**1. Основные понятия**

1. *Что такое отладчик и для чего он используется?*

Отладчик — это инструмент, который позволяет разработчику управлять выполнением программы для поиска и устранения ошибок. Он позволяет останавливать выполнение программы, просматривать состояние переменных, изменять их значения, выполнять код по шагам и выявлять причины сбоев или неправильного поведения.

1. *В чем разница между отладчиком и дизассемблером?*

* **Отладчик** — инструмент для пошагового выполнения программы, работы с исходным кодом или его промежуточным представлением, позволяет контролировать выполнение.
* **Дизассемблер** — программа, которая преобразует исполняемый файл (бинарный код) обратно в ассемблерный код, чтобы понять, что делает программа, без исходных исходников.

1. *Какие типы ошибок можно найти с помощью отладчика?*

* Логические ошибки (например, неправильные вычисления)
* Ошибки времени выполнения (например, деление на ноль, обращения к недопустимым адресам)
* Утечки памяти
* Неправильное использование переменных или функций
* Ошибки, связанные с условиями гонки и многопоточностью

1. *Что такое точка останова (breakpoint)?*

Это команда, которая останавливает выполнение программы в определенной точке, чтобы можно было проанализировать состояние программы, переменные и стек вызовов.

**2. Работа с отладчиком**

1. *Как запустить программу в режиме отладки в Visual Studio?*

* Открыть проект/файл в Visual Studio
* Нажать кнопку **"Start Debugging"** (F5) или выбрать меню **Debug → Start Debugging**
* Можно установить точки останова (breakpoints), чтобы программа остановилась в нужных местах

1. *Какие основные команды отладчика вы знаете (Step Into, Step Over, Continue)?*

* **Step Into (F11)** — войти внутрь вызываемой функции
* **Step Over (F10)** — выполнить текущую строку, не заходя внутрь функций
* **Continue (F5)** — продолжить выполнение до следующего брейкпоинта или окончания программы

1. *Как просмотреть значения переменных во время отладки?*

* В окне **Locals** отображаются локальные переменные текущей функции
* В окне **Autos** — переменные, используемые в текущей строке
* Можно навести курсор на переменную в коде или добавить её в **Watch**-окно

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

Это окно, в которое можно добавить выражения или переменные для отслеживания их значений во время выполнения программы. Помогает следить за изменениями в переменных в разных точках выполнения.

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

* Запустить программу в режиме отладки
* Настроить исключения (Exceptions) — чтобы программа останавливалась при возникновении ошибок
* Использовать просмотр стека вызовов, чтобы понять, где возникла проблема
* Анализировать лог или дамп памяти, если есть

**3. Дизассемблирование и анализ кода**

1. *Что такое дизассемблер и какие задачи он решает?*

Дизассемблер преобразует бинарный код программы в ассемблерный код, чтобы понять логику работы программы без исходных исходников. Он помогает найти уязвимости, понять алгоритм, обнаружить скрытые функции или проверки.

1. *Какие популярные дизассемблеры вы знаете? (Ghidra, IDA Pro, radare2)*

* **Ghidra** — бесплатный, мощный инструмент от NSA
* **IDA Pro** — коммерческий, широко используемый
* **radare2** — открытый и скриптованный инструмент

1. *Как найти функцию main в дизассемблированной программе?*

* По символам и именам функций (если есть символы)
* В большинстве случаев, main — это точка входа, которая обычно располагается в начале программы
* В Ghidra или IDA можно искать функции, просмотрев таблицу функций или анализируя начало программы

1. *Как дизассемблер помогает найти проверку пароля в программе?*

* Анализируя функции, можно искать строки, связанные с вводом пароля
* Ищите условные конструкции (cmp, je, jne) после обработки ввода
* Можно отслеживать логику сравнения и увидеть, где происходит проверка

1. *Как определить, какие библиотеки использует программа?*

* Анализируя импорты (import таблицу)
* Ищите вызовы функций из стандартных или сторонних библиотек
* В Ghidra или IDA можно просмотреть список импортов и вызовов

**4. Практическое применение**

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

* Запустить программу в режиме отладки
* Установить точки останова перед операцией деления
* Идентифицировать переменные, участвующие в делении
* Если программа падает, увидеть стек вызовов и локальные переменные, чтобы понять, почему деление было на ноль

**5. Обратная разработка (Reverse Engineering)**

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

* Использовать дизассемблер для анализа бинарника
* Найти ключевые функции и блоки логики
* Воссоздать алгоритм вручную, описывая его на псевдокоде или языке программирования
* Использовать отладчик для отслеживания исполнения

1. *Как найти ключевые функции в дизассемблированной программе?*

* Ищите функции с подозрительно названными или типичными для логики проверки пароля или шифрования
* Обращайте внимание на строки, связанные с вводом данных, вводом пароля
* Анализируйте вызовы и условные переходы

1. *Как определить, использует ли программа шифрование?*

* Ищите вызовы функций криптографических библиотек
* Анализируйте строки, связанные с ключами или шифрованными данными
* Обратите внимание на участки с длинными байтовыми массивами или функциями, похожими на шифрование/дешифрование

1. *Как проанализировать вредоносное ПО с помощью дизассемблера?*

* Определить точки входа и ключевые функции
* Ищите команды, связанные с сетевым взаимодействием, файлами или регистрацией
* Анализируйте, есть ли шифрование или сокрытие логики
* Используйте динамический анализ (отладчик) для отслеживания поведения

**6. Дополнительные вопросы**

1. *В чем разница между статическим и динамическим анализом?*

* **Статический анализ** — исследование кода без его запуска: анализ файла, дизассемблирование, просмотр строк
* **Динамический анализ** — запуск программы и наблюдение за ее поведением: отладка, мониторинг системных вызовов, сетевых соединений

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

* Не использовать инструменты для взлома чужих систем без разрешения
* Соблюдать законы и правила по защите информации
* Не распространять вредоносное ПО или не использовать полученные знания для вредоносных целей
* Использовать такие инструменты только для тестирования собственной системы или в рамках легальных задач