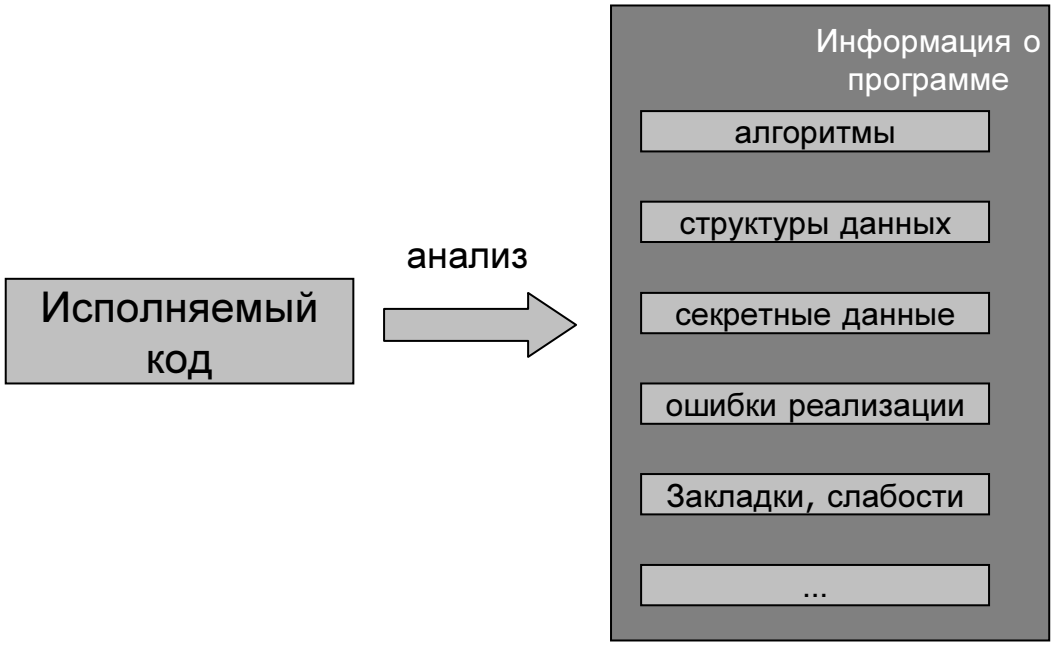
План

- Задачи анализа бинарного кода
- Статический и динамический анализ
- Средства статического и динамического анализа
  - IDA Pro
  - Zynamics: BinNavi, BinDiff и др.
  - BitBlaze: Vine, TEMU и др.
  - DynamoRIO
  - Avalanche
- Формы представления результатов анализа

Задачи анализа кода

- Поиск и классификация уязвимостей. Разработка защиты.
  - Проверка политик безопасности
- Обратная инженерия
  - Анализ зловредного кода
  - Сертификация ПО, выявление НДВ
  - Восстановление протоколов
  - «Наследственные» системы
- Тестирование ПО
- Отладка
  - Пользовательский уровень
  - Ядро ОС & драйверы периферийных устройств
  - Многопоточные приложения, IPC
  - Интеграция бинарных модулей

Анализ бинарного кода

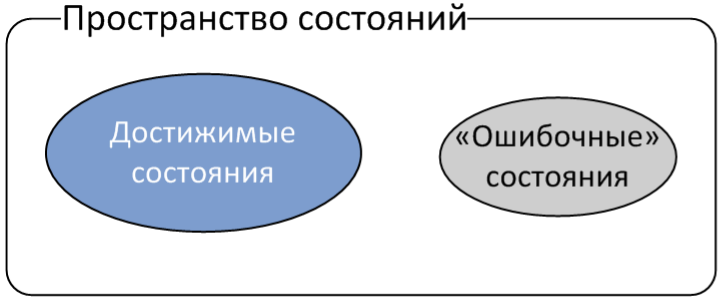


- Отсутствие исходных текстов.
- Противодействие со стороны анализируемой системы.
- Распределенность анализируемой системы по нескольким компьютерам.
- «Незамкнутость» анализируемой системы.
- Невозможно обойтись только статическим анализом. Обфускация (диспетчер), виртуальные машины, запаковка/шифрование кода.

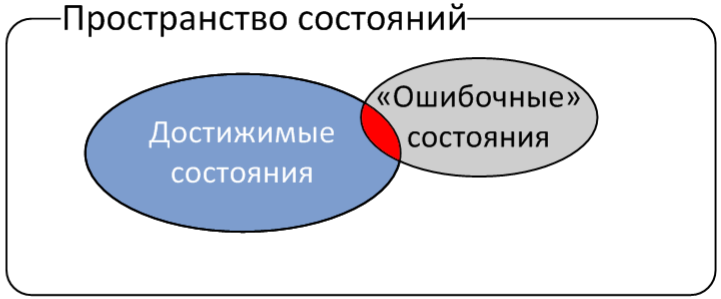
Анализ: статический и динамический

- Рассматриваются все потенциально достижимые состояния программы
- Используется приближенная оценка свойств программы
  - Domain
- Состояния программы
  - Достижимые
  - Возможно достижимые
  - Недостижимые

- Рассматривается только те состояния, которые возникают при обработке заданного набора входных данных
- Часть состояний не рассматривается



Программа без дефектов



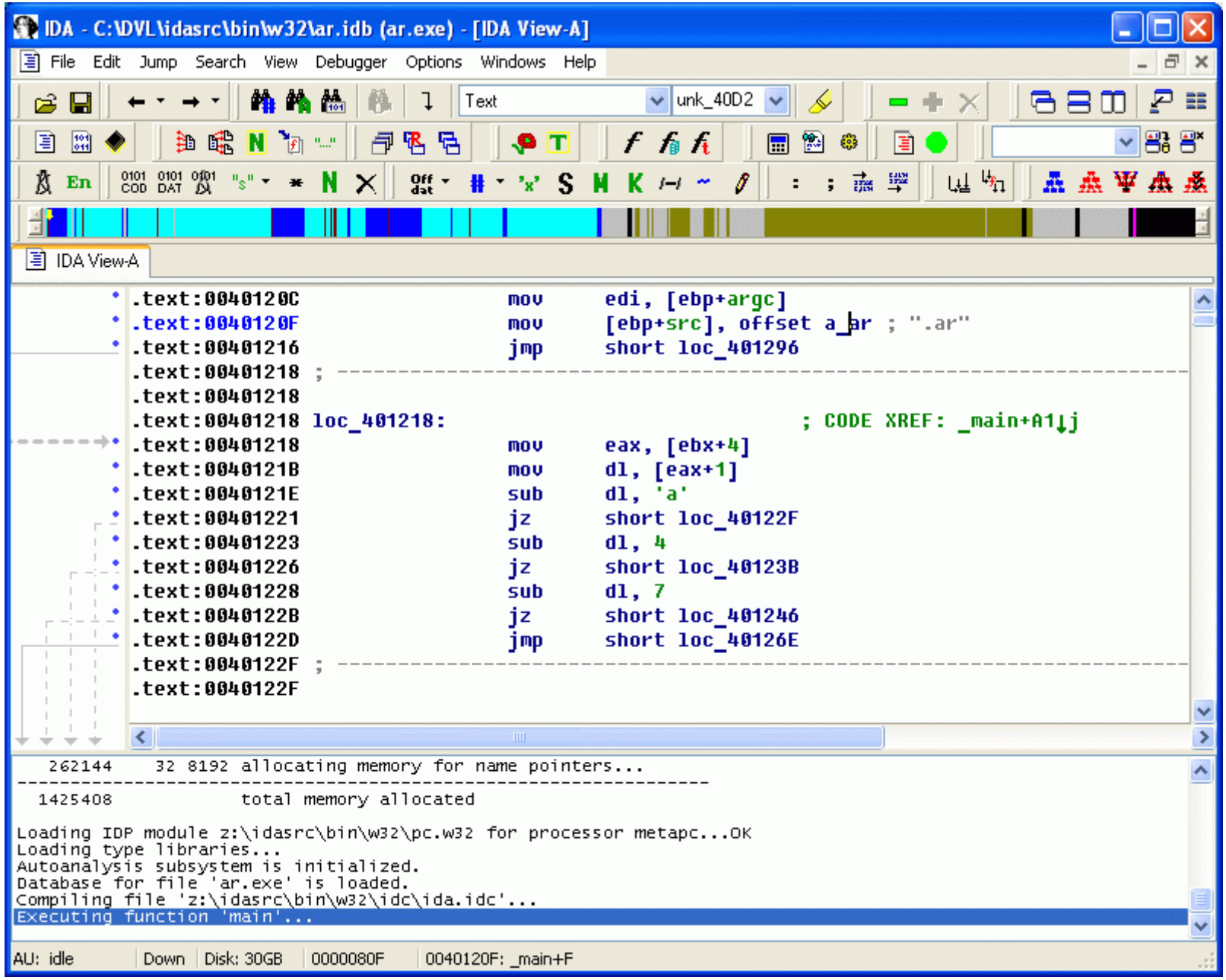
Программа с дефектами

Задачи анализа кода

	Исходный код	Бинарный код
Статический анализ	• Поиск дефектов (уязвимостей) • Обратная инженерия: структура программы • Верификация и валидация	Обратная инженерия: алгоритмы, структуры данных
Динамический анализ	• Тестирование • Отладка – функциональности, – производительности	Обратная инженерия защищенного кода: алгоритмы, структуры данных

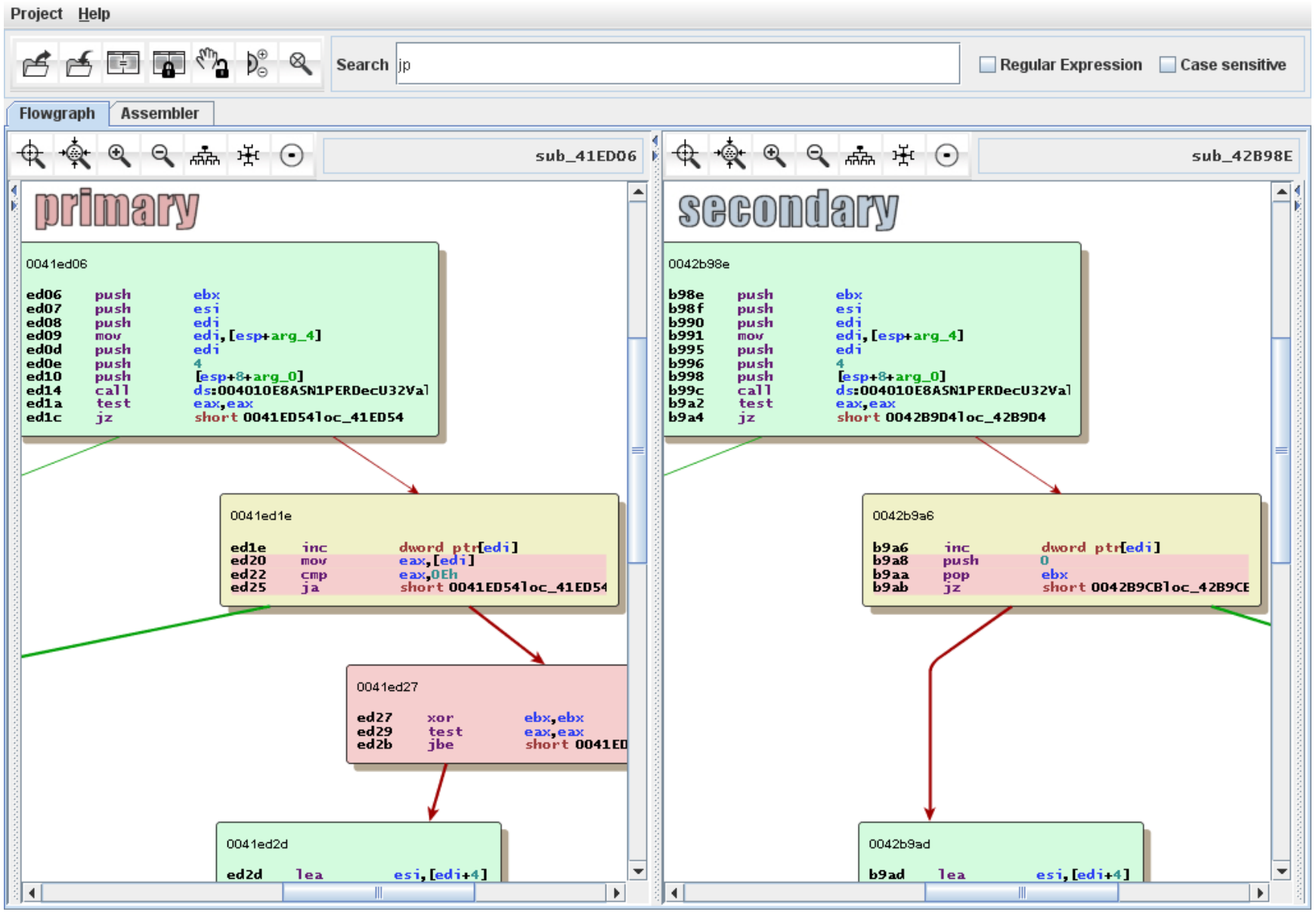
IDA Pro

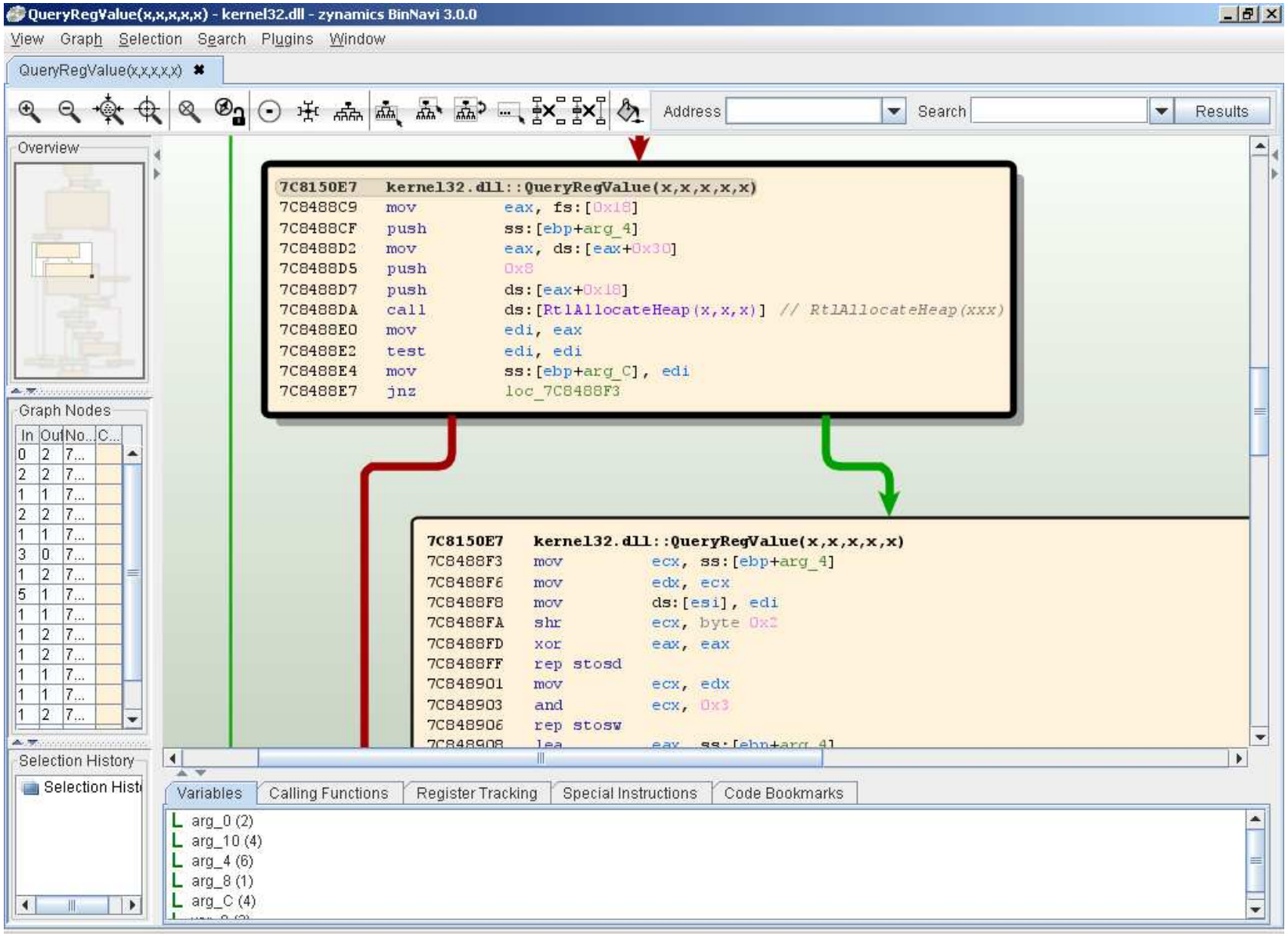
- Анализ статического кода программы
- Большое количество поддерживаемых архитектур и форматов данных
- Развитых графический интерфейс
  - Возможность рассмотрения всего кода
  - Навигация по коду
- Встроенный язык программирования IDC
- Модульная архитектура
- Распознавание библиотечных функций
- Декомпилятор для x86 и ARM
- Отладчик для Win32

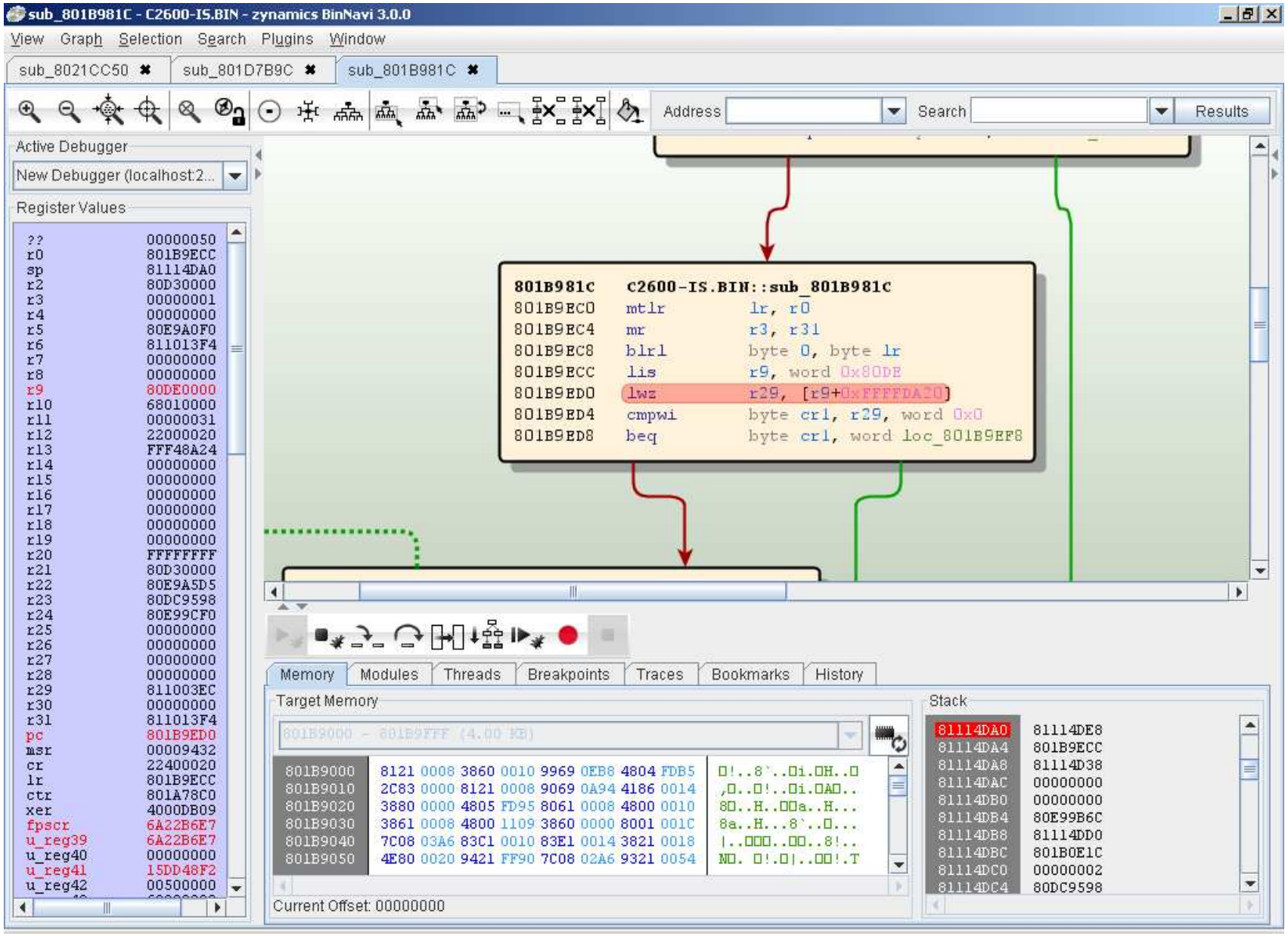


Пример интеграции с IDA Pro

- Zynamics
  - BinDiff – Сравнение различных версий
  - BinNavi – Расширение возможностей IDA Pro в части навигации по коду
  - VxClass – Классификация зловредного кода
    - Реестр зловредного кода
    - Удаление навесной защиты (unpacker & crypter)
  - BinCrowd – Коллективная база знаний
  - Специализированная утилита для анализа PDF-файлов



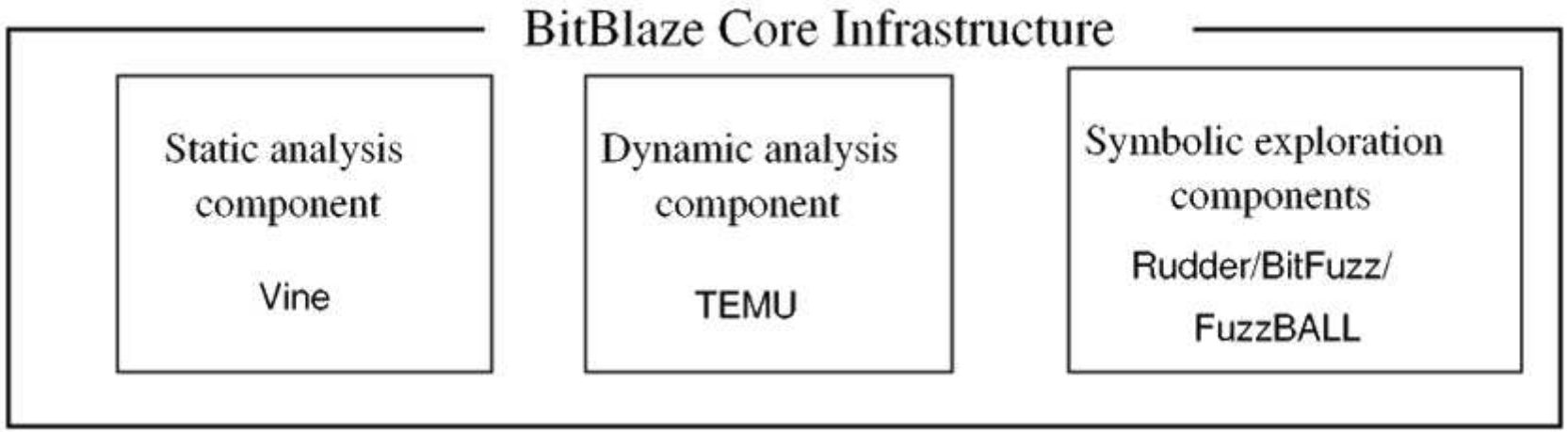




Визуализируемые данные

- Поток дизассемблированных (декодированных) команд
- Карта секций
- Граф вызовов функций (call graph)
  - Указатели, табличные функции
- Граф потока управления
  - Безусловные переходы с операндом регистром/памятью
- Граф зависимостей
  - Указатели/адресный код

BitBlaze



- Статический анализ кода (Vine)
- Динамический анализ кода (TEMU)
- Символьное исполнение исполнение кода
  - -по трассе BitFuzz
  - -по статическому представлению FuzzBALL

Статический анализ – Vine

- Представление программы в виде графов потока управления(CFG) и графа зависимостей по данным(PDG) возможных значений ячеек памяти (Value Set Analysis)
- Представление программы в SSA форме, удобное для анализа потоков данных.
- Алгоритм чопинга (chopping), получающий на вход источник и сток (точки) и выделяющий из программы её часть, которая использует данные определяемые в источнике и изменяет данные использующиеся в стоке.
- Анализа потоков данных и оптимизаций общего вида:

  - global value numbering

  -распространение констант

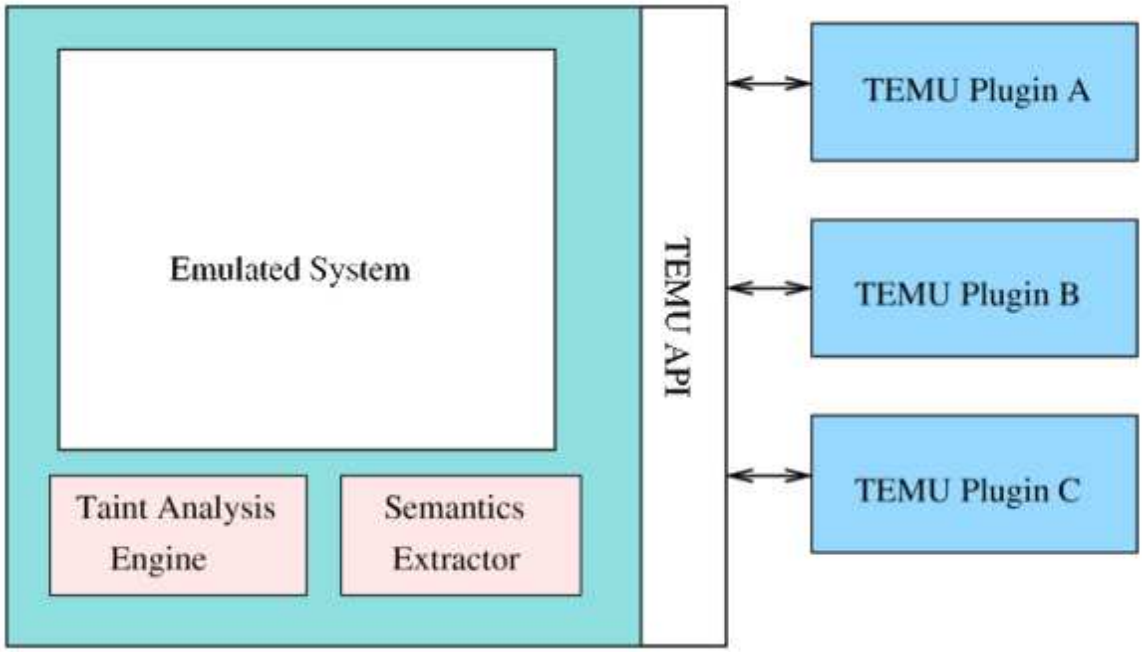
  -чистка мёртвого кода

  -анализ времени жизни переменных

  -целочисленный анализ диапазонов

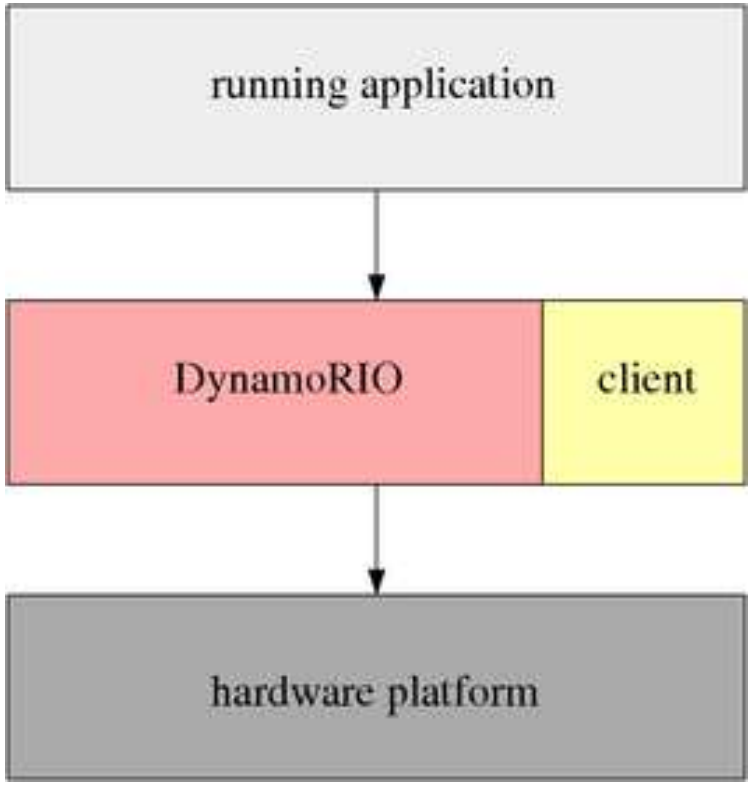
  -анализ алиасов на основе VSA.
- Верификация на основе вычисления слабейших предусловий: для любых удовлетворяющих условиям входных данных, ему удовлетворяющих, гарантируется корректное завершение программы
- Решение систем ограничений с помощью модуля STP

TEMU



Снятие бинарной трассы уровня машинных инструкций
Анализ помеченных данных (Taint analysus)
Получение высокоуровневой семантики (загрузка библиотек, запуск процессов)
Интерфейс для написания плагинов дополнительного анализа

Динамическое инструментирование – DynamoRIO



- Анализ кода пользовательского уровня
- Инструментирование кода программы «на лету»
- Регистрация интересующих свойств программы
- Получение сигналов от системы о наступлении интересующих событий
- Пример клиента системы TaintTrace – анализ помеченных данных

Поиск дефектов – Avalanche

- Итеративный запуск анализируемой программы на автоматически генерируемых входных данных, обеспечивающими максимально большое покрытие кода программы трассами
- Автоматическая генерация входных данных, демонстрирующих найденную ошибку или уязвимость в программе

Принцип работы Avalanche

- Запуск на случайных входных данных
- Входные данные заменяются символами
- В процессе исполнения программы составляются системы уравнений над символами
- В точках ветвлений зависящих от входных данных системы уравнений решаются – генерируются новые входные данные для входа в нереализовавшуюся ветвь
- В точках потенциально опасного использования (деление на 0) системы уравнений решаются – генерируются входные данные приводящие к ошибке