

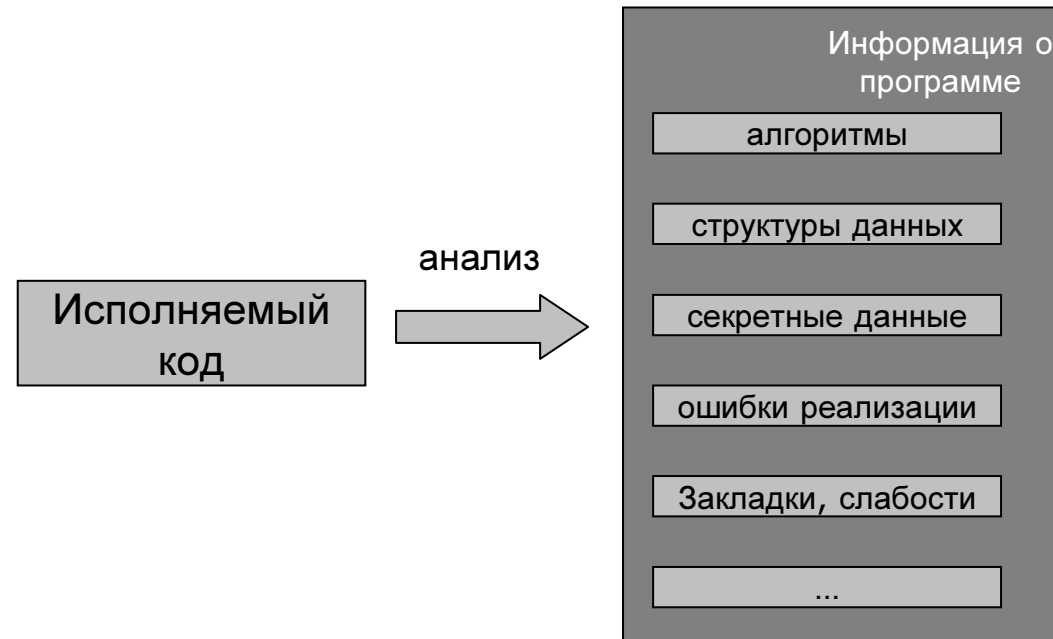
План

- 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

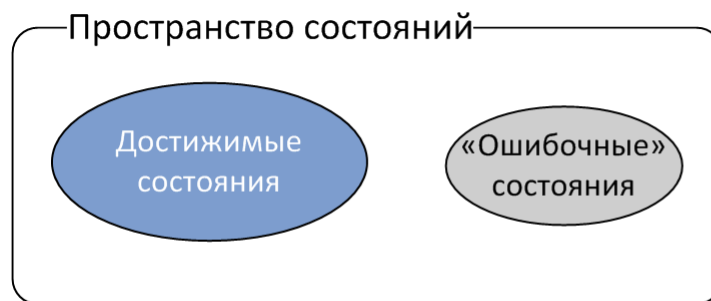
Анализ бинарного кода



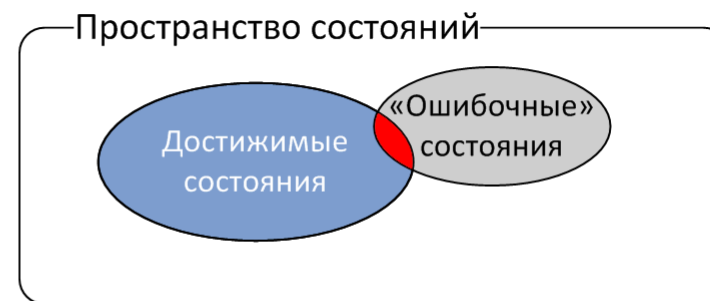
- Отсутствие исходных текстов.
- Противодействие со стороны анализируемой системы.
- Распределенность анализируемой системы по нескольким компьютерам.
- «Незамкнутость» анализируемой системы.
- Невозможно обойтись только статическим анализом. Обфускация (диспетчер), виртуальные машины, упаковка/шифрование 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	<ul style="list-style-type: none">• Поиск дефектов (уязвимостей)• Обратная инженерия: структура программы• Верификация и валидация	Обратная инженерия: алгоритмы, структуры данных
Динамический анализ	<ul style="list-style-type: none">• Тестирование• Отладка<ul style="list-style-type: none">– 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

IDA - C:\DVL\idasrc\bin\w32\ar.idb (ar.exe) - [IDA View-A]

File Edit Jump Search View Debugger Options Windows Help

Text unk_40D2

En 0101 COD 0101 DAT 0101 DA "s" * N X off # S N K /- ~ :

IDA View-A

```

.text:0040120C      mov     edi, [ebp+argc]
.text:0040120F      mov     [ebp+src], offset a_ar ; ".ar"
.text:00401216      jmp     short loc_401296
.text:00401218 ; -----
.text:00401218
.text:00401218  loc_401218:                ; CODE XREF: _main+A1↓j
.text:00401218      mov     eax, [ebx+4]
.text:0040121B      mov     dl, [eax+1]
.text:0040121E      sub     dl, 'a'
.text:00401221      jz     short loc_40122F
.text:00401223      sub     dl, 4
.text:00401226      jz     short loc_40123B
.text:00401228      sub     dl, 7
.text:0040122B      jz     short loc_401246
.text:0040122D      jmp     short loc_40126E
.text:0040122F ; -----
.text:0040122F

```

262144 32 8192 allocating memory for name pointers...

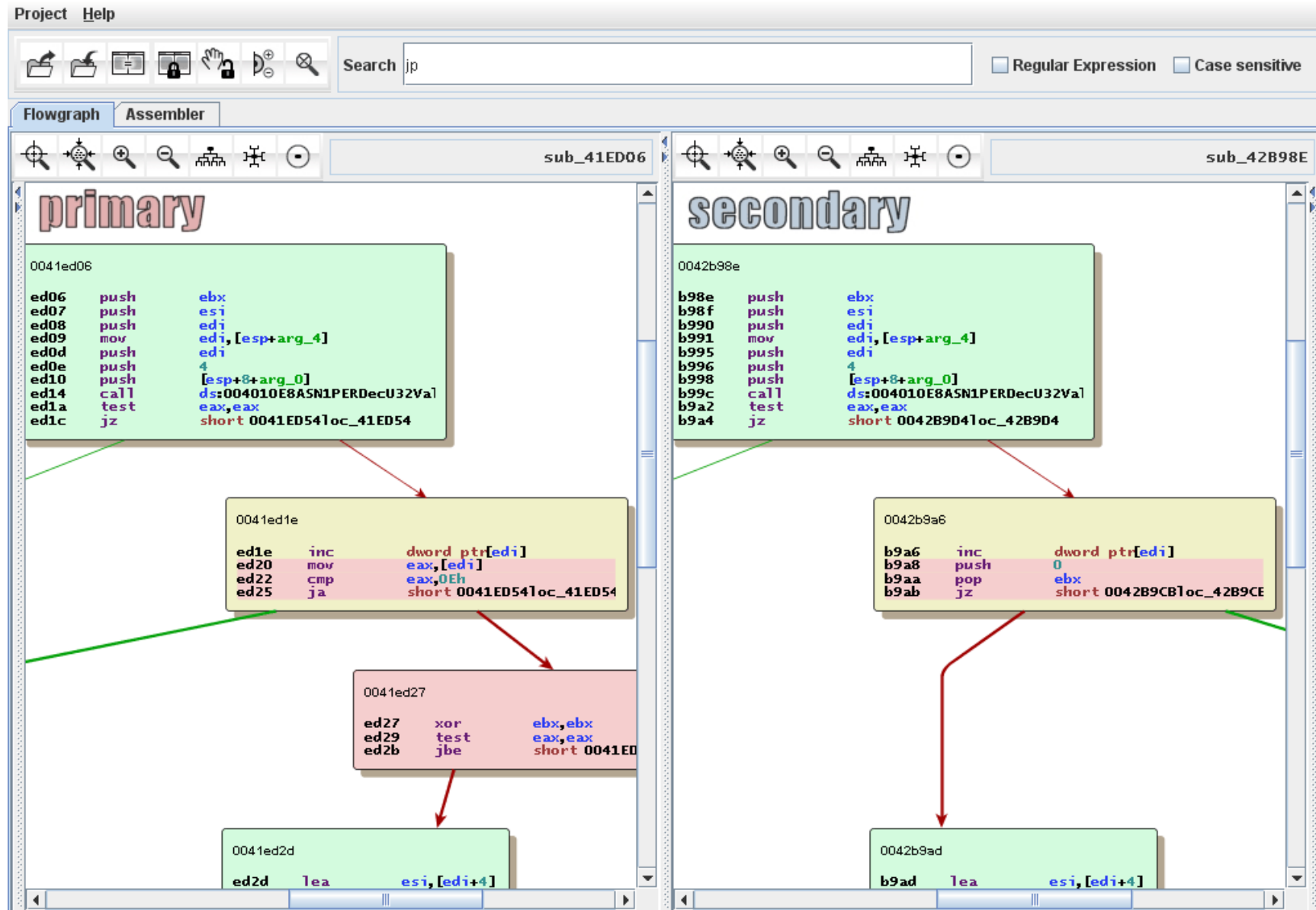
1425408 total memory allocated

Loading IDP module z:\idasrc\bin\w32\pc.w32 for processor metapc...OK
Loading type libraries...
Autoanalysis subsystem is initialized.
Database for file 'ar.exe' is loaded.
Compiling file 'z:\idasrc\bin\w32\idc\ida.idc'...
Executing function 'main'...

AU: idle Down Disk: 30GB 0000080F 0040120F: _main+F

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



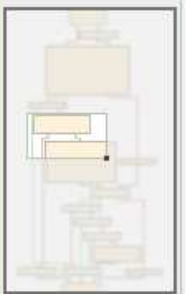
QueryRegValue(x,x,x,x,x) - kernel32.dll - zynamics BinNavi 3.0.0

View Graph Selection Search Plugins Window

QueryRegValue(x,x,x,x,x) *

Address Search Results

Overview



Graph Nodes

In	Out	No.	C...
0	2	7...	
2	2	7...	
1	1	7...	
2	2	7...	
1	1	7...	
3	0	7...	
1	2	7...	
5	1	7...	
1	1	7...	
1	2	7...	
1	2	7...	
1	1	7...	
1	1	7...	
1	2	7...	

Selection History

Selection Hist

Variables Calling Functions Register Tracking Special Instructions Code Bookmarks

- L arg_0 (2)
- L arg_10 (4)
- L arg_4 (6)
- L arg_8 (1)
- L arg_C (4)

```

7C8150E7 kernel32.dll::QueryRegValue(x,x,x,x,x)
7C8488C9 mov     eax, fs:[0x18]
7C8488CF push   ss:[ebp+arg_4]
7C8488D2 mov     eax, ds:[eax+0x30]
7C8488D5 push   0x8
7C8488D7 push   ds:[eax+0x18]
7C8488DA call   ds:[RtlAllocateHeap(x,x,x)] // RtlAllocateHeap(xxx)
7C8488E0 mov     edi, eax
7C8488E2 test   edi, edi
7C8488E4 mov     ss:[ebp+arg_C], edi
7C8488E7 jnz    loc_7C8488F3

7C8150E7 kernel32.dll::QueryRegValue(x,x,x,x,x)
7C8488F3 mov     ecx, ss:[ebp+arg_4]
7C8488F6 mov     edx, ecx
7C8488F8 mov     ds:[esi], edi
7C8488FA shr     ecx, byte 0x2
7C8488FD xor     eax, eax
7C8488FF rep stosd
7C848901 mov     ecx, edx
7C848903 and     ecx, 0x3
7C848906 rep stosw
7C848908 lea   eax, ss:[ebp+arg_4]

```

sub_801B981C - C2600-IS.BIN - zynamics BinNavi 3.0.0

View Graph Selection Search Plugins Window

sub_8021CC50 * sub_801D7B9C * sub_801B981C *

Address Search Results

Active Debugger
New Debugger (localhost:2...)

Register Values

??	00000050
r0	801B9ECC
sp	81114DA0
r2	80D30000
r3	00000001
r4	00000000
r5	80E9A0F0
r6	811013F4
r7	00000000
r8	00000000
r9	80DE0000
r10	68010000
r11	00000031
r12	22000020
r13	FFF48A24
r14	00000000
r15	00000000
r16	00000000
r17	00000000
r18	00000000
r19	00000000
r20	FFFFFFFF
r21	80D30000
r22	80E9A5D5
r23	80DC9598
r24	80E99CF0
r25	00000000
r26	00000000
r27	00000000
r28	00000000
r29	811003EC
r30	00000000
r31	811013F4
pc	801B9ED0
msr	00009432
cr	22400020
lr	801B9ECC
ctr	801A78C0
xer	4000DB09
fpscr	6A22B6E7
u_reg39	6A22B6E7
u_reg40	00000000
u_reg41	15DD48F2
u_reg42	00500000

801B981C C2600-IS.BIN::sub_801B981C

```

801B9EC0 mtlr    lr, r0
801B9EC4 mr      r3, r31
801B9EC8 blrl   byte 0, byte lr
801B9ECC lis    r9, word 0x80DE
801B9ED0 lws    r29, [r9+0xFFFFDA20]
801B9ED4 cmpwi  byte cr1, r29, word 0x0
801B9ED8 beq    byte cr1, word loc_801B9EF8

```

Memory Modules Threads Breakpoints Traces Bookmarks History

Target Memory

801B9000 - 801B9FFF (4.00 KB)

801B9000	8121 0008 3860 0010 9969 0EB8 4804 FDB5	0!..8`.0i.OH..0
801B9010	2C83 0000 8121 0008 9069 0A94 4186 0014	,D..0!..0i.OAO..
801B9020	3880 0000 4805 FD95 8061 0008 4800 0010	80..H..00a..H...
801B9030	3861 0008 4800 1109 3860 0000 8001 001C	8a..H...8`.0...
801B9040	7C08 03A6 83C1 0010 83E1 0014 3821 0018	!..000..00..8!..
801B9050	4E80 0020 9421 FF90 7C08 02A6 9321 0054	ND. 0!.0!..00!..T

Stack

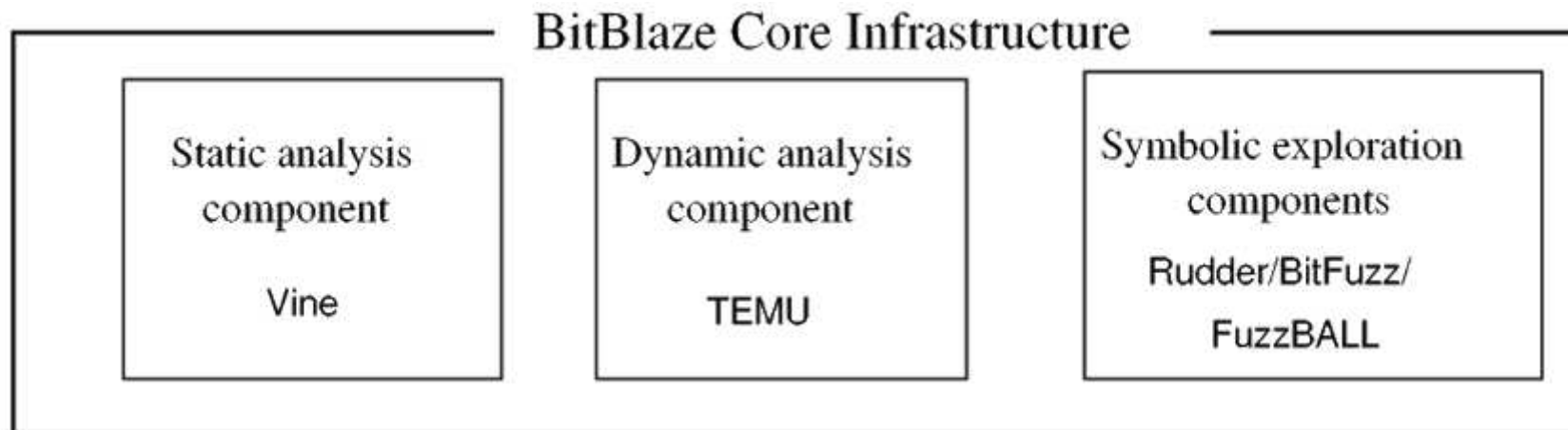
81114DA0	81114DE8
81114DA4	801B9ECC
81114DA8	81114D38
81114DAC	00000000
81114DB0	00000000
81114DB4	80E99B6C
81114DB8	81114DD0
81114DBC	801B0E1C
81114DC0	00000002
81114DC4	80DC9598

Current Offset: 00000000

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

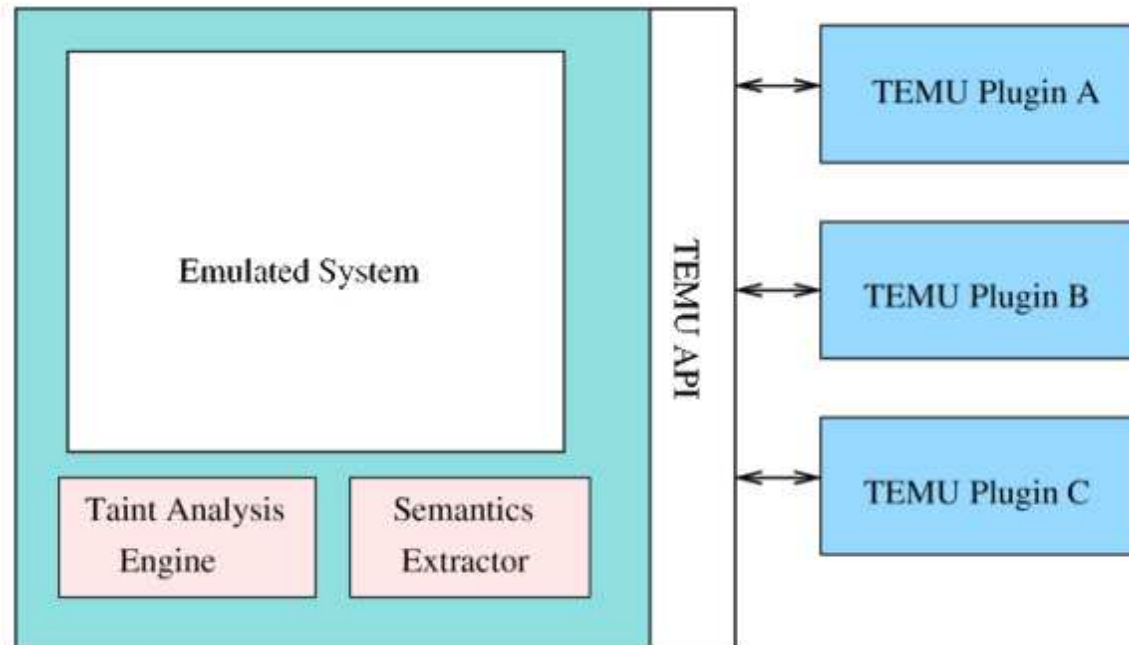


- Статический анализ кода (Vine)
- Динамический анализ кода (TEMU)
- Символьное исполнение кода
 - по трассе BitFuzz
 - по статическому представлению FuzzBALL

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 используемые в стоке.
- Анализа потоков данных и оптимизаций общего вида:
 - global value numbering
 - распространение констант
 - очистка 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



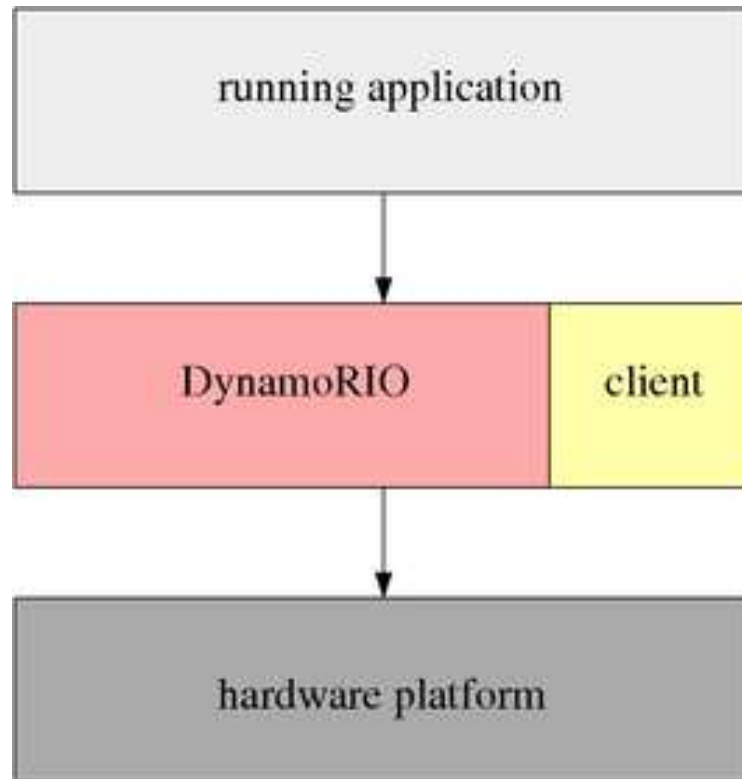
Снятие бинарной трассы уровня машинных инструкций

Анализ помеченных данных (Taint analysis)

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