**1. Основные понятия**

**1. Отладчик: назначение и функции**

Отладчик – специализированная программа для интерактивного контроля выполнения приложений. Основные возможности:

* Построчное выполнение кода с возможностью остановки
* Инспектирование и модификация переменных в runtime
* Установка контрольных точек (breakpoints)
* Трассировка стека вызовов функций

**2. Сравнение отладчика и дизассемблера**

| **Характеристика** | **Отладчик** | **Дизассемблер** |
| --- | --- | --- |
| Режим работы | Динамический (с выполнением) | Статический (без запуска) |
| Возможности | Изменение состояния программы | Анализ структуры кода |
| Основное применение | Поиск логических ошибок | Обратная разработка |

**3. Типы обнаруживаемых ошибок**

* Алгоритмические (некорректная бизнес-логика)
* Работы с памятью (buffer overflow, memory leaks)
* Исключительные ситуации (access violation)
* Проблемы потоков (race conditions, deadlocks)

**4. Точки останова и их виды**

Breakpoint – маркер, приостанавливающий выполнение программы. Разновидности:

* **Простые** – срабатывают при достижении строки
* **Условные** – активируются по логическому выражению
* **Аппаратные** – используют регистры процессора

**2. Техники отладки**

**5. Запуск отладки в Visual Studio**

1. Открыть решение
2. Нажать **F5** (или через меню *Debug → Start Debugging*)
3. Для детализации использовать:
   * **F10** – обход функций
   * **F11** – вход в функции

**6. Управление выполнением**

| **Действие** | **Комбинация** | **Назначение** |
| --- | --- | --- |
| Продолжить | F5 | Выполнение до следующей точки |
| Шаг с обходом | F10 | Пропуск внутренних функций |
| Шаг с входом | F11 | Детализация вызовов |
| Выход из функции | Shift+F11 | Возврат на уровень выше |

**7. Инструменты инспектирования**

* **Locals** – автоматический вывод локальных переменных
* **Watch** – мониторинг произвольных выражений
* **Immediate Window** – интерактивная консоль

**8. Работа с Watch-окном**

1. Открыть: *Debug → Windows → Watch → Watch 1*
2. Добавить переменные или выражения (напр., buffer[offset])
3. Возможность изменения значений в runtime

**9. Диагностика "немых" сбоев**

1. Включить все исключения (*Exception Settings*)
2. Режим диагностики:
   * Логи в файл/консоль
   * Анализ core dump (WinDbg, gdb)
   * Трассировка системных вызовов

**3. Анализ бинарного кода**

**10. Дизассемблеры: принцип работы**

Преобразуют машинные инструкции в:

* Ассемблерный листинг
* Псевдокод высокого уровня (Ghidra, IDA)

**11. Популярные инструменты**

* **Ghidra** – open-source решение от NSA
* **IDA Pro** – профессиональный стандарт
* **Binary Ninja** – облачная платформа
* **radare2** – консольный инструментарий

**12. Поиск точки входа**

Методы для разных языков:

* C/C++: main(), WinMain()
* Go: runtime.main
* Rust: std::rt::lang\_start

**13. Анализ аутентификации**

1. Поиск строковых констант ("Password", "Access denied")
2. Локализация функций сравнения (strncmp, bcrypt)
3. Анализ условных переходов после проверок

**14. Идентификация зависимостей**

* Windows:

powershell

Copy

Download

dumpbin /DEPENDENTS program.exe

* Linux:

bash

Copy

Download

ldd ./program

objdump -p ./program | grep NEEDED

**4. Практические сценарии**

**15. Обработка исключений**

Для деления на ноль:

1. Установить breakpoint на операцию /
2. Проверить делитель в *Watch*
3. Добавить валидацию:

c

Copy

Download

if (divisor == 0) return ERROR\_INVALID\_PARAMETER;

**5. Реверс-инжиниринг**

**16. Восстановление логики**

1. Идентификация точек ввода/вывода
2. Анализ преобразований данных
3. Реконструкция алгоритма через графы вызовов

**17. Поиск критичных функций**

* Обработка данных: CryptDecrypt, AES\_decrypt
* Сетевое взаимодействие: send, recv
* Работа с файлами: CreateFileW, fopen

**18. Детектирование криптографии**

Признаки:

* Импорт функций из advapi32.dll, libcrypto
* Наличие S-боксов (AES) или больших простых чисел (RSA)
* Вызовы CryptGenKey, EVP\_CIPHER\_CTX\_new

**19. Анализ вредоносного кода**

Методология:

1. Изолированная среда (VM, sandbox)
2. Мониторинг поведения (Process Monitor, Wireshark)
3. Дизассемблирование ключевых модулей

**6. Дополнительные аспекты**

**20. Методы анализа**

| **Критерий** | **Статический** | **Динамический** |
| --- | --- | --- |
| Требует запуска | Нет | Да |
| Глубина анализа | Структура кода | Поведение в runtime |
| Инструменты | Ghidra, IDA | x64dbg, OllyDbg |

**21. Юридические аспекты**

Разрешенные сценарии:

* Анализ собственного ПО
* Исследование уязвимостей (с согласия владельца)
* Обучение и сертификация

Запрещенные действия:

* Обход лицензионной защиты
* Модификация чужих программ
* Распространение декомпилированного кода

**Примечание:** В большинстве стран реверс-инжиниринг без согласия правообладателя нарушает DMCA и GDPR.