Пара № 12

1. *Основание понятия*

1) Отладчик — это инструмент, который используется программистами для поиска и исправления ошибок в программном коде. Он позволяет выполнять программу по шагам, наблюдать за состоянием переменных и памяти, а также контролировать выполнение программы.

2) Отладчик используется для выполнения программы в контролируемом режиме, позволяя разработчикам наблюдать и изменять состояние программы во время её выполнения. Дизассемблер, с другой стороны, преобразует машинный код обратно в ассемблерный код, что позволяет анализировать программу на более низком уровне. Дизассемблеры часто используются для анализа бинарных файлов, когда исходный код недоступен.

3) Синтаксические ошибки в языке программирования.

Логические ошибки в программе, которые приводят к неправильному поведению.

Ошибки времени выполнения , которые возникают во время выполнения программы, такие как деление на ноль или обращение к несуществующей памяти.

Ошибки памяти такие как утечки памяти или обращение к освобожденной памяти.

4) Точка останова - это специальная метка в коде программы, которая указывает отладчику остановить выполнение программы на этом месте. Точки останова позволяют разработчикам приостановить выполнение программы и исследовать её состояние на этом этапе, например, проверить значения переменных или состояние стека вызовов.

2. *Работа с отладчиком*

5) Установите точку останова в коде, щелкнув на левом поле рядом с номером строки, где вы хотите приостановить выполнение. выберите “Отладка” и “Начать отладку” в меню. Это запустит программу в режиме отладки.

6) Step Into - Эта команда позволяет войти в вызов функции или метода, чтобы выполнить его по шагам.

Step Over - Используется для выполнения текущей строки кода и перехода к следующей строке. Если текущая строка содержит вызов функции, эта функция будет выполнена полностью без пошагового выполнения.

Continue - Продолжает выполнение программы до следующей точки останова или до завершения программы.

7) Навести курсор мыши на переменную в коде, чтобы увидеть её текущее значение во всплывающей подсказке.

Использовать окна "Locals" (Локальные переменные) и "Watch" (Наблюдение), которые отображают текущие значения переменных в области видимости.

8) Watch-окно в отладчике позволяет наблюдать за значениями переменных и выражений в процессе отладки.

Откройте его, выбрав " Отладка ", "Windows" (Окна) > " Наблюдение " > "Watch 1" (или другое доступное окно).

9) Убедитесь, что включены все возможные опции логирования и отладки в настройках проекта. Используйте точки останова в потенциально проблемных местах кода. Проверьте системные логи и журналы событий на наличие информации о сбое. Запустите программу в режиме отладки и используйте окна "Call Stack" (Стек вызовов) и "Exception Settings" (Настройки исключений) для анализа состояния программы на момент сбоя.

3. *Дизассемблирование и анализ кода*

10) Дизассемблер — это инструмент, который преобразует машинный код программы обратно в ассемблерный код. Это позволяет анализировать программу на более низком уровне, чем исходный код, и используется в основном для анализа бинарных файлов, когда исходный код недоступен.

11) Ghidra: Бесплатный и открытый инструмент для реверс-инжиниринга, разработанный NSA.

12) Использовать встроенные функции поиска в дизассемблере для поиска по имени функции.

Анализировать точку входа программы и следовать по вызовам функций, так как main часто вызывается из стандартной библиотеки.

Использовать информацию о символах, если она доступна в бинарном файле.

13) Дизассемблер помогает найти проверку пароля в программе, позволяя анализировать ассемблерный код на наличие строковых сравнений или вызовов функций, связанных с обработкой ввода пользователя.

Поиск строк, связанных с паролем, в секции данных программы.

Анализ кода, который сравнивает введенные данные с заранее определенными значениями.

Использование точек останова на функциях сравнения строк для анализа их поведения.

14. Анализировать секции импорта в дизассемблере, которые показывают, какие внешние функции и библиотеки используются программой.

Использовать инструменты анализа бинарных файлов, которые предоставляют информацию о зависимостях и импортируемых функциях.

4. *Практическое применение*

15. 1) Поместите точки останова в тех местах кода, где происходит деление. Это позволит вам приостановить выполнение программы непосредственно перед выполнением операции деления.

2) Запустите программу в отладчике, чтобы начать выполнение с установленными точками останова.

3) Когда выполнение программы приостановится на точке останова, используйте команды пошагового выполнения Step Over или Step Into, чтобы пройти через код, который выполняет деление.

4) Перед выполнением операции деления проверьте значения переменных, участвующих в операции. Убедитесь, что делитель не равен нулю.

5) Добавьте переменные, участвующие в операции деления, в Watch-окно, чтобы наблюдать за их значениями в процессе выполнения программы.

5. *Обратная разработка*

16) Используйте дизассемблер, чтобы преобразовать бинарный код в ассемблерный.

Изучите поток управления программой, чтобы понять, как выполняются различные части кода.

Определите ключевые функции и процедуры в коде.

Изучите, как программа использует данные, включая глобальные переменные, структуры данных и передачу параметров между функциями.

Запустите программу в отладчике, чтобы наблюдать за её поведением в реальном времени. Это может помочь понять, как различные части программы взаимодействуют друг с другом.

17) Если имена функций не были удалены, используйте поиск по имени, чтобы найти функции, которые могут быть ключевыми.

Ищите функции, которые вызываются часто или из ключевых мест программы, таких как точка входа или основные обработчики событий.

Просмотрите строки в программе, так как они могут указывать на важные функции, например, те, которые взаимодействуют с пользователем или выполняют критическую логику.

18) Поиск известных алгоритмов -ищите вызовы функций или последовательности инструкций, которые соответствуют известным алгоритмам шифрования.

Анализ данных - проверьте, как программа обрабатывает данные.

Использование библиотек - проверьте, импортирует ли программа библиотеки, известные для предоставления функций шифрования.

19) Идентификация точки входа - найдите точку входа программы и начните анализ с этого места.

Анализ поведения - используйте отладчик для наблюдения за поведением программы в контролируемой среде, чтобы понять, что она делает.

Анализ строк - просмотрите строки в программе, так как они могут содержать URL-адреса, команды или сообщения, которые указывают на вредоносное поведение.

Использование песочницы - запускайте программу в изолированной среде (песочнице), чтобы безопасно наблюдать за её поведением.

6. *Дополнительные ответы*

20) Статический анализ проводится без выполнения программы. Он включает изучение исходного кода, бинарных файлов или байт-кода.

Динамический анализ проводится во время выполнения программы. Он включает наблюдение за поведением программы в реальном времени.

21) Соблюдение законов - убедитесь, что ваши действия соответствуют местным и международным законам, касающимся реверс-инжиниринга и анализа программного обеспечения

Уважение прав интеллектуальной собственности - не используйте дизассемблеры и отладчики для нелегального копирования, модификации или распространения программного обеспечения, защищенного авторскими правами.

Конфиденциальность - уважайте конфиденциальность данных и информации, с которыми вы работаете. Не анализируйте программное обеспечение, которое содержит личные или конфиденциальные данные, без соответствующего разрешения.

Использование для защиты - используйте свои навыки и инструменты для улучшения безопасности и защиты программного обеспечения, а не для эксплуатации уязвимостей в вредоносных целях.