

- Инструменты: IDA Pro, Ghidra, OllyDbg.

6. Уязвимости в недостаточно защищенных программах:

- Незащищённые API-ключи или пароли в коде.

- Логические ошибки (например, проверки лицензии, которые можно обойти).

- Утечки памяти или уязвимости ввода/вывода (SQL-инъекции, переполнение буфера).

7. Распространённые методы защиты программного обеспечения:

- Обфускация кода (затруднение чтения).

- Шифрование (данных и частей кода).

- Контроль целостности (проверка, не изменён ли код).

- Лицензирование (ключи, онлайн-активация).

- Использование антитампер-механизмов.

8. Как работает шифрование данных и его виды:

Принцип: Преобразование данных в нечитаемый формат с помощью ключей.

Виды:

- Симметричное (AES, DES — один ключ для шифрования/дешифрования).

- Асимметричное (RSA — открытый и закрытый ключ).

- Хэширование (SHA-256 — для проверки целостности данных).

9. Что такое обфускация кода и как она помогает?

Что это? Преобразование кода в трудночитаемый, но функционально эквивалентный вариант.

Как помогает:

- Затрудняет анализ кода.

- Скрывает логику работы программы.

- Пример: переименование переменных в `a`, `b`, `c`.

10. Инструменты для обфускации кода:

- Python: PyArmor, Cython (компиляция в C).

- Java: ProGuard, Allatori.

- .NET: Dotfuscator.

- JavaScript: JavaScript Obfuscator.

11. Как использовать лицензирование для защиты программного обеспечения?

- Ключевые файлы (например, license.key).

- Онлайн-активация (проверка на сервере).

- Токены безопасности (например, аппаратные ключи).

- Ограничение по времени (пробные версии).

12. Эффективные типы лицензий:

- Перпетуальные (пожизненная активация).

- Подписка (ежемесячная/годовая оплата).

- Пробные (ограниченный срок или функционал).

- Node-locked (привязка к конкретному устройству).

13. Как тестировать защитные механизмы программы на уязвимости?

- Пентестинг (имитация атак).

- Статический анализ (поиск уязвимостей в коде).

- Динамическое тестирование (анализ работы программы).

- Использование инструментов: Burp Suite, OWASP ZAP.

14. Важность регулярных аудитов безопасности:

- Выявление новых уязвимостей.

- Проверка эффективности текущих методов защиты.

- Соответствие стандартам (например, GDPR, ISO 27001).

15. Как автоматизированное тестирование помогает в оценке защиты?

- SAST (статический анализ кода).

- DAST (динамический анализ работающего приложения).

- Инструменты: SonarQube, Checkmarx, Selenium (для UI-тестов).

16. Примеры успешных практик защиты ПО:

- Игры: Anti-cheat системы (например, Easy Anti-Cheat).

- Облачные сервисы: Шифрование данных на стороне клиента (Tresorit).

- Банковские приложения: Двухфакторная аутентификация, обфускация.

17. Как адаптировать методы защиты под тип ПО?

- Мобильные приложения: Обфускация, проверка root/jailbreak.

- Веб-сервисы: Защита API, шифрование трафика (HTTPS).

- IoT-устройства: Аппаратная защита (TPM), обновления прошивки.

18. Известные случаи взлома ПО:

- WannaCry (взлом Windows через уязвимость EternalBlue).

- Equifax (утечка данных из-за непатченной библиотеки Apache Struts).

- Colonial Pipeline (взлом через уязвимость в VPN).

19. Как машинное обучение улучшает защиту ПО?

- Обнаружение аномалий (например, необычное поведение программы).

- Автоматическая классификация вредоносного кода.

- Генерация обфусцированного кода с помощью ИИ.

20. Юридические аспекты защиты ПО:

- Авторское право (защита исходного кода).

- Патенты (на уникальные алгоритмы).

- NDA (неразглашение коммерческой информации).

- GDPR (защита персональных данных).

21. Современные тренды в защите ПО:

- Обфускация на уровне байт-кода (например, LLVM).

- Облачные проверки (например, лицензия проверяется на сервере).

- Использование WebAssembly для защиты логики в браузере.

22. Как развитие технологий влияет на защиту от реверс-инжиниринга?

- Аппаратные механизмы (например, Intel SGX для изолированных вычислений).

- Многоуровневая защита (шифрование + обфускация + контроль целостности).

- Использование JIT-компиляции (динамическое изменение кода во время выполнения).

23. Уязвимости IoT и AI в программном обеспечении:

- IoT: Слабые пароли по умолчанию, незащищённые API.

- AI: Манипуляции с данными для обучения (например, отравление датасетов).

- Общие риски: Утечки данных через «умные» устройства, взлом голосовых помощников.