|  |  |  |
| --- | --- | --- |
| Действие | eicar.com | game.exe |
| Создание файлов | Нет | Да (может создавать копии себя, другие вредоносные файлы) |
| Изменение реестра | Нет | Да (может прописываться в автозагрузку, менять настройки системы) |
| Сетевые запросы | Нет | Да (может связываться с C&C сервером, загружать другие модули, участвовать в ботнете) |

Задача 2

1. Признаки вредоносного поведения

* Создание скрытых файлов (в %Temp%, %AppData%, системных директориях).
* Изменение реестра: автозагрузка (HKLM\Software\Microsoft\Windows\CurrentVersion\Run), отключение безопасности.
* Сетевая активность: подключение к подозрительным IP/доменам (C&C-серверы), передача данных.
* Шифрование/удаление файлов (ransomware).
* Сокрытие процессов: инжектирование в легитимные процессы (svchost.exe, explorer.exe).
* Сбор данных: логирование клавиш, скриншоты, кража cookies.

2. Почему EICAR безопасен для анализа?

* Не исполняет код: Это текстовый файл со строкой X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR..., которую антивирусы распознают как тестовый сигнал.
* Нет реальных действий: Не создает файлы, не меняет реестр, не взаимодействует с сетью.
* Создан для тестирования: Цель — проверить реакцию антивируса (карантин/детект).

3. Как FakeNet помогает анализировать сетевую активность?

FakeNet эмулирует сетевые сервисы, чтобы:

* Перехватывать запросы: Показывает HTTP/DNS/FTP-трафик, даже если C&C-сервер нерабочий.
* Имитировать ответы: Подменяет ответы серверов (например, отправляет "успешную" заглушку на запрос вредоноса).
* Раскрывать payloadы: Позволяет скачать дополнительные вредоносные модули, которые запрашивает образец.
* Определять назначение трафика: Анализирует, какие данные передаются (конфиги, украденная информация).

4. Что такое ВПО и его основные типы

* Вредоносное ПО (ВПО) — программы, созданные для нанесения ущерба, кражи данных или контроля над системой без согласия пользователя.
* Типы: Троян, Рансомвер,Червь,Шпионское ПО

5. Основные цели анализа вредоносного ПО

1. Идентификация угрозы: Определить тип ВПО (троян, ransomware и т.д.).
2. Понимание функционала: Выяснить, что делает образец (кража данных, шифрование, бот-сеть).
3. Извлечение IOC-индикаторов: IP-адреса C&C, домены, хеши файлов, сигнатуры для блокировки.
4. Разработка контрмер: Создание правил для IDS/IPS, сигнатур для антивирусов.
5. Расследование инцидентов: Установить источник атаки, масштаб заражения.
6. Прогнозирование: Анализ трендов (например, новые техники уклонения от детектирования).

6. Разница между статическим и динамическим анализом

|  |  |  |
| --- | --- | --- |
| Критерий | Статический анализ | Динамический анализ |
| Суть | Исследование без запуска кода. | Исследование во время выполнения. |
| Методы | Дизассемблирование, декомпиляция, изучение строк, импортов. | Мониторинг в песочнице: реестр, файлы, сеть, API-вызовы. |
| Преимущества | Безопасен, выявляет скрытые функции. | Показывает реальное поведение, обходит обфускацию. |
| Недостатки | Сложность с обфусцированным кодом. | Риск заражения; требует изоляции (песочница). |
| Инструменты | IDA Pro, Ghidra, PEStudio. | FakeNet, Process Monitor, Wireshark, Cuckoo Sandbox. |

7. Инструменты статического анализа и их функции

IDA Pro: Дизассемблирование, декомпиляция, анализ графов вызовов, кросс-ссылки, плагины (Hex-Rays)

Ghidra: Бесплатное дизассемблирование/декомпиляция, анализ потоков данных, скриптинг на Python

PEStudio: Анализ заголовков PE-файлов, импортов DLL, строк, цифровых подписей, IOC-индикаторов

CFF Explorer: Просмотр и редактирование структур PE-файлов, ресурсов, манифестов

8. Обфускация кода во вредоносных программах

* Что это: Намеренное усложнение кода для затруднения анализа.
* Методы:
  + Шифрование/пакинг: Код распаковывается только при запуске (UPX, VMProtect).
  + Полиморфизм: Автоматическое изменение кода при каждом заражении.
  + Мусорные инструкции: Добавление "пустых" операций (nop, push/pop).
  + Сплит-строки: Разделение строк на части с последующей сборкой в памяти.

9. Признаки вредоносного кода в ПО

* Подозрительные импорты DLL:  
  Kernel32 (доступ к процессам), User32 (кейлоггинг), Wininet (сеть).
* Отсутствие цифровой подписи или поддельный сертификат.
* Странные строки: IP-адреса, домены, пути вроде C:\Windows\Temp\svchost.exe.
* Анти-отладочные техники: Проверка на наличие отладчика (IsDebuggerPresent).

10. Песочница (sandbox) в анализе вредоносного ПО

* Что это: Изолированная среда для безопасного запуска подозрительных программ.
* Преимущества:
  + Автоматический мониторинг: файлы, реестр, сеть, API-вызовы.
  + Снимки состояния системы до/после выполнения.
  + Обход рутинных задач (Cuckoo Sandbox, ANY.RUN).

11. Методы обхода антивирусов

Метаморфизм: Полная пересборка кода с сохранением функционала

Память-резидентность: Инжектирование кода в процессы (explorer.exe) без записи на диск

12. Инъекция кода (Code Injection)

* Что это: Внедрение вредоносного кода в память легитимного процесса.
* Техники:
  + DLL Injection: Принудительная загрузка DLL в целевой процесс (CreateRemoteThread).
  + Process Hollowing: Замена кода легитимного процесса (например, svchost.exe) вредоносным.

13. Этапы динамического анализа

1. Подготовка:
   * Настройка изолированной VM (без общих папок).
   * Установка инструментов: Process Monitor, Wireshark, Process Hacker.
2. Запуск образца:
   * Фиксация процессов, файловых операций, изменений реестра.
3. Мониторинг сети:
   * Перехват DNS/HTTP-запросов через FakeNet.
4. Анализ дампов памяти:
   * Использование Volatility для поиска инжектированного кода.
5. Очистка: Сброс VM в исходное состояние.

14. Данные статического анализа

* Заголовки файлов: Тип (PE, ELF), точки входа, секции.
* Импорты API: WriteProcessMemory, RegSetValue, URLDownloadToFile.
* Строки: IP-адреса, пароли, пути, C&C-команды.
* Ресурсы: Скрытые файлы (дропперы), иконки, конфиги.
* Сетевые сигнатуры: URL, домены, порты.

15. Польза изучения поведения вредоносов

* Проактивная защита:
  + Создание YARA-правил по IOC.
  + Настройка IDS/IPS для блокировки C&C-трафика.
* Понимание TTPs (Tactics, Techniques, Procedures):
  + Например: использование schtasks.exe для персистентности → мониторинг задач Windows.
* Обучение ИИ-систем: Базы для алгоритмов ML в EDR-решениях.

16. Сценарии использования вредоносного ПО

* Кража данных: Логины, банковские реквизиты, документы (шпионское ПО).
* Шантаж: Шифрование файлов (ransomware).
* Ботнеты: DDoS-атаки, рассылка спама.
* Криптомайнинг: Скрытный майнинг на зараженных ПК.
* Саботаж: Уничтожение данных (wiper-модули).

17. Обратная разработка (Reverse Engineering)

* Что это: Анализ бинарного кода для восстановления логики программы без исходников.
* Применение в анализе:
  + Расшифровка payload (например, вирусных конфигов).
  + Поиск уязвимостей в эксплоитах.
  + Изучение алгоритмов C&C-коммуникации.

18. Риски анализа в реальной среде

* Заражение сети: Утечка через общие ресурсы VM.
* Срабатывание реальных payload: Шифрование дисков, атаки на другие системы.
* Юридические последствия: Если образец содержит конфиденциальные данные жертв.
* Утечка IOC: Передача данных на C&C во время анализа.

19. Меры предосторожности

* Изоляция: Анализ в VM без NAT (только host-only сеть).
* Инструменты:
  + REMnux (спецдистрибутив для анализа).
  + Физическая "ловушка" сети (например, Raspberry Pi с Wireshark).
* Процедуры:
  + Отключение общих папок в VM.
  + Использование временных аккаунтов без привилегий.
  + Регулярные снимки состояния.

20. Польза анализа в расследованиях кибератак

* Атрибуция: Стиль кода → хакерская группа (например, Lazarus Group).
* Поиск компрометации:
  + Извлечение IOC (IP, домены, хеши) для поиска в логах.
  + Определение вектора атаки: фишинг, уязвимость в ПО.
* Восстановление: Для ransomware — поиск ключей дешифрации в памяти.

21. Этические аспекты

* Контроль образцов: Использовать только легально полученные образцы (не из взломанных систем).
* Ответственное разглашение: Сообщать авторам о уязвимостях перед публикацией (если ПО легитимно).
* Конфиденциальность: Не анализировать данные жертв (фото, документы) из вредоноса.
* Compliance: Следовать законам (например, GDPR при работе с персональными данными).