ISP RAS

Динамический анализ бинарного кода

Падарян Вартан
vartan@ispras.ru

Цели и задачи

Защита информации

- Поддержание работоспособности ПО
- Соблюдение политик безопасности
- Сертификация ПО, НДВ

Анализ

- Анализ реализации
- Несанкционированный доступ к данным
- Отказ в обслуживании

Типовые задачи

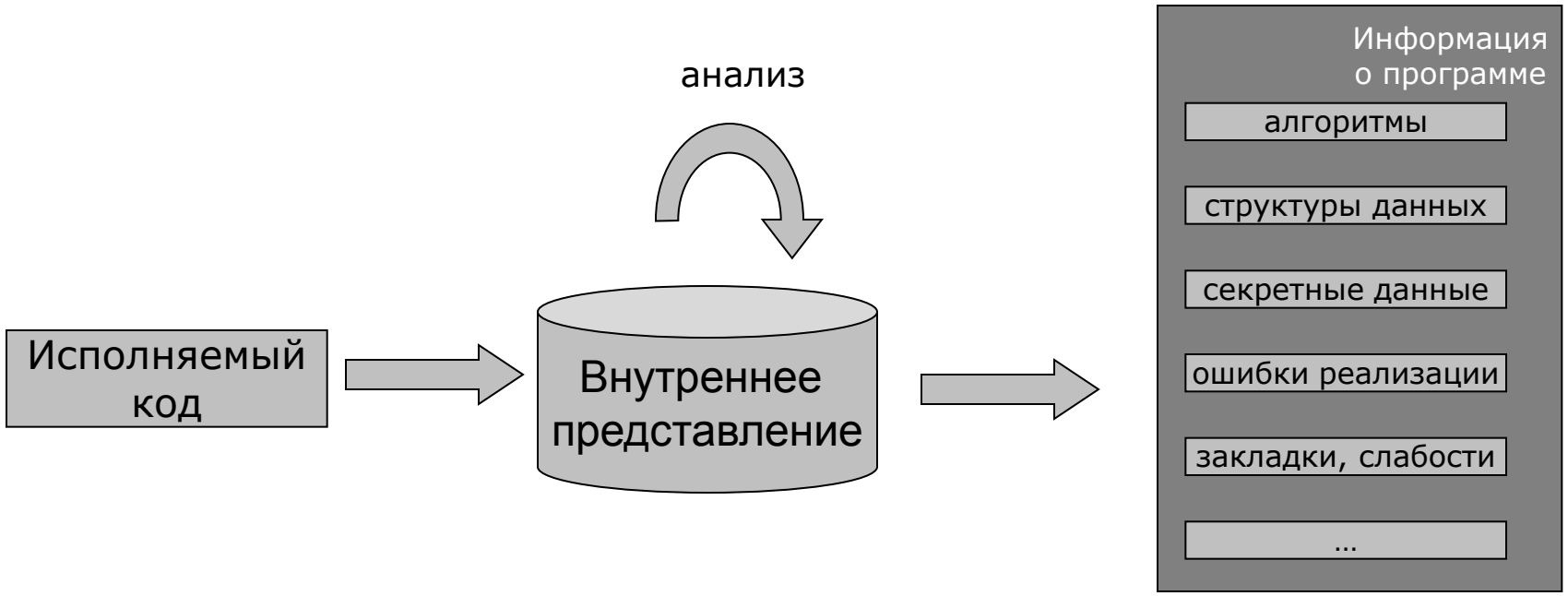
- Восстановление алгоритмов и форматов данных
- Поиск ошибок
- Сопоставление модели и ее реализации

Оборудование

- Персональные компьютеры, сервера
- Коммуникационное оборудование
- Микроконтроллеры
- Мобильные устройства

Направления работ

- **Среда анализа / инфраструктура**
  - Запуск задач и сохранение результатов
  - Формы представления кода программ
  - Визуализация
- **Виртуальная машина / источник данных для анализа**
  - QEMU: http://wiki.qemu.org/Main_Page
  - Детерминированное воспроизведение
  - Виртуальные периферийные устройства
  - Общие задачи оптимизации
- Автоматизированный поиск дефектов и ошибок
- Мобильные устройства / Android / ARM
- Глубокая инспекция сетевых пакетов (DPI)



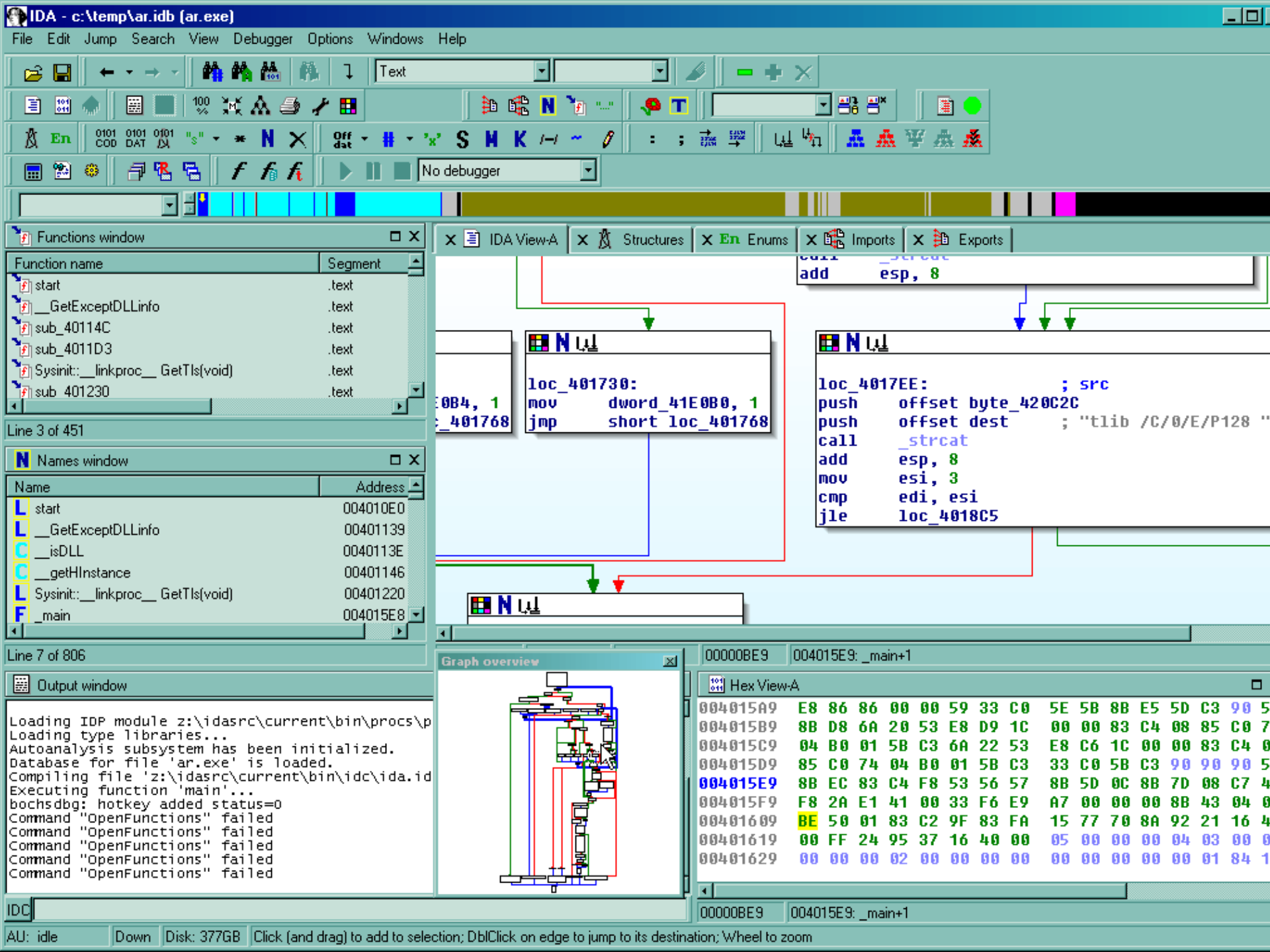
- Отсутствие исходных текстов.
- Противодействие со стороны анализируемой системы.
  - Антиотладочные приемы
  - Обфускация
  - Шифрование кода и данных
- Распределенность анализируемой системы по нескольким компьютерам.
- «Незамкнутость» анализируемой системы.

Некоторые методы защиты от анализа

- Навесная: упаковка/шифрование исполняемого файла
  - Последний этап защиты после компиляции из исходных кодов, с учетом дополнительной информации полученной при компиляции
- Встраиваемая
  - Запутывание программного кода во время компиляции
  - Встраивание различных приемов, защиты от тестирования отладчиками
- Пример часто применяемой: виртуальная машина
  - Встраивание интерпретатора инструкций сторонней архитектуры и исполнение на нем частей программы

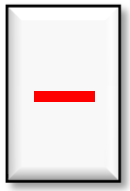
Средства анализа

- Статический анализ
  - Двоичный редактор
  - Дизассемблер (IDA Pro)
  - Декомпилятор (IDA Pro/HexRays)
  - Среда анализа (IDA Pro + дополнения, CodeSurfer/x86)
- Динамический анализ
  - Обычный отладчик, Отладчик ядра (SoftICE, OllyDbg, WinDbg)
  - Отладчик на основе виртуальной машины
  - Трассировщик (Post-Mortem vs. Online)
- Исследовательские проекты
  - BitBlaze, CodeSurfer/x86, S2E …


IDA - c:\temp\ar.idb (ar.exe)
File Edit Jump Search View Debugger Options Windows Help
Text
No debugger
Functions window
Function name
Segment
start .text
__GetExceptDLLinfo .text
sub_40114C .text
sub_4011D3 .text
Sysinit::__linkproc__ GetTls(void) .text
sub_401230 .text
Line 3 of 451
Names window
Name
Address
start 004010E0
__GetExceptDLLinfo 00401139
__isDLL 0040113E
__getHInstance 00401146
Sysinit::__linkproc__ GetTls(void) 00401220
_main 004015E8
Line 7 of 806
IDA View-A
Structures
Enums
Imports
Exports
add esp, 8
loc_401730:
mov dword_41E0B0, 1
jmp short loc_401768
loc_4017EE: ; src
push offset byte_420C2C
push offset dest ; "tlib /C/0/E/P128 "
call _strcat
add esp, 8
mov esi, 3
cmp edi, esi
jle loc_4018C5
Graph overview
00000BE9 004015E9: _main+1
Output window
Loading IDP module z:\idasrc\current\bin\procs\p
Loading type libraries...
Autoanalysis subsystem has been initialized.
Database for file 'ar.exe' is loaded.
Compiling file 'z:\idasrc\current\bin\idc\ida.id
Executing function 'main'...
bochsdbg: hotkey added status=0
Command "OpenFunctions" failed
Command "OpenFunctions" failed
Command "OpenFunctions" failed
Command "OpenFunctions" failed
Command "OpenFunctions" failed
Hex View-A
004015A9 E8 86 86 00 00 59 33 C0 5E 5B 8B E5 5D C3 90 5
004015B9 8B D8 6A 20 53 E8 D9 1C 00 00 83 C4 08 85 C0 7
004015C9 04 B0 01 5B C3 6A 22 53 E8 C6 1C 00 00 83 C4 0
004015D9 85 C0 74 04 B0 01 5B C3 33 C0 5B C3 90 90 90 5
004015E9 8B EC 83 C4 F8 53 56 57 8B 5D 0C 8B 7D 08 C7 4
004015F9 F8 2A E1 41 00 33 F6 E9 A7 00 00 00 8B 43 04 0
00401609 BE 50 01 83 C2 9F 83 FA 15 77 70 8A 92 21 16 4
00401619 00 FF 24 95 37 16 40 00 05 00 00 00 04 03 00 0
00401629 00 00 00 02 00 00 00 00 00 00 00 00 00 01 84 1
00000BE9 004015E9: _main+1
IDC
AU: idle Down Disk: 377GB Click (and drag) to add to selection; DblClick on edge to jump to its destination; Wheel to zoom

Недостатки и достоинства

- Статическое представление программы – естественное для человеческого мышления



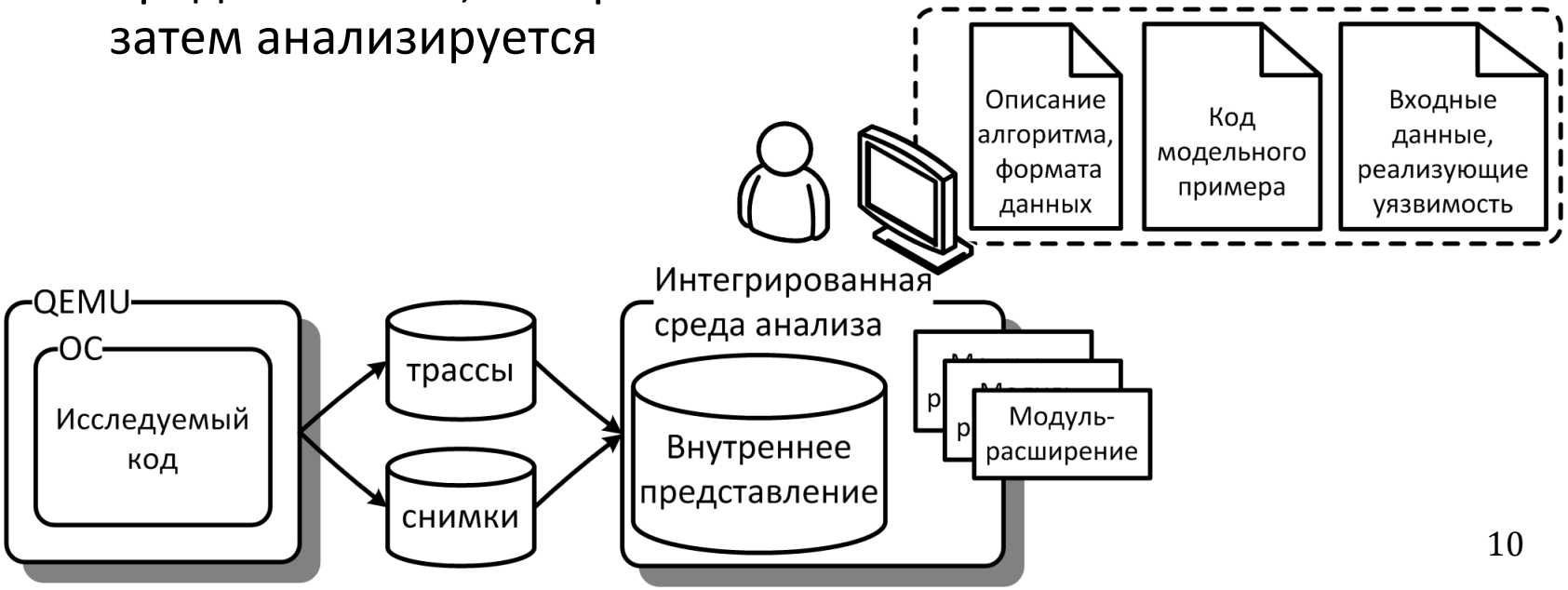
- Косвенная адресация и динамическое изменение кода
  - Восстановленное статическое представление не полное
- IDA Pro анализирует отдельный модуль
  - Минимальные ухищрения – доступно все адресное пространство процесса
- IDA Pro – инфраструктура
  - Анализ – IDC-скрипты и модули-расширения, иногда очень дорогие (Zynamics, поглощенный Google)

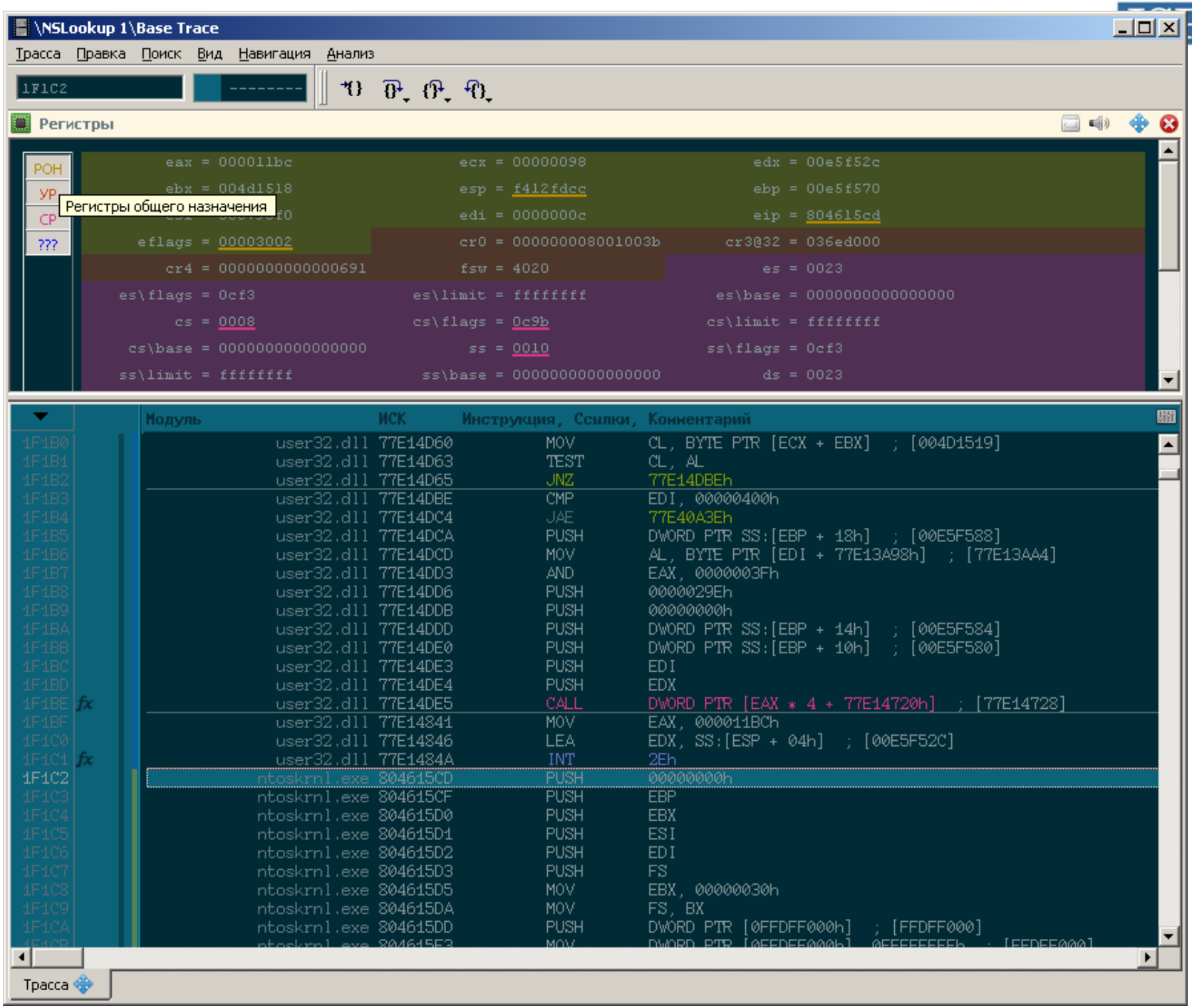
Пример:
некоторый «нестандартный случай»

- Поведение определяется обрабатываемыми данными
  - Необходимы результаты динамического анализа
- Нестандартная операционная система
  - Наилучший случай: модифицированный Linux, BSD, …
- Доступна только прошивка, часть аппаратуры неизвестна

Подход к анализу

- Трасса содержит все машинные команды, выполненные процессором
- По трассам восстанавливаются высокоуровневые события
- По трассам строится статическое представление, которое затем анализируется




\NSLookup 1\Base Trace
Трасса Правка Поиск Вид Навигация Анализ
1F1C2
Регистры
РОН
УР
СР
???
Регистры общего назначения
eax = 000011bc
ecx = 00000098
edx = 00e5f52c
ebx = 004d1518
esp = f412fdcc
ebp = 00e5f570
edi = 0000000c
eip = 804615cd
eflags = 00003002
cr0 = 000000008001003b
cr3@32 = 036ed000
cr4 = 0000000000000691
fsw = 4020
es = 0023
es\flags = 0cf3
es\limit = ffffffff
es\base = 0000000000000000
cs = 0008
cs\flags = 0c9b
cs\limit = ffffffff
cs\base = 0000000000000000
ss = 0010
ss\flags = 0cf3
ss\limit = ffffffff
ss\base = 0000000000000000
ds = 0023
Модуль ИСК Инструкция, Ссылки, Комментарий
1F1B0 user32.dll 77E14D60 MOV CL, BYTE PTR [ECX + EBX] ; [004D1519]
1F1B1 user32.dll 77E14D63 TEST CL, AL
1F1B2 user32.dll 77E14D65 JNZ 77E14DBEh
1F1B3 user32.dll 77E14DBE CMP EDI, 00000400h
1F1B4 user32.dll 77E14DC4 JAE 77E40A3Eh
1F1B5 user32.dll 77E14DCA PUSH DWORD PTR SS:[EBP + 18h] ; [00E5F588]
1F1B6 user32.dll 77E14DCD MOV AL, BYTE PTR [EDI + 77E13A98h] ; [77E13AA4]
1F1B7 user32.dll 77E14DD3 AND EAX, 0000003Fh
1F1B8 user32.dll 77E14DD6 PUSH 0000029Eh
1F1B9 user32.dll 77E14DDB PUSH 00000000h
1F1BA user32.dll 77E14DDD PUSH DWORD PTR SS:[EBP + 14h] ; [00E5F584]
1F1BB user32.dll 77E14DE0 PUSH DWORD PTR SS:[EBP + 10h] ; [00E5F580]
1F1BC user32.dll 77E14DE3 PUSH EDI
1F1BD user32.dll 77E14DE4 PUSH EDX
1F1BE fx user32.dll 77E14DE5 CALL DWORD PTR [EAX * 4 + 77E14720h] ; [77E14728]
1F1BF user32.dll 77E14841 MOV EAX, 000011BCh
1F1C0 user32.dll 77E14846 LEA EDX, SS:[ESP + 04h] ; [00E5F52C]
1F1C1 fx user32.dll 77E1484A INT 2Eh
1F1C2 ntoskrnl.exe 804615CD PUSH 00000000h
1F1C3 ntoskrnl.exe 804615CF PUSH EBP
1F1C4 ntoskrnl.exe 804615D0 PUSH EBX
1F1C5 ntoskrnl.exe 804615D1 PUSH ESI
1F1C6 ntoskrnl.exe 804615D2 PUSH EDI
1F1C7 ntoskrnl.exe 804615D3 PUSH FS
1F1C8 ntoskrnl.exe 804615D5 MOV EBX, 00000030h
1F1C9 ntoskrnl.exe 804615DA MOV FS, BX
1F1CA ntoskrnl.exe 804615DD PUSH DWORD PTR [0FFDFF000h] ; [FFDFF000]
Трасса

Методика анализа: повышение уровня представления (1/2)

- Первоначальная разметка трассы на процессы и нити
  - регистры-маркеры, процессы/треды/зоны
- Восстановление поколений кода и статического представления кода в памяти
  - области записи/чтения/выполнения
- Выделение обработчиков прерываний
  - прерывания: вложенные, каскадные, …
- Выделение границ вызовов функций и восстановление стека вызовов
  - сколько есть способов вызвать функцию и вернуть управление?

Методика анализа: повышение уровня представления (2/2)

- Построение статических объектов: функций, точек вызова, графа вызовов и привязка функций к вызовам
  - На каком этапе это лучше делать?
- Восстановление экземпляров модулей, загруженных в память в момент снятия трассы
  - Модуль может быть многократно загружен даже в одном процессе
- Разметка адресного пространства отдельных модулей
- Построение межпроцедурного графа потока управления (PDS)

Что такое функция?

Атрибуты функций в языках высокого уровня:

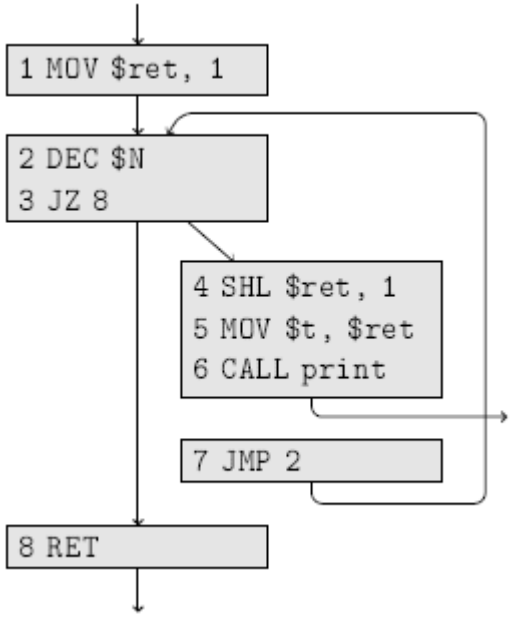
- Имя
- Множество параметров
- Возвращаемое значение
- ...

Данные признаки не применимы к бинарному коду

Что такое функция?

- Функция возвращает управление на инструкцию, следующую за вызовом функции

Существуют исключения:
exit(), fork() и т.д.

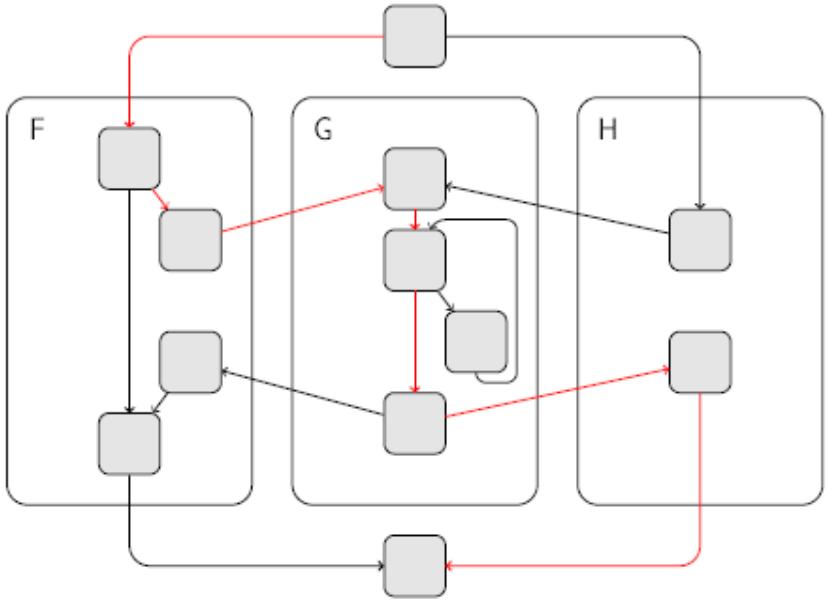


Push Down Systems (PDS)

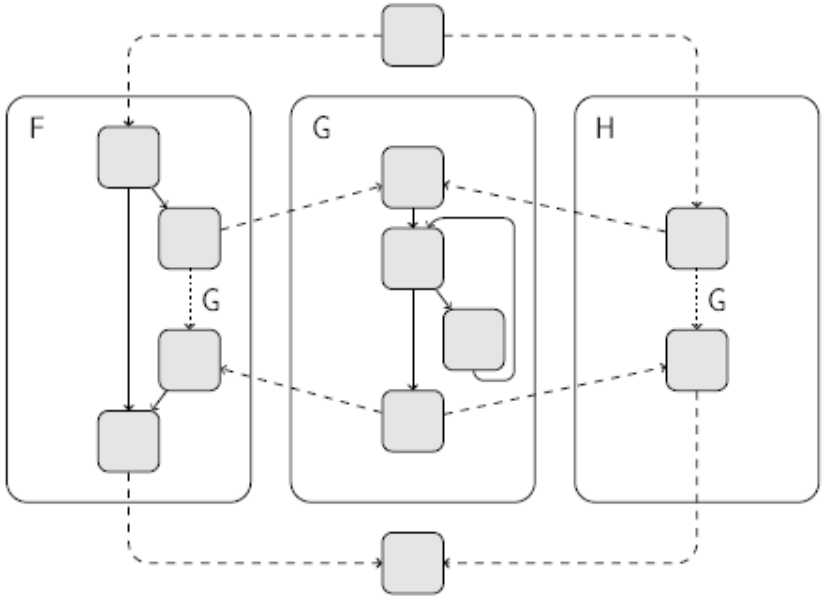
- «Система с магазинной памятью» – это вариант представления внутренней структуры программы
- Базовые блоки в PDS соединены ребрами 4 типов:
  - Переход к следующему блоку
  - Ребро вызова
  - Ребро возврата
  - Ребро, соединяющее точку вызова с точкой возврата
- Возможен проход только по одному ребру возврата

Возможные пути в CFG и PDS

CFG



PDS



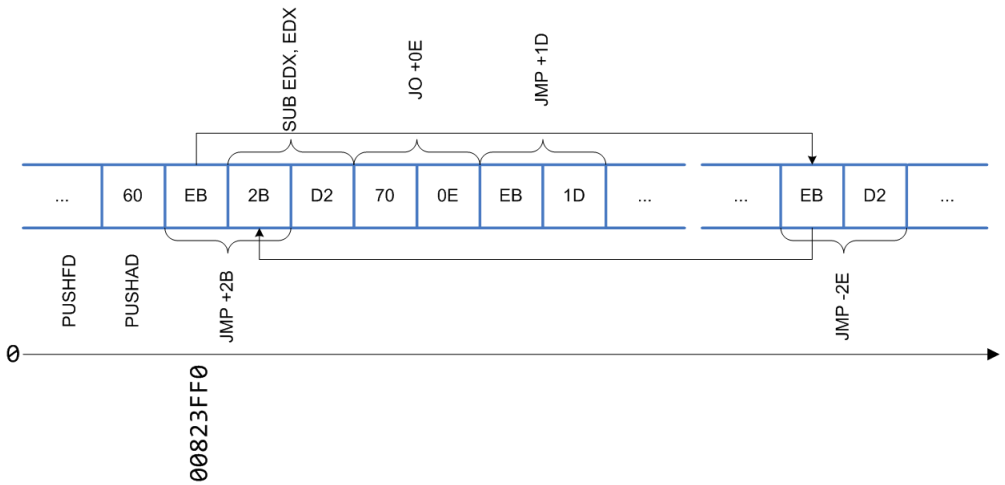
Сложности при построении

- Возможность модификации кода
- Наложение инструкции
- Выход из нескольких функций одной командой

Наложение инструкций

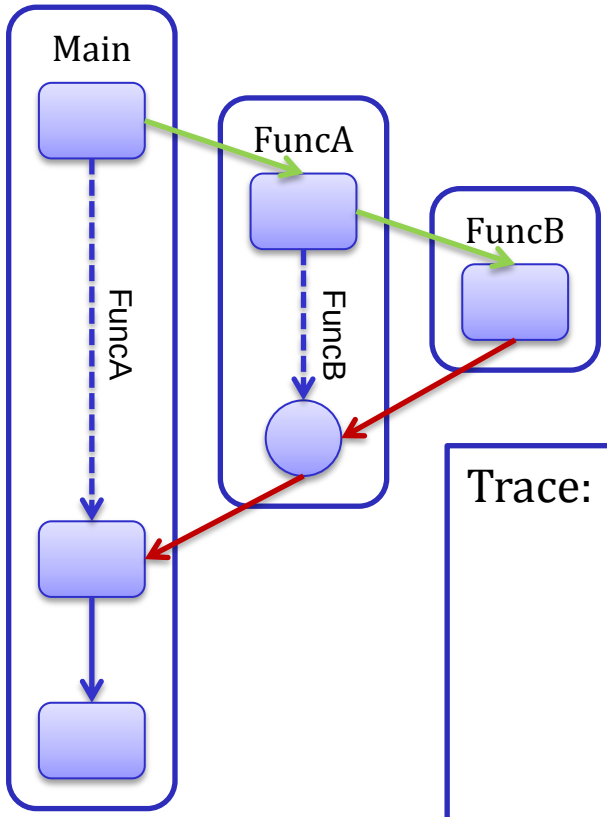
Трасса выполнения

Адрес	Машинная команда	Ассемблерная инструкция	
00823FEF	60	PUSHAD	
00823FF0	EB2B	JMP	0082401Dh
0082401D	EBD2	JMP	00823FF1h
00823FF1	2BD2	SUB	EDX, EDX
00823FF3	700E	JO	00824003h
00823FF5	EB1D	JMP	00824014h



Выход из нескольких функций

```
Main:   CALL FuncA
        MOV EAX, 0

FuncA:  JMP @FuncB

FuncB:  MOV EAX, 1
        RET
```




```
Trace:  CALL FuncA
        JMP @FuncB
        MOV EAX, 1
        RET
        MOV EAX, 0
```

Методика анализа: дополнение статического представления

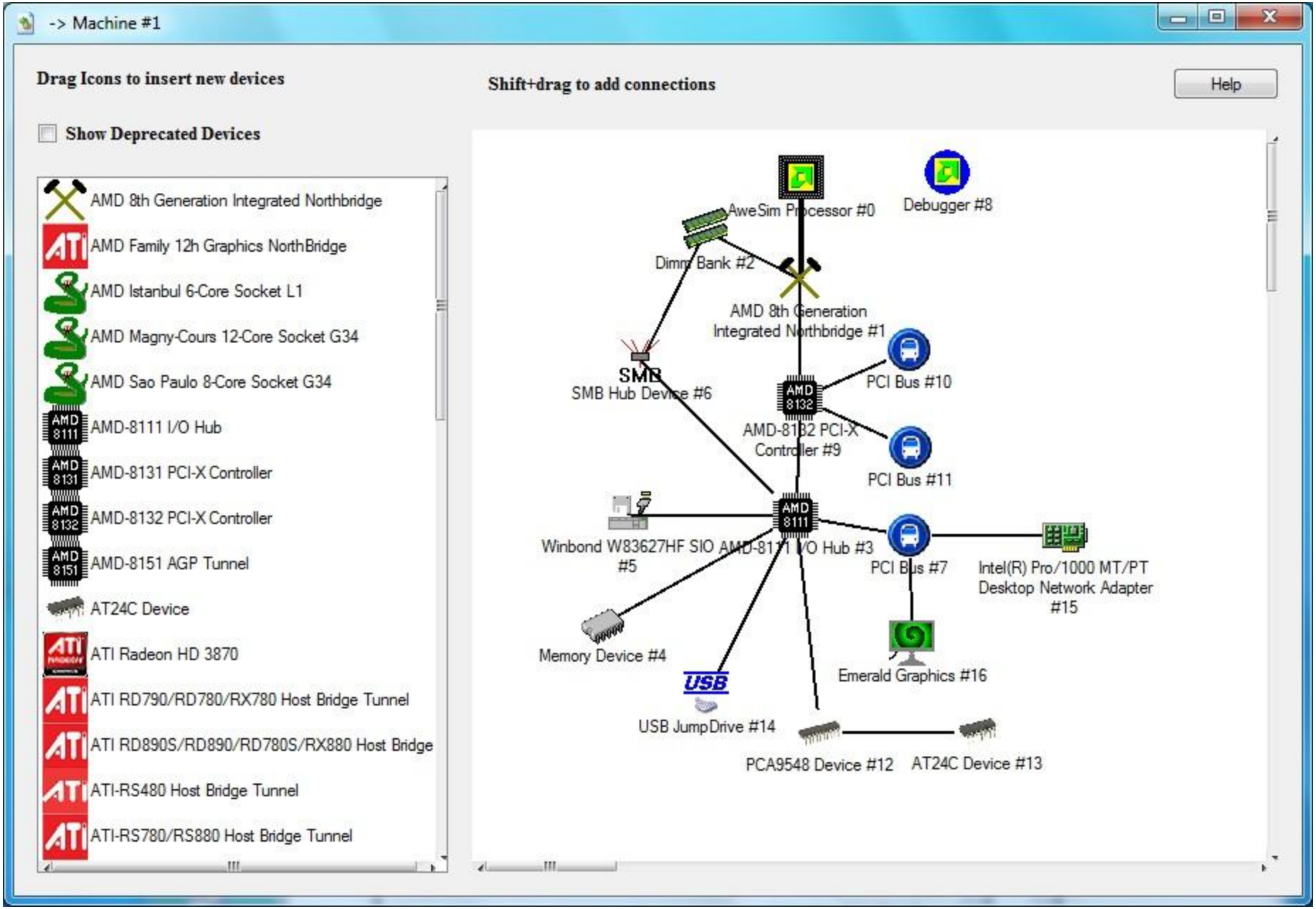
- Слияние PDS, полученных для связанных трасс
  - несколько запусков на различных входных данных с общего начального состояния
- Статическое дизассемблирование незадействованных ветвей
  - управляемое воспроизведение работы виртуальной машины
  - доступ в память виртуальной машины через интерфейс отладчика в требуемом месте трассы

Методика анализа: получение конечных результатов

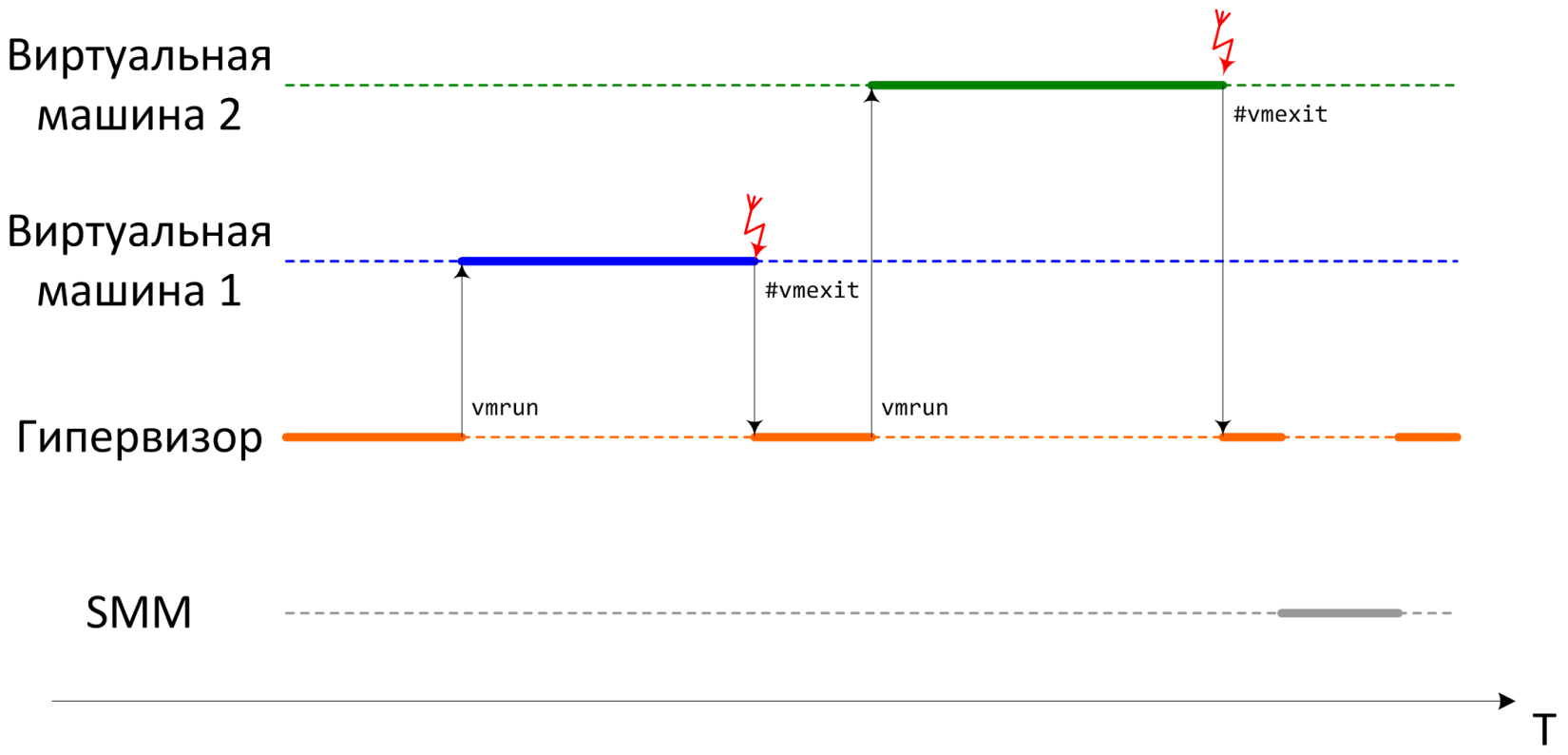
- Построение оценочных метрик
- Выделение кода определенного алгоритма
  - Адаптация программного слайсинга
  - Отслеживание зависимостей на уровне машинных команд (данные и управление)
- Упрощение представления запутанного алгоритма
  - Извлечение алгоритма из виртуальной машины
- Восстановление ассемблерного листинга
  - Портирование алгоритмов, контрольный пример
- Поиск ошибок: дефекты и ошибки
  - Анализ помеченных данных
  - Символьная интерпретация машинных команд

Современные виртуальные машины: механизмы виртуализации

- Программный эмулятор (интерпретатор)
  - AMD SimNow, Bochs, Simics, …
- Бинарная трансляция
  - QEMU, VMWare Workstation
- Аппаратная виртуализация
  - VMWare Workstation, Microsoft Hyper-V, Xen, KVM
  - Основные аппаратные механизмы
    - AMD SVM (Secure Virtual Machine)
    - Intel VT-x (Virtualization Extensions)
  - В ближайшей перспективе – ARM


-> Machine #1
Drag Icons to insert new devices
Shift+drag to add connections
Help
Show Deprecated Devices
AMD 8th Generation Integrated Northbridge
AMD Family 12h Graphics NorthBridge
AMD Istanbul 6-Core Socket L1
AMD Magny-Cours 12-Core Socket G34
AMD Sao Paulo 8-Core Socket G34
AMD-8111 I/O Hub
AMD-8131 PCI-X Controller
AMD-8132 PCI-X Controller
AMD-8151 AGP Tunnel
AT24C Device
ATI Radeon HD 3870
ATI RD790/RD780/RX780 Host Bridge Tunnel
ATI RD890S/RD890/RD780S/RX880 Host Bridge
ATI-RS480 Host Bridge Tunnel
ATI-RS780/RS880 Host Bridge Tunnel
AweSim Processor #0
Debugger #8
Dimm Bank #2
AMD 8th Generation Integrated Northbridge #1
SMB
SMB Hub Device #6
PCI Bus #10
AMD-8132 PCI-X Controller #9
PCI Bus #11
Winbond W83627HF SIO #5
AMD-8111 I/O Hub #3
PCI Bus #7
Intel(R) Pro/1000 MT/PT Desktop Network Adapter #15
Memory Device #4
Emerald Graphics #16
USB
USB JumpDrive #14
PCA9548 Device #12
AT24C Device #13

Аппаратная виртуализация



QEMU – виртуальная машина на основе бинарной трансляции

- Программа с открытым кодом
- Три режима работы
  - В том числе полносистемная эмуляция
- Поддержка распространенных платформ
  - x86, PowerPC, ARM, Sparc, …
- Возможность расширения списка поддерживаемых периферийных устройств
- Активно используется разработчиками мобильных платформ
  - Android, Symbian, Tizen, …
- Недостаток: известны случаи противодействия со стороны анализируемых программ

Снимки виртуальной машины

- QEMU позволяет полностью сохранить текущее состояние виртуальной машины в виде файла-снимка
- Сохраненное состояние может быть затем восстановлено
- Можно использовать для:
  - переноса работающей виртуальной машины на другой компьютер,
  - быстрого восстановления после сбоя откатом на последнее сохраненное состояние
  - «выключения» виртуальной машины на время ее неиспользования

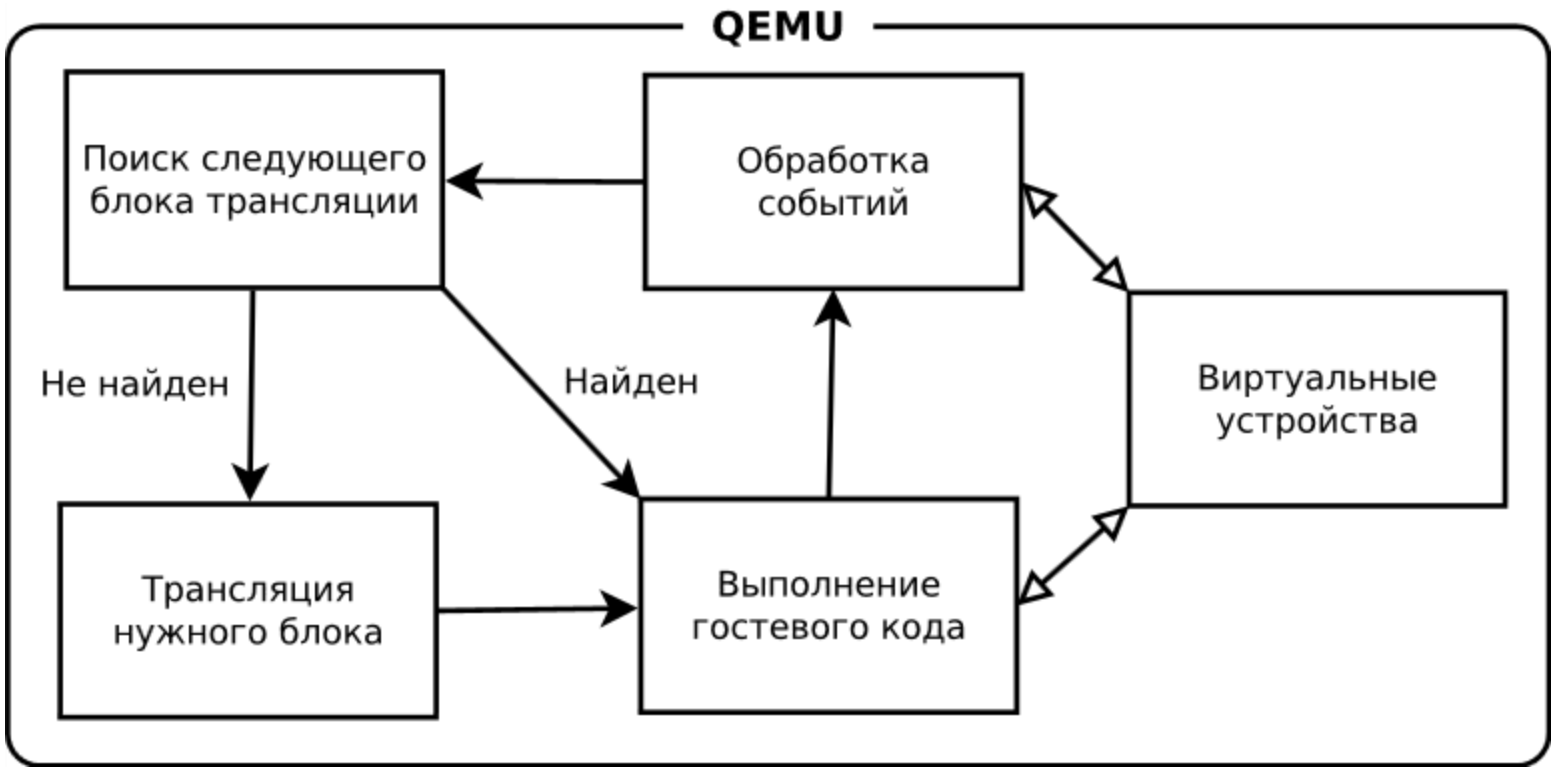
Интерфейс с отладчиком

- QEMU имеет встроенный gdb сервер. К работающему QEMU можно подключиться, используя обычный gdb и отслеживать гостевой код, включая код ядра ОС и драйверов
- Если сам QEMU тоже запущен под gdb, то можно параллельно смотреть на выполнение кода драйвера и кода, эмулирующего соответствующее устройство
- Единственное требование: «прямые» руки и навыки работы с gdb

Ядро QEMU

- Машинно-независимое ядро QEMU связывает
  - разбор гостевого машинного кода,
  - генерацию кода основной архитектуры,
  - эмуляцию различных устройств,
  - взаимодействие с основной ОС.
- Каждая из частей может разрабатываться независимо и они могут быть использованы в любых имеющих смысл комбинациях.

Ядро QEMU


QEMU
Поиск следующего блока трансляции
Обработка событий
Виртуальные устройства
Не найден
Найден
Трансляция нужного блока
Выполнение гостевого кода

Трансляция кода в QEMU: единица трансляции

- Трансляция гостевого кода производится небольшими ациклическими участками, называемыми блоками трансляции
- Транслированные блоки запоминаются в кэше трансляции и могут быть использованы повторно
- Блоки трансляции могут быть модифицированы в ходе работы так, чтобы передавать управление напрямую следующему блоку (сцепление)
- При определенных условиях сцепление базовых блоков может быть разорвано в ходе работы

Трансляция кода в QEMU: основные этапы

- Декодирование гостевого кода во внутреннее представление
- Оптимизация полученного кода
- Распределение регистров и генерация кода для основной архитектуры

Внутреннее представление QEMU

- Переменные трех видов
  - глобальные
  - локальные
  - временные
- Набор основных операций
  - пересылка данных
  - арифметические и логические операции
  - вызовы вспомогательных функций,
  - ветвление внутри блока трансляции,
  - загрузка и сохранение в адресном пространстве QEMU
  - загрузка и сохранение в адресном пространстве гостя

Вспомогательные функции

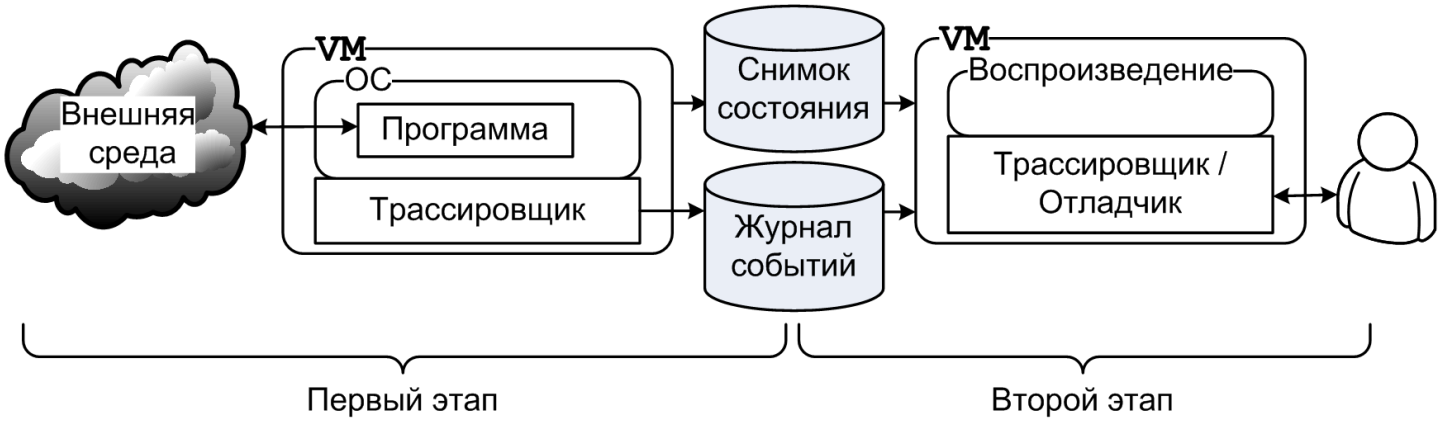
```
void helper_rdtsc(void) {
    uint64_t val;

    if ((env->cr[4] & CR4_TSD_MASK) &&
        ((env->hflags & HF_CPL_MASK) != 0)) {
            raise_exception(EXCP0D_GPF);
    }
    helper_svm_check_intercept_param(SVM_EXIT_RDTSC, 0);

    val = cpu_get_tsc(env) + env->tsc_offset;
    EAX = (uint32_t)(val);
    EDX = (uint32_t)(val >> 32);
}
```

Виртуальные устройства в QEMU

- Существует два основных типа интерфейса взаимодействия виртуальных устройств с гостевой ОС:
  - управляющие регистры, отображаемые в адресное пространство,
  - прерывания
- Помимо того, существует несколько интерфейсов для взаимодействия устройства с основной ОС
- Реализация устройства в QEMU обязательно включает в себя:
  - инициализацию и сброс состояния устройства
  - сохранение и загрузку устройства (снимки состояния)
  - функции чтения и записи управляющих регистров,
  - логику работы с одним или несколькими прерываниями

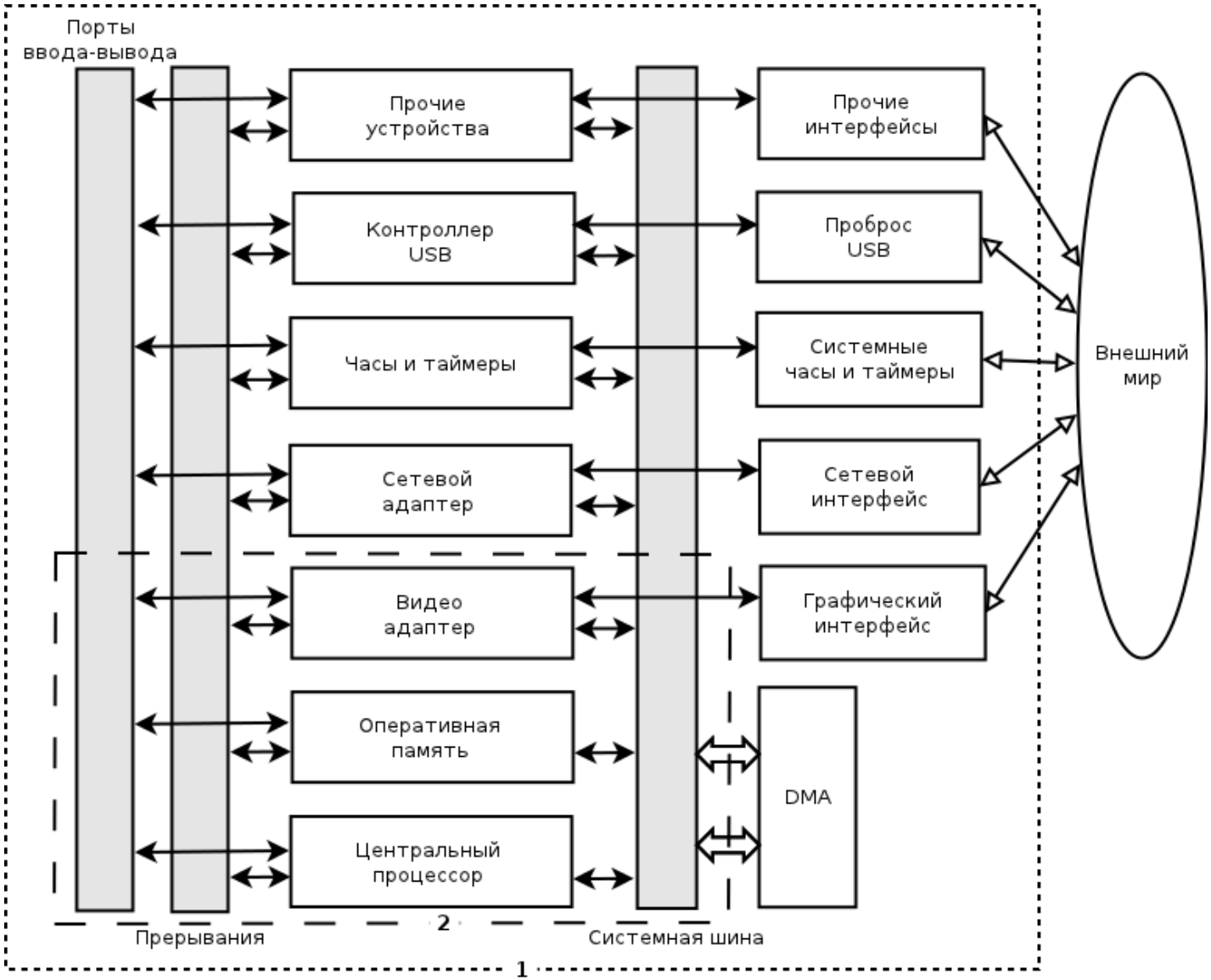
Двухпроходная трассировка

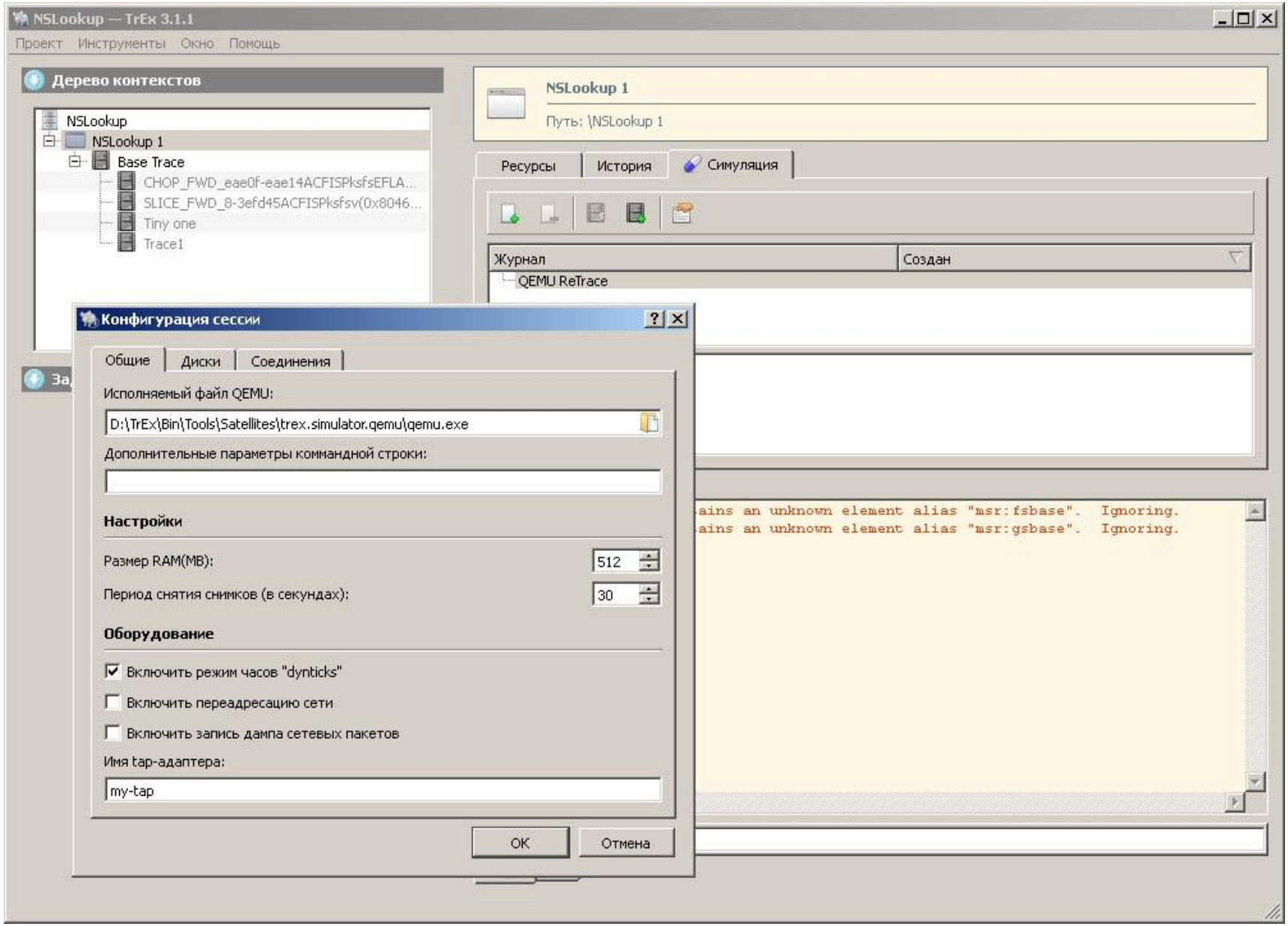
- Цель – получение детальной трассы, исключая возможность обнаружения процесса трассировки извне
- Все взаимодействие с внешним окружением фиксируется на первом этапе в журнале событий
  - Крайне низкие накладные расходы на трассировку
- Второй этап – детерминированное воспроизведение
  - Подключение интерактивного отладчика
  - Снятие детальной трассы для последующего анализа



Необходимые для восстановления трассы данные и события

- Начальное состояние гостевой системы:
  - Содержимое занятой памяти
  - Значения регистров
- События во время выполнения:
  - Прерывания
  - Доступ к портам I/O, MSR’ам
  - Данные I/O, в т.ч. данные DMA-транзакций


Порты
ввода-вывода
Прочие
устройства
Контроллер
USB
Часы и таймеры
Сетевой
адаптер
Видео
адаптер
Оперативная
память
Центральный
процессор
Прерывания
2
Системная шина
Прочие
интерфейсы
Проброс
USB
Системные
часы и таймеры
Сетевой
интерфейс
Графический
интерфейс
DMA
Внешний
мир
1


NSLookup — TrEx 3.1.1
Проект Инструменты Окно Помощь
Дерево контекстов
NSLookup
NSLookup 1
Base Trace
CHOP_FWD_eae0f-eae14ACFISPksfsEFLA...
SLICE_FWD_8-3efd45ACFISPksfsv(0x8046...
Tiny one
Trace1
NSLookup 1
Путь: \NSLookup 1
Ресурсы
История
Симуляция
Журнал
Создан
QEMU ReTrace
Конфигурация сессии
Общие
Диски
Соединения
Исполняемый файл QEMU:
D:\TrEx\Bin\Tools\Satellites\trex.simulator.qemu\qemu.exe
Дополнительные параметры коммандной строки:
Настройки
Размер RAM(MB):
512
Период снятия снимков (в секундах):
30
Оборудование
Включить режим часов "dynticks"
Включить переадресацию сети
Включить запись дампа сетевых пакетов
Имя tap-адаптера:
my-tap
OK
Отмена
ains an unknown element alias "msr:fsbase". Ignoring.
ains an unknown element alias "msr:gsbase". Ignoring.

7 B0пр0сы 7