Отчет

Тема: Применение отладчиков и дизассемблера

1. Основные понятия

1. Что такое отладчик и для чего он используется?

Отладчик (debugger) — это инструмент, позволяющий выполнять программу пошагово, ставить точки останова и анализировать состояние программы во время выполнения.

Используется:

- Для поиска ошибок.

- Анализа логики работы программы.

- Понимания поведения кода при определенных входных данных.

2. В чем разница между отладчиком и дизассемблером?

- Отладчик работает с исполняемым кодом, позволяет управлять его выполнением, просматривать переменные, регистры, память.

- Дизассемблер преобразует машинный код в ассемблерную инструкцию, но не запускает программу. Используется для анализа без исходников.

3. Какие типы ошибок можно найти с помощью отладчика?

- Логические ошибки (неправильная последовательность действий).

- Ошибки доступа к памяти (NULL-поинтеры, выход за границы массива).

- Деление на ноль.

- Бесконечные циклы.

- Неверные значения переменных.

4. Что такое точка останова (breakpoint)?

Точка останова (breakpoint) — это маркер, который выставляется в определённой строке кода или адресе памяти. Программа останавливается перед её выполнением, чтобы можно было проанализировать текущее состояние программы.

2. Работа с отладчиком

5. Как запустить программу в режиме отладки в Visual Studio?

- В меню выберите \*\*Debug → Start Debugging\*\* (или F5).

- Перед этим установите точки останова и проверьте конфигурацию проекта (должен быть запущен в режиме Debug).

6. Какие основные команды отладчика вы знаете?

- \*\*Step Into (F11):\*\* Заходит внутрь вызова функции.

- \*\*Step Over (F10):\*\* Выполняет функцию целиком, не заходя внутрь.

- \*\*Continue (F5):\*\* Возобновляет выполнение до следующей точки останова.

7. Как просмотреть значения переменных во время отладки?

- Наведите курсор мыши на переменную в редакторе.

- Используйте окна:

- Locals — показывает локальные переменные текущей области видимости.

- Autos — автоматически показывает используемые переменные.

8. Что такое Watch-окно в отладчике? Как его использовать?

Watch-окно — это инструмент, позволяющий добавлять переменные и выражения для наблюдения за их значениями в реальном времени.

Как использовать:

- Добавьте интересующую переменную через контекстное меню или вручную.

- Можно отслеживать сложные выражения, например: `x + y > 10`.

9. Как отладить программу, если она падает без сообщения об ошибке?

- Запустите программу в отладчике (например, в Visual Studio или GDB).

- Включите "Break on exceptions" (остановка на исключениях).

- Следите за call stack и значениями переменных.

- Проверьте места, где могут быть деление на ноль, NULL-поинтеры, переполнения.

3. Дизассемблирование и анализ кода

10. Что такое дизассемблер и какие задачи он решает?

Дизассемблер — программа, которая преобразует машинный код в человеко-читаемый ассемблерный код.

Решает задачи:

- Обратная разработка.

- Анализ вредоносного ПО.

- Поиск багов в скомпилированных приложениях.

11. Какие популярные дизассемблеры вы знаете?

- Ghidra (разработан NSA) — мощный и бесплатный.

- IDA Pro — коммерческий инструмент, один из лидеров в reverse engineering.

- radare2 — open-source фреймворк для анализа и модификации бинарных файлов.

12. Как найти функцию main в дизассемблированной программе?

- Ищите строки, содержащие `"main"` в списке символов.

- Используйте сигнатуры или вызовы стандартных функций (`\_\_libc\_start\_main` в Linux).

- В Ghidra/IDA: поиск по строкам или графам вызовов.

13. Как дизассемблер помогает найти проверку пароля в программе?

- Ищутся вызовы сравнения строк (`strcmp`, `memcmp`).

- Находят места, где ввод пользователя сравнивается с хардкодным значением.

- Анализируются ветви условий (например, `jz`, `je` после сравнения).

14. Как определить, какие библиотеки использует программа?

- В Windows: используйте \*\*Dependency Walker\*\* или \*\*Process Explorer\*\*.

- В Linux: команда `ldd имя\_программы`.

- В дизассемблерах: откройте секцию \*\*Imports\*\*.

4. Практическое применение

15. Как с помощью отладчика найти ошибку "деление на ноль"?

- Установите точку останова перед операцией деления.

- Проследите значение знаменателя.

- Включите остановку на исключениях (например, Arithmetic Exception).

- Используйте watch-переменные, чтобы отслеживать изменение значения.

5. Обратная разработка (Reverse Engineering)

16. Как восстановить алгоритм программы, если у вас нет исходного кода?

- Используйте дизассемблер для получения ассемблерного кода.

- Применяйте декомпиляторы (например, Ghidra, IDA Pro).

- Анализируйте вызовы функций и данные.

- Строите блок-схемы и псевдокод для понимания логики.

17. Как найти ключевые функции в дизассемблированной программе?

- Ищите вызовы системных функций (например, `ReadFile`, `CreateWindow`).

- Анализируйте строки: часто встречаются в коде ("password", "key", "error").

- Изучайте граф вызовов (call graph), особенно те, что связаны с вводом/выводом.

18. Как определить, использует ли программа шифрование?

- Ищите вызовы известных библиотек шифрования (например, `AES\_encrypt`, `EVP\_CIPHER\_CTX\_new`).

- Анализируйте наличие повторяющихся математических операций (XOR, таблиц замены).

- Смотрите на наличие строк с названиями алгоритмов: `"AES"`, `"RSA"`, `"SHA256"` и т. д.

19. Как проанализировать вредоносное ПО с помощью дизассемблера?

- Откройте файл в Ghidra или IDA Pro.

- Изучите импорты: может быть вызов `CreateRemoteThread`, `VirtualAlloc`, `ConnectSocket`.

- Ищите сетевые и файловые операции.

- Следите за потоком управления, ищите антиотладочные трюки.

6. Дополнительные вопросы

20. В чем разница между статическим и динамическим анализом?

|  |  |
| --- | --- |
| Тип анализа | Описание |
| Статический | Анализ кода без запуска программы (например, через дизассемблер). |
| Динамический | Анализ выполняющейся программы (через отладчик, мониторинг системных вызовов, сетевых запросов). |

21. Какие этические аспекты нужно учитывать при работе с дизассемблерами и отладчиками?

- Авторские права: Анализ чужого ПО может нарушать закон, если нет явного разрешения.

- Безопасность: Анализ вредоносного ПО должен проводиться в изолированной среде (песочнице).

- Цель использования: Эти инструменты должны использоваться только в исследовательских, образовательных или защитных целях, а не для взлома.