

Диаген.

# Метод динамического контроля выполнения программы

Е.В. Маньков  
ООО «Газинформсервис»

*в соавторстве с Компанией Р.И. и Ковалевым В.В.*

# Малоосвещаемые вопросы ИБ функционирования ПО

## Непредумышленные:

- дефекты программирования и используемого ПО
- ошибки бизнес логики
- недокументированные возможности (НДВ)

## Предумышленные:

- модификации исполняемого кода (в памяти)
- атаки на исполняемый код (в памяти)

# Технология ИРИДА

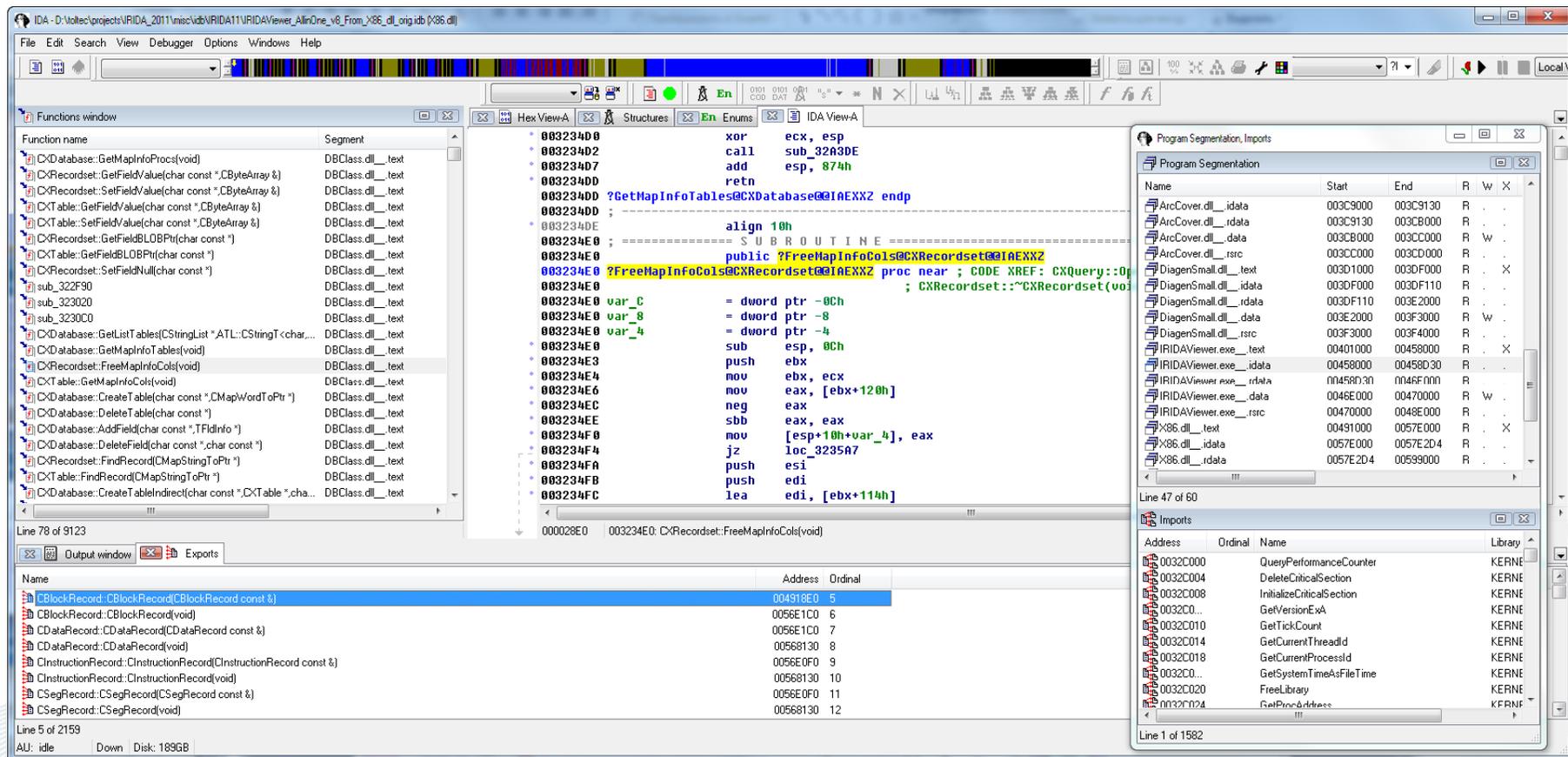
1. Дизассемблирование исполняемых модулей
2. «Склейка»: формирование общего адресного пространства, восстановление межмодульных связей (таблицы экспорта-импорта)
3. Построение модели программы (УГП всех подпрограмм)
4. Анализ статических маршрутов в подпрограмме, выявление структуры (управляющие конструкции) и нарушений структурированности
5. Формирование множества (доверенных) маршрутов – установка контрольных точек:
  - на вызовы подпрограмм (все, выборочные) – задача контроля/тестирования;
  - на передачи управления (например, в областях с нарушением структурированности) – задача контроля/тестирования;
  - на интересующие (критические по ИБ) участки кода – задача защиты от модификаций/атак;

## Технология ИРИДА

6. Создание грамматики описания множества (доверенных) маршрутов
7. Внедрение контрольных точек в контролируруемую/защищаемую программу
- 8. Генерация паспорта динамического контроля для исполняемого кода программы**
9. Динамический анализ маршрутов выполнения – задача контроля/тестирования
10. Сопоставление результатов СА и ДА, восстановление косвенных передач управления.
11. Уточнение и изменение набора контрольных точек и выполнение шагов 6-7 или результирующий паспорт: шаг 8
- 12. Контроль выполнения программы**

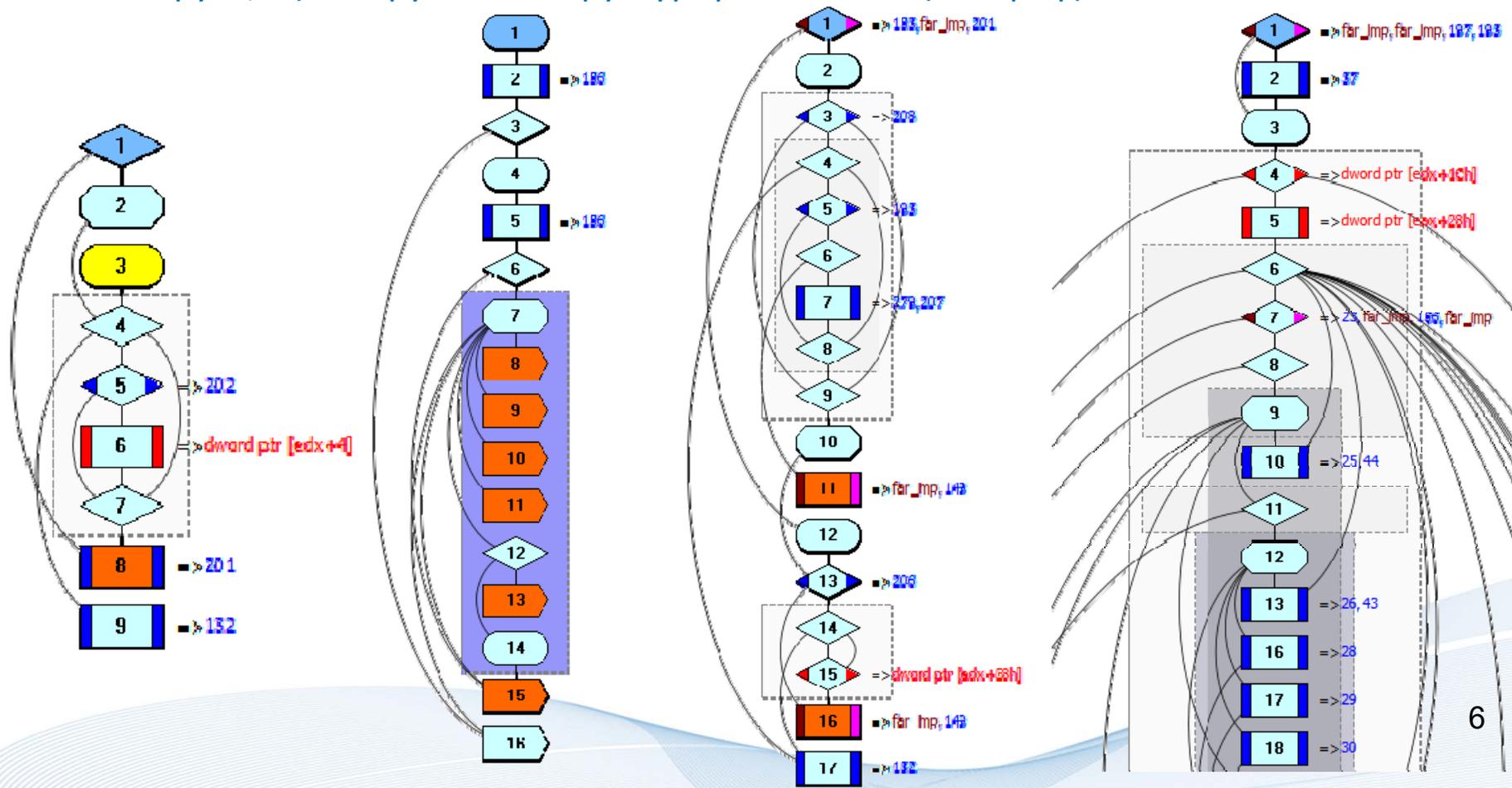
# Дизассемблирование, «Склейка»

1. Дизассемблирование исполняемых модулей (IDA Pro).
2. «Склейка»: формирование общего адресного пространства, восстановление межмодульных связей (таблицы экспорта-импорта) – разработанный плагин AllinOne под IDA Pro формирования единой idb.



# УГП - модели, управляющие структуры

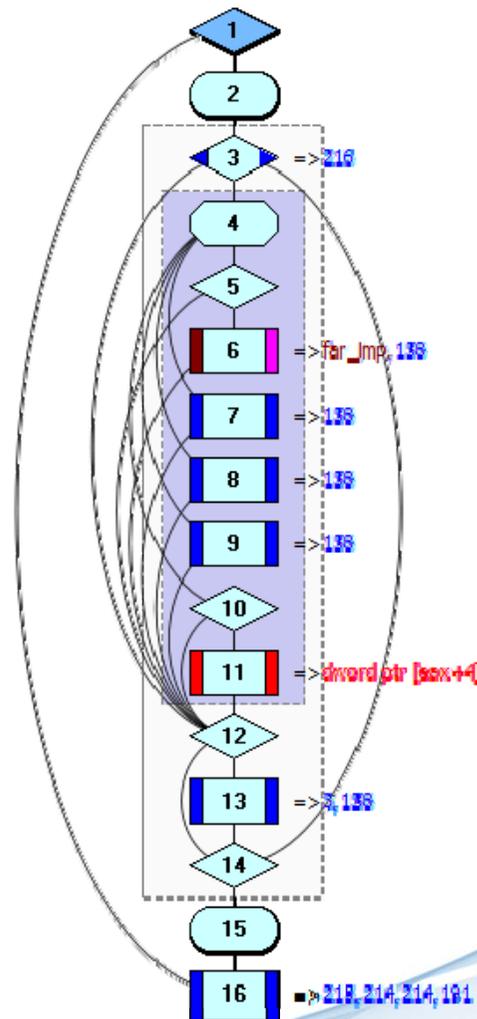
3. Построение модели программы (УГП всех подпрограмм).
4. Анализ статических маршрутов в подпрограмме, выявление структуры (управляющие конструкции) и нарушений структурированности (4-й граф).



# УГП - модели, управляющие структуры

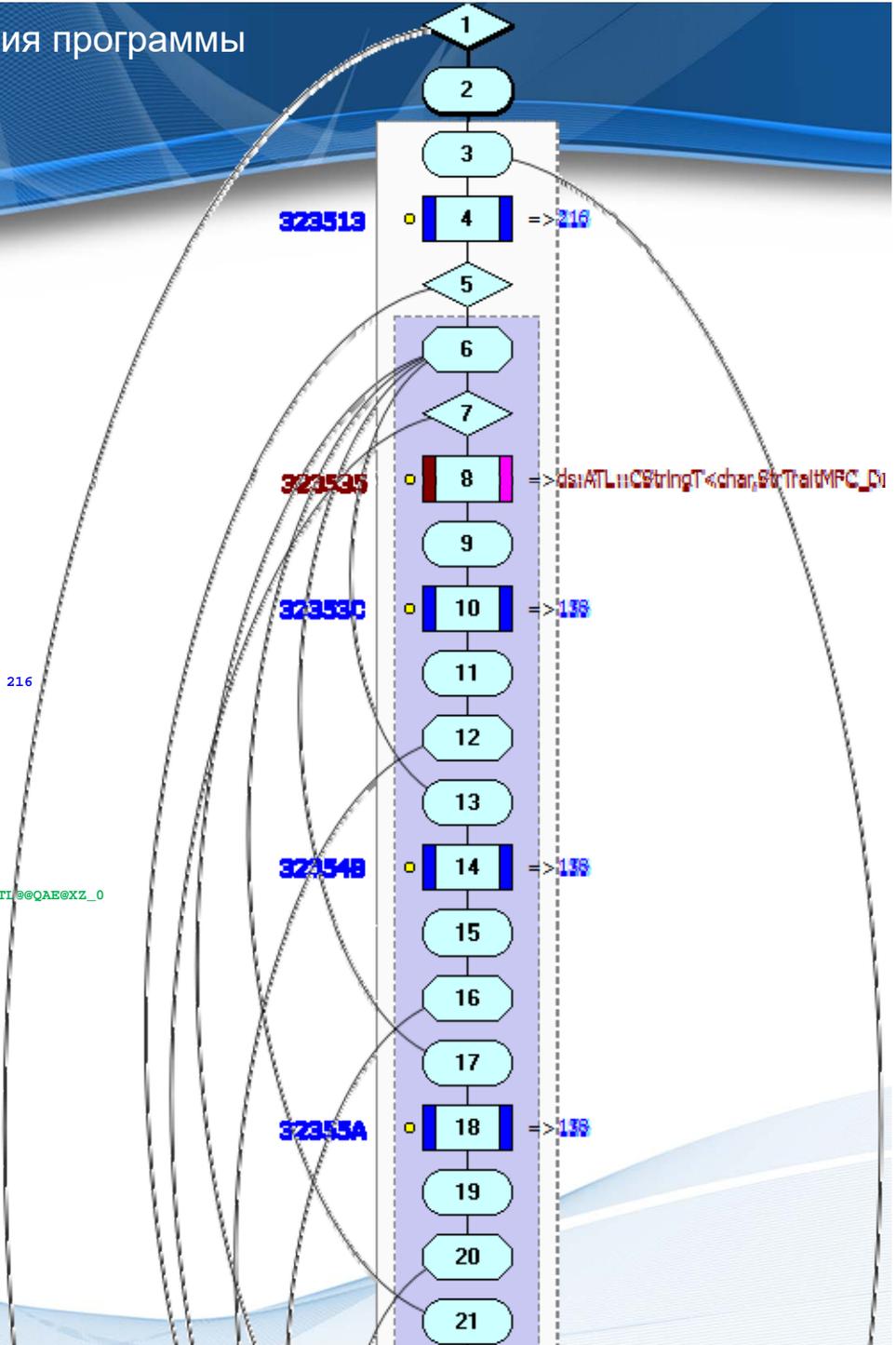
Пример подпрограммы на C++, дизассемблированный код и модель IDA

```
void CXRecordset::FreeMapInfoCols()
{
    POSITION CurPos;
    void * pVoid, * pValue;
    for(CurPos=m_oMapMemElements.GetStartPosition();CurPos!=NULL;)
    {
        m_oMapMemElements.GetNextAssoc(CurPos,pVoid,pValue);
        switch(((TFldInfo *)pVoid)->nFldType)
        {
            case FTString:
            case FTMemo:
                delete (CString*)pValue;
                break;
            case FTInteger:
                delete (int*)pValue;
                break;
            case FTLong:
            case FTCounter:
                delete (long int*)pValue;
                break;
            case FTSingle:
                delete (float*)pValue;
                break;
            case FTDouble:
                delete (double*)pValue;
                break;
            case FTBool:
                delete (BOOL*)pValue;
                break;
            case FTBLOB:
                delete (CByteArray*)pValue;
                break;
        };
        delete (TFldInfo *)pVoid;
    };
    m_oMapMemElements.RemoveAll();
    oInfoColList.RemoveAll();
    oValueColList.RemoveAll();
    oColList.RemoveAll();
}
```



# 5. Установка контрольных точек

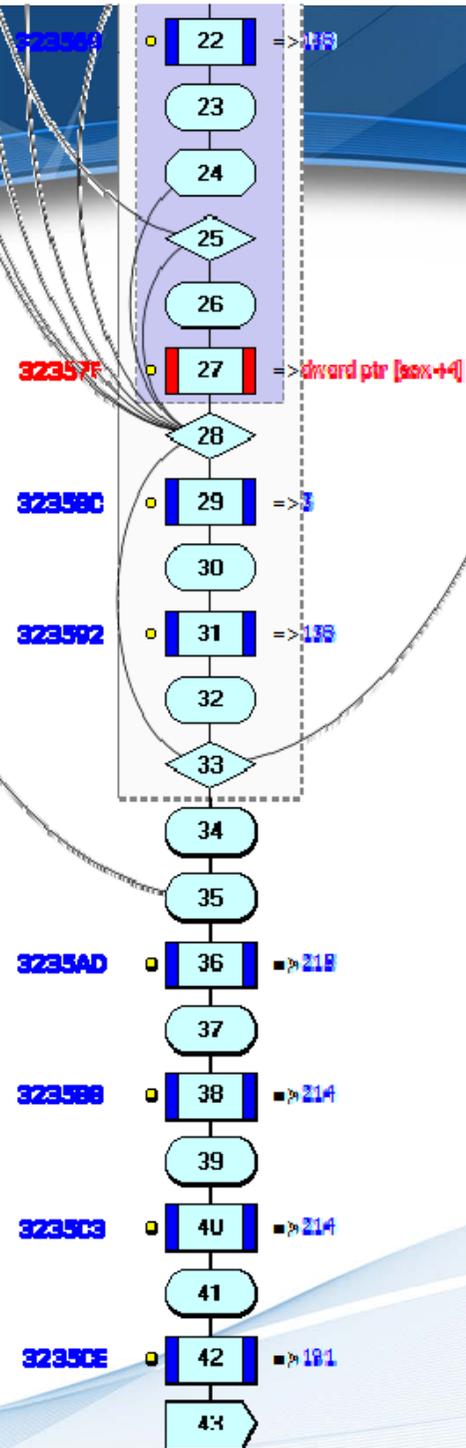
```
00323400 ; ----- SUBROUTINE -----  
00323400 ; --BLOCK--1  
00323400 public ?FreeMapInfoCols@CXRecordset@IAEXXZ  
00323400 ?FreeMapInfoCols@CXRecordset@IAEXXZ proc near  
00323400 var_C = dword ptr -0Ch  
00323400 var_8 = dword ptr -8  
00323400 var_4 = dword ptr -4  
00323400 sub esp, 0Ch  
00323403 push ebx  
00323404 mov ebx, ecx  
00323406 mov eax, [ebx+120h]  
00323408 neg eax  
0032340A sbb eax, eax  
0032340C mov [esp+10h+var_4], eax  
0032340E jz loc_3235A7  
00323410 ; --BLOCK--2  
00323410 push esi  
00323411 push edi  
00323412 lea edi, [ebx+114h]  
00323502 ; --BLOCK--3  
00323502 loc_323502: lea eax, [esp+18h+var_C]  
00323504 push eax  
00323505 lea ecx, [esp+1Ch+var_8]  
00323507 push ecx  
00323508 lea edx, [esp+20h+var_4]  
0032350A push edx  
0032350C mov ecx, edi  
00323510 ; --BLOCK--4  
00323510 call ?GetNextAssoc@CMapPtrToPtr@@QBEXAAPAU_POSITION@@AAPAXI@Z ; call 216  
00323512 ; --BLOCK--5  
00323512 mov eax, [esp+18h+var_8]  
00323514 mov eax, [eax+4]  
00323516 cmp eax, 8  
00323518 ja short loc_323582  
00323520 ; --BLOCK--6  
00323520 jmp ds:off_3235D8[eax*4]  
00323522 ; --BLOCK--7  
00323522 loc_32352B: mov ecx, [esp+18h+var_C]  
00323524 test ecx, ecx  
00323526 mov esi, ecx  
00323528 jz short loc_323582  
00323530 ; --BLOCK--8  
00323530 call ds:??1?@CStringT@DV?$StrTraitMFC_DLL@DV?$ChTraitsCRT@D@ATL@@@ATL@@@QAB@XZ_0  
00323532 push esi  
00323534 call ??3@YAXPAX@Z_0 ; call 138  
00323536 add esp, 4  
00323538 jmp short loc_323582  
00323540 ; --BLOCK--9  
00323540 loc_323546: mov ecx, [esp+18h+var_C]  
00323542 push ecx  
00323544 call ??3@YAXPAX@Z_0 ; call 138  
00323546 add esp, 4  
00323548 jmp short loc_323582  
00323550 ; --BLOCK--10  
00323550 loc_323555: mov edx, [esp+18h+var_C]  
00323552 push edx  
00323554 call ??3@YAXPAX@Z_0 ; call 138  
00323556 add esp, 4  
00323558 jmp short loc_323582  
00323560 ; --BLOCK--11  
00323560 loc_323564: mov eax, [esp+18h+var_C]  
00323562 push eax
```



# УГП – модель п/п

```

00323568 ; --BLOCK--22-----
00323569      call    ??3@YAXPAX@Z_0          ; call 138
0032356E ; --BLOCK--23-----
0032356E      add     esp, 4
00323571 ; --BLOCK--24-----
00323571      jmp     short loc_323582
00323573 ; --BLOCK--25-----
00323573 loc_323573:  mov     ecx, [esp+18h+var_C]
00323577      test    ecx, ecx
00323579      jz     short loc_323582
00323579 ; --BLOCK--26-----
0032357B      mov     eax, [ecx]
0032357D      push   1
0032357F ; --BLOCK--27-----
0032357F      call   dword ptr [eax+4]
00323582 ; --BLOCK--28-----
00323582 loc_323582:  mov     ecx, [esp+18h+var_8]
00323586      test    ecx, ecx
00323588      mov     esi, ecx
0032358A      jz     short loc_32359A
0032358C ; --BLOCK--29-----
0032358C      call   ??1TFldInfo@@QAE@XZ        ; call 3
00323591 ; --BLOCK--30-----
00323591      push  esi
00323592 ; --BLOCK--31-----
00323592      call   ??3@YAXPAX@Z_0          ; call 138
00323597 ; --BLOCK--32-----
00323597      add     esp, 4
0032359A ; --BLOCK--33-----
0032359A loc_32359A:  cmp     [esp+18h+var_4], 0
0032359F      jnz    loc_323502
003235A5 ; --BLOCK--34-----
003235A5      pop     edi
003235A6      pop     esi
003235A7 ; --BLOCK--35-----
003235A7 loc_3235A7:  lea    ecx, [ebx+114h]
003235AD ; --BLOCK--36-----
003235AD      call   ?RemoveAll@CMapPtrToPtr@@QAE@XZ
003235B2 ; --BLOCK--37-----
003235B2      lea    ecx, [ebx+14Ch]
003235B8 ; --BLOCK--38-----
003235B8      call   ?RemoveAll@CMapStringToPtr@@QAE@XZ
003235B8 ; --BLOCK--39-----
003235B8      lea    ecx, [ebx+130h]
003235B8 ; --BLOCK--40-----
003235B8      call   ?RemoveAll@CMapStringToPtr@@QAE@XZ
003235C3 ; --BLOCK--41-----
003235C3      lea    ecx, [ebx+168h]
003235C8 ; --BLOCK--42-----
003235C8      call   ?RemoveAll@CStringList@@QAE@XZ
003235D3 ; --BLOCK--43-----
003235D3      pop     ebx
003235D4      add     esp, 0Ch
003235D7      retn
003235D7 ?FreeMapInfoCols@CXRecordset@@@IAEXXZ endp
003235D8 ; --DATA BLOCK 1 (SWITCH 1 MAP)-----
003235D8 off_3235D8 dd offset loc_32352B          ; DATA XREF: CXRecordset::FreeMapInfoCols(void)+44r
003235D8      dd offset loc_32352B
003235D8      dd offset loc_323546
003235D8      dd offset loc_323555
003235D8      dd offset loc_323555
003235D8      dd offset loc_323555
003235D8      dd offset loc_323564
003235D8      dd offset loc_323546
003235D8      dd offset loc_323555
003235D8      dd offset loc_323555
003235D8      dd offset loc_323573
003235FC      align 10h
    
```



## 6. Формирование грамматики

### Грамматика в терминах блоков

Route\_78 :

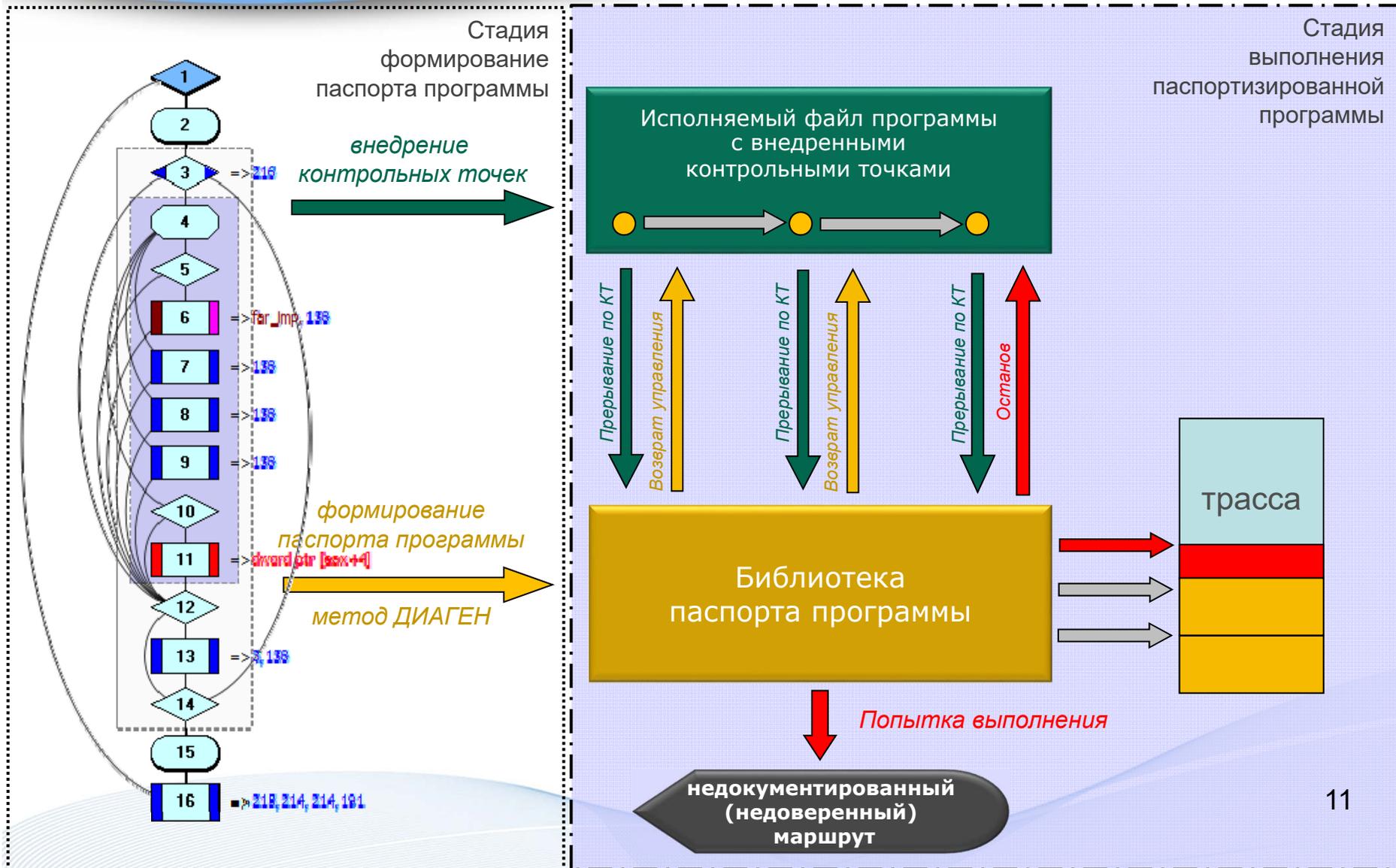
```
( 4  
  (  
    8 10  
    | 14  
    | 18  
    | 22  
    | 27  
  )?  
  (29 31)  
)*  
(36 38 40 42) ;
```

### Грамматика в терминах КТ (паспорт)

Route\_78 :

```
(KT_323513 m4 KT_323513  
  (  
    KT_323535 m8 KT_323535  
    KT_32353C m10 KT_32353C  
    | KT_32354B m14 KT_32354B  
    | KT_32355A m18 KT_32355A  
    | KT_323569 m22 KT_323569  
    | KT_32357F m27 KT_32357F  
  )?  
  (KT_32358C m29 KT_32358C  
    KT_323592 m31 KT_323592)  
)*  
(KT_3235AD m36 KT_3235AD  
  KT_3235B8 m38 KT_3235B8  
  KT_3235C3 m40 KT_3235C3  
  KT_3235CE m42 KT_3235CE) ;  
// адрес вызываемой ближней п/п-мы  
m4 : {NEAR_CALL(0x00328513)} ;  
// адрес вызываемой дальней п/п-мы  
m8 : {FAR_CALL(00407000h)} ;  
...
```

# Механизм работы метода



## Диаген. Особенности

- Контролируемая программа «не знает» куда она передает управления в точках контроля!
- В точках контроля программа всегда вызывает одну и ту же функцию Hook без параметров, экспортируемую паспортом
- Функция Hook получает с вершины стека вызовов процедур адрес возврата – адрес контрольной точки и передает это значение на вход конечного автомата (КА) паспорта
- КА определяет по своему текущему состоянию допустимость нахождения контролируемой программы в данной точке в данный момент времени ее выполнения; в случае допустимости КА осуществляет вызов уже реальной подпрограммы и переходит в следующее свое состояние, либо прерывает выполнение контролируемой программы

## Диаген. Особенности

- Таким образом, куда передавать управление сообщает программе ее паспорт на основании состояния своего КА
- Взламывать и/или атаковать (с целью изменения потока управления) целесообразно прежде всего модуль паспорта, а не контролируемую программу
- Для взлома паспорта необходимо решить задачу восстановления грамматики конечного автомата в терминах установленных контрольных точек
- Для усложнения взлома конечный автомат может быть обфускирован, зашифрован и привязан к токenu
- Паспорт может выполняться на доверенной системе, в отличие от контролируемой программы

# IRIDA 1.1

Проект: Calc\_test; Сессия: Session 2 - IRIDA Viewer

Файл Правка Анализ Вид Язык Помощь

**Подпрограммы**

N <sup>o</sup>	Метка п/п-мы
1	sub_401000
2	main
3	sub_4012E0
4	?fail@ios_base@std@@QBFI
5	unknown_libname_1
6	sub_401340
7	sub_401360
8	sub_401380
9	sub_4013A0
10	sub_4013C0
11	sub_4013E0
12	sub_401400
13	??21?ncala@std@@VPAE@VZ

Найти... Перейти...

**Линейные участки**

N <sup>o</sup>	Метка блока
1	sub_401000
2	_1
3	_2
4	_3
5	_4
6	_5
7	_6
8	_7
9	_8
10	_9
11	_10
12	loc_40108F
13	loc_4010A9

Найти... Перейти...

WinGraph32 - Subroutine: sub\_401000

18 95% (0,0) 23 nodes, 96 edge segments, 0 cros

**Инструкции**

Адрес	Имя	Оп. 1	Оп. 2	Оп. 3	Стек	N <sup>o</sup> строки
00401000	push	ebp			000	76
00401001	mov	ebp	esp		004	77
00401003	sub	esp	20h		004	78
00401006	mov	[ebp + var_1C]	ecx		024	79
00401009	mov	ecx	[ebp + var_1C]		024	80

Информация  
Тек. п/п-ма: sub\_401000  
номер: 1  
начало: 61  
конец: 151

Минимаксное покрытие вершин управляющего графа для подпрограммы 'sub\_401000'

N <sup>o</sup>	Показ.	Точки трассировки	Путь
1	<input type="checkbox"/>	<input type="checkbox"/>	1 2 3 4 5 6 7 8 9 10 11 21 22 23
2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	1 2 3 4 5 6 7 8 9 10 11 12 13 14 23
3	<input type="checkbox"/>	<input type="checkbox"/>	1 2 3 4 5 6 7 8 9 10 11 15 16 17 23
4	<input type="checkbox"/>	<input type="checkbox"/>	19 20 23
5	<input type="checkbox"/>	<input type="checkbox"/>	1 2 3 4 5 6 7 8 9 10 11 18

Ограничения на установку точек трассировки

Без ограничений  Устанавливать только на след. переходы:

- Близкий вызов  Передача управления на блок с данными
- Дальний вызов  Передача управления в тело другой подпрограммы
- Регистровая передача управления  Передача управления в свернутую подпрограмму
- Передача управления путем явной адресации

Установить и показать

Subroutine 'sub\_401000' (1) Call Tree

```

graph TD
    1[0040100C call sub_401AA0] --> 2[00401019 call sub_401AE0]
    2 --> 3[88 00401AEA call sub_401C]
    3 --> 4[111 00401CFA call sub]
    4 --> 5[135 004021DA ce]
    5 --> 6[112 00401D1A call sub]
    6 --> 7[145 00402323 cal]
    7 --> 8[3 00401021 call sub_401AA0]
    8 --> 9[4 0040102E call sub_401AE0]
    9 --> 10[5 00401042 jmp ds:off_4010AB]
    10 --> 11[6 00401059 call sub_401AC0]
    11 --> 12[87 00401ACE call sub_401C]
    12 --> 13[106 00401C8D call sub]
    13 --> 14[107 00401C97 call sub]
    14 --> 15[108 00401CB0 call sub]
    15 --> 16[146 00402357 cal]
    16 --> 17[418 004046B]
    17 --> 18[419 004046C]
    18 --> 19[109 00401CC8 call sub]
    19 --> 20[110 00401CD7 call sub]
    20 --> 21[136 0040221C ca]
    21 --> 22[137 00402235 cal]
    22 --> 23[138 0040223E ca]
    23 --> 24[139 00402259 cal]
    24 --> 25[140 0040226D ca]
    25 --> 26[141 00402274 ca]
    
```

П/п-мы, вызывающие данную п/п-му

N <sup>o</sup>	Метка п/п-мы	Кол-во
2	_main	1

## IRIDA 1.1

- Синтаксический разбор и анализ ассемблерных кодов и дизассемблированных листингов, и построение модели программы в БД MS Access/ MySQL
- Поддержка ассемблеров платформ Intel x86 и MIPS
- Структурирование и упорядочивание моделей подпрограмм; выявление нарушений структурированности программного кода
- Построение дерева вызова подпрограмм
- Формирование описания управляющего графа в виде грамматики в терминах контрольных точек дальних и ближних вызовов подпрограмм
- Создание паспортов потоков управления в исполняемых модулях (метод «Диаген»)
- Встраивание в контролируемую/исследуемую программу контрольных точек и подключение к ней паспорта
- Контроль выполнения программы посредством паспорта программы
- Получение трассы выполнения и ее наложение на модельное представление; как результат решение следующих задач:
  - покрытия кода;
  - получение статистики по вызовам и передачам управления;
  - выявление невызываемых подпрограмм.

# Диаген. Метод динамического контроля выполнения программы

## IRIDA 2.0. state of the art

The screenshot displays the IRIDA 2.0 interface with several key components:

- Modules:** A table listing loaded modules with their addresses and titles.
- Subs-IRIDA-ASM:** A table listing subroutines with their addresses and titles.
- Component Diagram:** A graph showing the relationships between modules and subroutines, with edges labeled with counts.
- Instructions-IRIDA-ASM:** A table showing the assembly instructions for a specific subroutine.

num	title	addr_beg	addr_end
1	DBClass.dll	3280896	3358720
2	SubAnalyze.dll	3411968	3436544
3	SubAnalyzeOpti...	3477504	3518464
4	NodeCover.dll	3543040	3592192
5	SubCallTree.dll	3608576	3637248
6	Diagen.dll	3674112	3821568
7	CommonLang.dll	3870720	3907584
8	ArcCover.dll	3936256	3985408
9	DiagenSmall.dll	4001792	4145152

num	title	addr_beg	addr_end
89	C:\Database:Dele...	3299480	3299464
90	C:\Query:Open(v...	3299472	3299744
91	C\T able:FindRec...	3299744	3299944
92	C\T able:GetID(v...	3299952	3300020
93	FieldsFromList(CS...	3300032	3300200
94	C\Database:Che...	3300208	3300616
95	C\T able:AddFro...	3300624	3301952
96	sub_32623F	3301951	3302020
97	C\Database:Com...	3302192	3303360
98	sub_3267C0	3303360	3303372

id	block_num	num	address	title	OP1	OP2	OP3	OP4	stack	comments	repeatable_comments	auto_comments
1	511327	68	5836	3301130	mov	edx			92			// Move Data (from to)
2	511328	69	5837	3301133	mov	esi			92	[ebp+var_18]		// Move Data (from to)
3	511329	69	5838	3301135	lea	eax			92			// Load Effective Address
4	511330	69	5839	3301138	push	eax			92			// Push Operand onto the Stack
5	511331	69	5840	3301139	push	esi			96			// Push Operand onto the Stack
6	511332	69	5841	3301140	mov	ecx			100	edi		// Move Data (from to)
7	511333	69	5842	3301142	mov	[ebp+var_1C]			100	edx		// Move Data (from to)
8	511334	70	5843	3301145	call	C:\Recordset:SetFieldValue(char const *,long &)			100			// Call Procedure
9	511335	71	5844	3301150	add	[ebp+var_24]			92	1		// Add
10	511336	72	5845	3301154	jmp	loc_325DA1			92			// Jump

## IRIDA 2.0. Изменения

- Полный цикл работы с многомодульным ПО, начиная с этапа формирования БД в среде дизассемблера (собственный плагин под дизассемблер IDA Pro) и заканчивая ДА и паспортом программы
- Плагиновая архитектура, собственный SDK
- Тесная интеграция с дизассемблерами (HexRays IDA Pro), поддержка его встроенных отладчиков
- Поддержка различных форматов исполняемых файлов (PE, ELF, MIPS, etc.)
- Трассировка и динамический анализ внешними средствами

## IRIDA 2.0. Изменения

- Сопоставление результатов СА и ДА
- Уточнение модели ПО по результатам сопоставления результатов СА и ДА
- Выявление участков модифицируемого кода
- Выявление нарушений структурированности
- IRIDA Sources – интегрированный в IRIDA модуль (исходные тексты программ)
- Поддержка пользовательских изменений модели ПО
- Поддержка различных СУБД: Oracle, MySQL, PostgreSQL
- Кроссплатформенность: Windows/Linux/MacOS

## To Do

1. Внедрение точек на условные передачи управления
2. Переход с грамматического аппарата типа LL(k) на LL(\*) для разрешения проблемы глубины поиска альтернатив в конечном автомате
3. Интеграция метода «Диаген» с методом семантической верификации на базе теории размерностей и использование атрибутивных денотационных грамматик

## To Do

4. Реализация общего паспорта многомодульного ПО: N исполняемых модулей (в рамках 1 адресного пространства) – 1 библиотека паспорта
5. Реализация выполнения автомата в потоке другого ядра (для многоядерной/многoproцессорной архитектуры); в общем оптимизация по скорости выполнения
6. Защита исполняемого кода паспорта программы посредством шифрования, обфускации, использования токенов, контроля запуска отладчиков в процессе выполнения паспорта

## To Do

7. Реализация внедрения контрольных точек не на уровне исполняемых кодов, а на этапе создания исполняемого кода компилятором (этапы компиляции и линковки) – профилирование программы в терминах КТ

### Причины необходимости такой реализации

1. Недостаточность на уровне исходных текстов
2. Техническая сложность внедрения на некоторые передачи управления (условные передачи управления) на уровне исполняемых кодов
3. Современные антивирусы идентифицируют технологию Диаген как хакерскую

## To Do

7. Реализация внедрения контрольных точек не на уровне исполняемых кодов, а на этапе создания исполняемого кода компилятором (этапы компиляции и линковки) – профилирование программы в терминах КТ

### Особенности такой реализации

1. Необходимость отдельной реализации для различных компиляторов под различные языки программирования
2. Как следствие, необходимость поддержки с выходом новых версий компиляторов
3. **Возможность внедрения КТ на любой участок контролируемого кода**

# Выводы

- НДВ всегда присутствуют в ПО
- Контроль отсутствия/наличия НДВ в ПО требует специализированных методик и инструментов
- Создание условий невозможности проявления НДВ (запрета выполнения недоверенного кода) реализуемо программно на практике
- Паспортизация ПО – один из путей решения проблемы защиты ПО как от скрытых НДВ, так и от атак на ПО в памяти

# Выводы

- ПО с исходными текстами целесообразно (начинать) контролировать еще на этапе разработки (но недостаточно только лишь на этом этапе)
- ПО без исходных текстов можно паспортизировать на уровне исполняемых кодов с помощью метода «Диаген»
- Для решения задачи комплексной защиты ПО в процессе его функционирования необходима интеграция метода как с существующими (шифрование, обфускация, tokens), так и с перспективными (семантический контроль информационных потоков) методами защиты ПО

# Спасибо за внимание

ООО «Газинформсервис»  
[www.gaz-is.ru](http://www.gaz-is.ru)  
Санкт-Петербург

*Компаниец Радион Иванович  
Ковалев Виктор Васильевич  
Маньков Евгений Викторович*

*kompaniec-r@gaz-is.ru  
kovalev-v@gaz-is.ru  
mankov-e@gaz-is.ru*