## 1. Основные цели защиты ПО от исследования

* Защита интеллектуальной собственности (алгоритмы, логика)
* Предотвращение несанкционированного доступа/модификации
* Борьба с пиратством и нелегальным копированием
* Сохранение конкурентных преимуществ

## 2. Важность защиты ПО для бизнеса

* Защита инвестиций в разработку (Python-проекты часто содержат уникальные алгоритмы)
* Сохранение доходов от лицензирования
* Предотвращение утечки чувствительных данных (например, в Django-приложениях)
* Поддержание репутации компании

## 3. Риски при отсутствии защиты

* Утечка исходного кода (.py файлов)
* Модификация поведения программы (инъекция кода через exec() или pickle)
* Кража лицензионных ключей (как в примере с license.key)
* Обход ограничений (декомпиляция .pyc файлов)

## 4. Реверс-инжиниринг и угрозы

* Анализ bytecode (.pyc) инструментами типа uncompyle6
* Декомпиляция в исходный код
* Поиск уязвимостей в алгоритмах проверки лицензий
* Извлечение секретов (API-ключи, пароли)

## 5. Методы реверс-инжиниринга

* Декомпиляция (для Python: uncompyle6, decompyle3)
* Динамический анализ (отладка с pdb)
* Анализ сетевого трафика (для веб-приложений)
* Дамп памяти процесса

## 6. Уязвимости в незащищенном ПО

* Хардкод секретов (API keys в коде)
* Небезопасная десериализация (pickle)
* Слабые проверки лицензий (как в примере)
* Уязвимости в зависимостях (requirements.txt)

## 7. Распространенные методы защиты

* Обфускация кода (pyarmor, cython)
* Шифрование чувствительных данных
* Нативные расширения (C-модули)
* Серверная валидация лицензий

## 8. Шифрование данных

* Симметричное (AES): cryptography library
* Асимметричное (RSA): rsa package
* Хеширование (для паролей): bcrypt, argon2

## 9. Обфускация кода

* Преобразование кода в трудночитаемую форму
* Инструменты: pyarmor, pyminifier

## 10. Инструменты обфускации

* pyarmor (коммерческий)
* pyminifier (удаление комментариев, минификация)
* cython (компиляция в C-расширения)
* nuitka (компиляция в бинарники)

## 11. Лицензирование для защиты

* Проверка лицензий на сервере
* Привязка к оборудованию (MAC-адрес)
* Временные лицензии (срок действия)
* Цифровые подписи лицензий

## 12. Эффективные типы лицензий

* Плавающие (серверные) лицензии
* Подписки с онлайн-проверкой
* Лицензии с аппаратной привязкой
* Токены (JWT) с подписью

## 13. Тестирование защитных механизмов

* Фаззинг-тесты (hypothesis)
* Атаки на лицензионные проверки
* Анализ зависимостей (safety, bandit)
* Пентесты (например, попытки обхода аутентификации)

## 14. Важность аудитов безопасности

* **Обнаружение уязвимостей** – выявление проблем в коде (SQL-инъекции, небезопасный pickle, хардкод секретов)
* **Проверка зависимостей** – анализ requirements.txt на уязвимые библиотеки (pip-audit)
* **Тестирование защиты** – оценка обфускации, лицензионных проверок и шифрования данных
* **Соответствие стандартам** – проверка на соблюдение PEP-8 и GDPR
* **Профилактика атак** – предупреждение реверс-инжиниринга (декомпиляция .pyc)

## 15. Автоматизированное тестирование защиты

* Интеграция в CI/CD (bandit, safety)
* Тесты на уязвимости (OWASP ZAP для веб)
* Мониторинг зависимостей (dependabot)
* Fuzzing-тесты для API

## 16. Примеры успешных практик

* Использование cryptography вместо самописного шифрования
* Серверная валидация лицензий (не только клиентская)
* Регулярное обновление зависимостей
* Многофакторная аутентификация в веб-API

## 17. Адаптация методов защиты

* Веб-приложения: CSRF-токены, CORS
* Десктоп: обфускация + нативные расширения
* Мобильные: привязка к устройству
* Микросервисы: JWT с коротким временем жизни

## 18. Известные случаи взлома

* Декомпиляция Python-игр (например, Ren'Py)
* Кража API-ключей из открытых репозиториев
* Инъекции через pickle в ML-моделях
* Обход лицензий в популярных библиотеках

## 19. ML в защите ПО

* Аномалии в использовании (выявление взлома)
* Генерация обфусцированного кода
* Анализ паттернов атак
* Динамическое обновление правил защиты

## 20. Юридические аспекты

* Лицензии GPL для открытого ПО
* Патентование алгоритмов
* DMCA против декомпиляции
* GDPR для защиты данных

## 21. Новые тренды защиты

* Runtime Application Self-Protection (RASP)
* Zero Trust Architecture
* Контейнеризация с ограничениями
* Аппаратные ключи безопасности

## 22. Влияние технологий на защиту

* Квантовые вычисления → постквантовое шифрование
* WASM для защиты веб-кода
* eBPF для мониторинга в реальном времени
* Угрозы от ИИ-ассистированного реверс-инжиниринга

## 23. Уязвимости IoT и AI

* Небезопасные API в IoT-устройствах
* Атаки на ML-модели (adversarial examples)
* Утечки данных через сенсоры
* Уязвимости в edge-вычислениях