ООО «Газинформсервис»

Методы и инструменты для статического и динамического анализа программного обеспечения.

Р.И.Компаниец
В.В.Ковалев
Дрозд Ю.А.

19.03.2015 Солнечногорск

Методы защиты программ

ГАЗИНФОРМСЕРВИС

В общем случае методы защиты программ подразделяются на две группы.

В первой – методы, создающие условия, затрудняющие проявление НДВ.

Во второй – методы, позволяющие выявить аномальное поведение программы и предотвратить возможные негативные последствия.

Методы первой группы обычно базируются на различных моделях разграничения доступа и создании замкнутой доверенной среды вычислений.
Методы второй группы связывают с использованием систем обнаружения вторжений (IDS), основанных на эвристиках нормального (или аномального) поведения конкретной программы.
Во время выполнения всех программ IDS сравнивает системные вызовы с имеющимися шаблонами. И если какая-то программа предпринимает попытку системного вызова, не предусмотренного в ее описании, защитная система блокирует подозрительные действия.

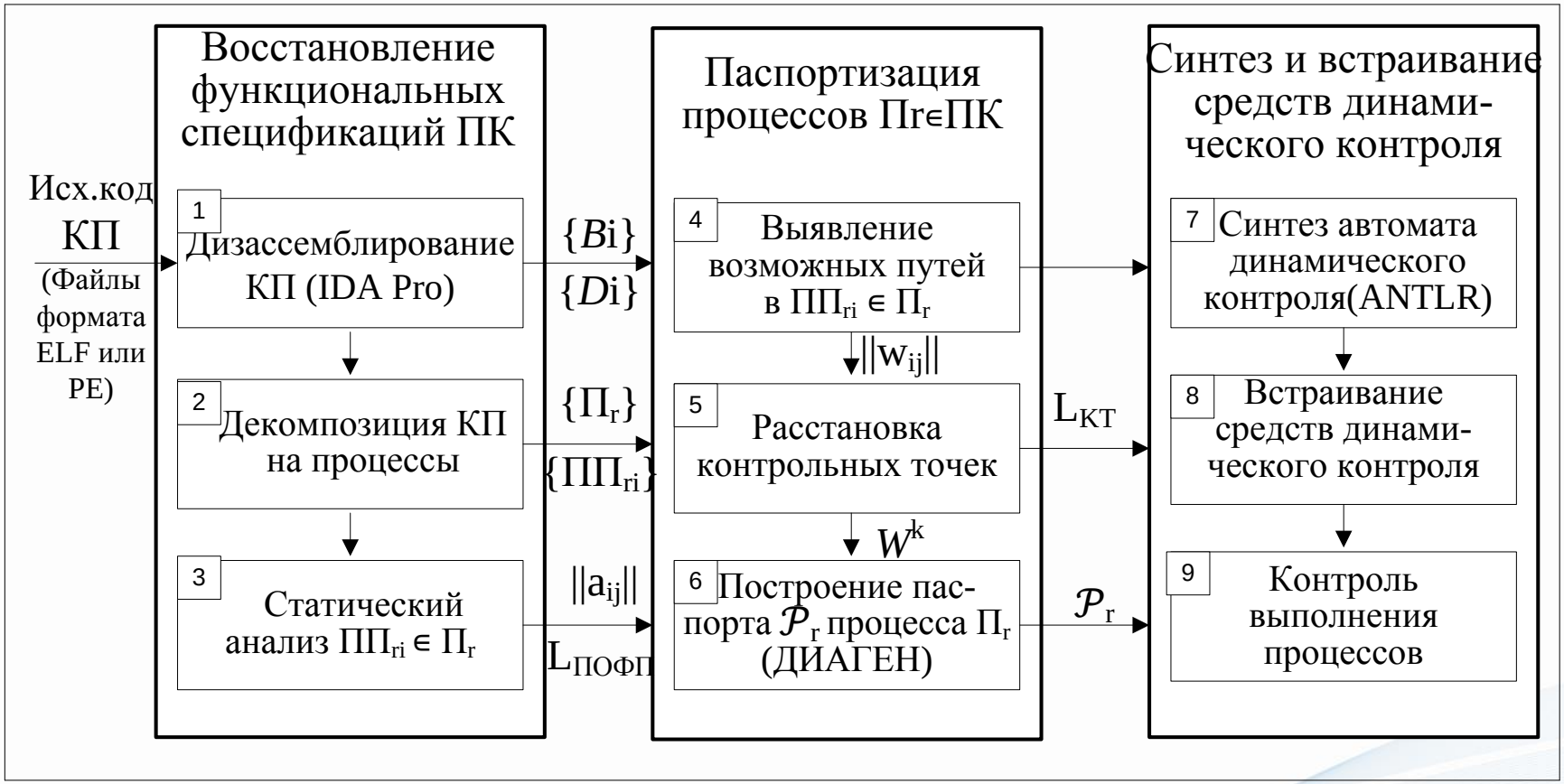
Предлагаемый метод контроля и предотвращения недекларированного выполнения команд дополняет идею метода анализа программ, проведением контроля не только системных вызовов, но и любых других вызовов функциональных объектов на всю глубину их вложенности. При этом контролируется не только состав вызовов, но и порядок их следования в строгом соответствии с алгоритмом выполнения программы.

Назначение, цели и задачи метода контроля и предотвращения недекларированного выполнения программ

Данный метод предназначен для проведения углубленного статического и последующего динамического анализа исполняемых кодов программ с целью выявления и/или предотвращения использования НДВ.

Задача контроля и предотвращения недекларированного выполнения программных комплексов, представленных исполняемым кодом, **решается в три этапа.**

Взаимосвязь этапов технологии контроля и предотвращения недекларированного выполнения программ



Первый этап контроля и предотвращения недекларированного выполнения ПК

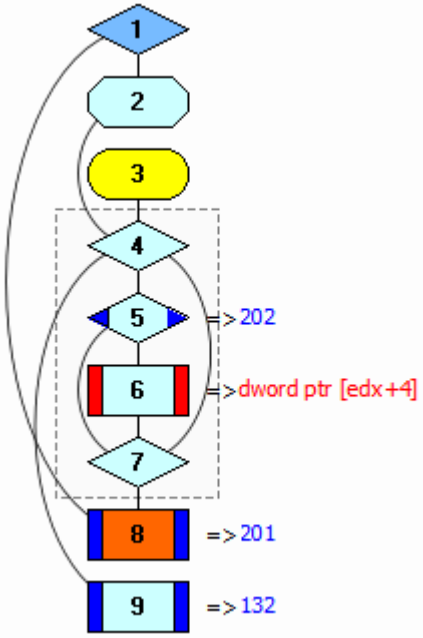
Цель первого этапа - восстановление функциональных спецификаций компонентов ПК путем выделения реализуемых в нем процессов $\Pi = \{\Pi_r\}$ и составляющих их подпрограмм $ПП_{ri} \in \Pi_r$.

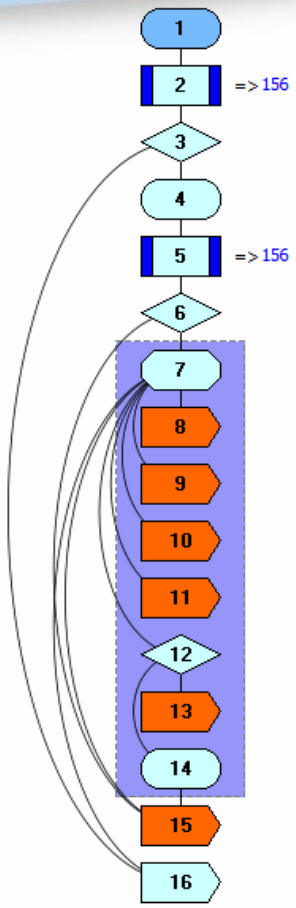
Достижение этой цели обеспечивается последовательным решением следующих задач:

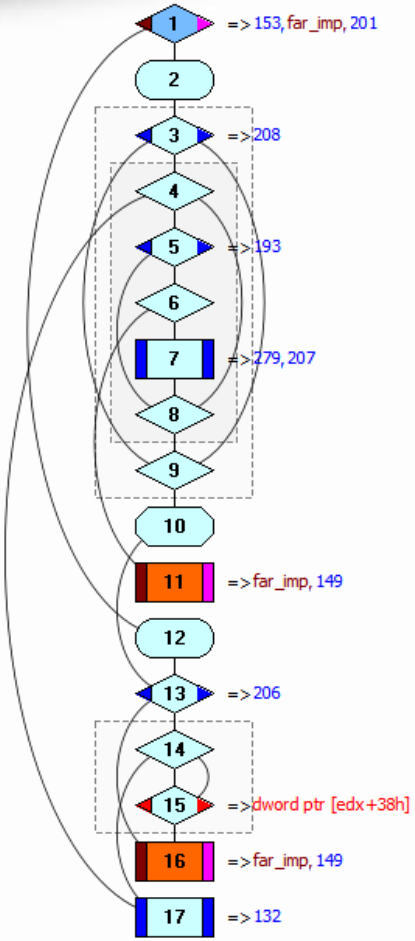
- Дизассемблирование исследуемого ПК .
- Декомпозиция ПК на процессы $\Pi = \{\Pi_r\}$.
- Статический анализ $ПП_{ri} \in \Pi_r$.

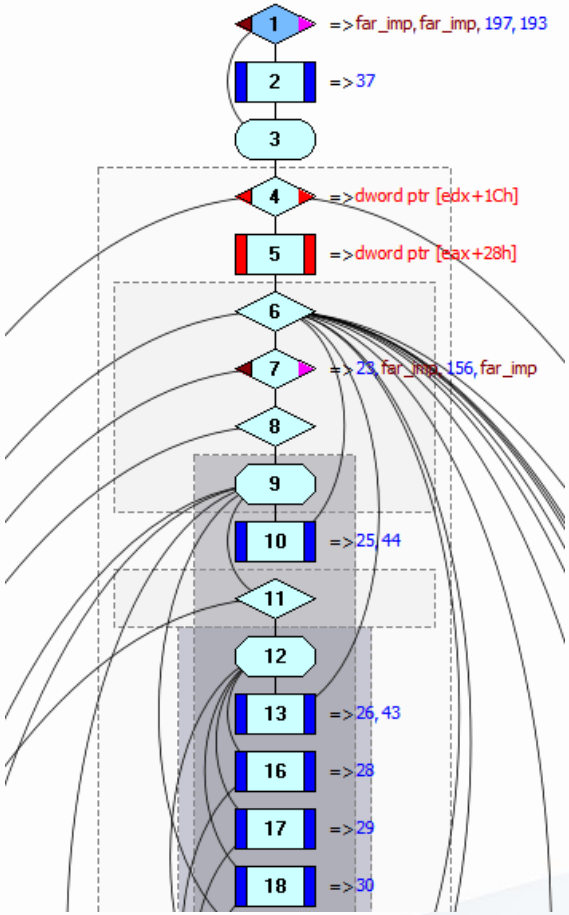
Исходный код ПК дизассемблируется и подвергается факторизации, что обеспечивает построение топологических (древовидных и графовых) моделей для последующих статического и динамического анализа компонентов ПК. На выходе первого этапа мы получаем дизассемблированный код ПК, в котором можно выделять процессы, составляющие их ПП, статические связи по управлению и данным между ними и линейными фрагментами ПП.

Примеры различных упорядоченных графов П









Статический анализ подпрограмм

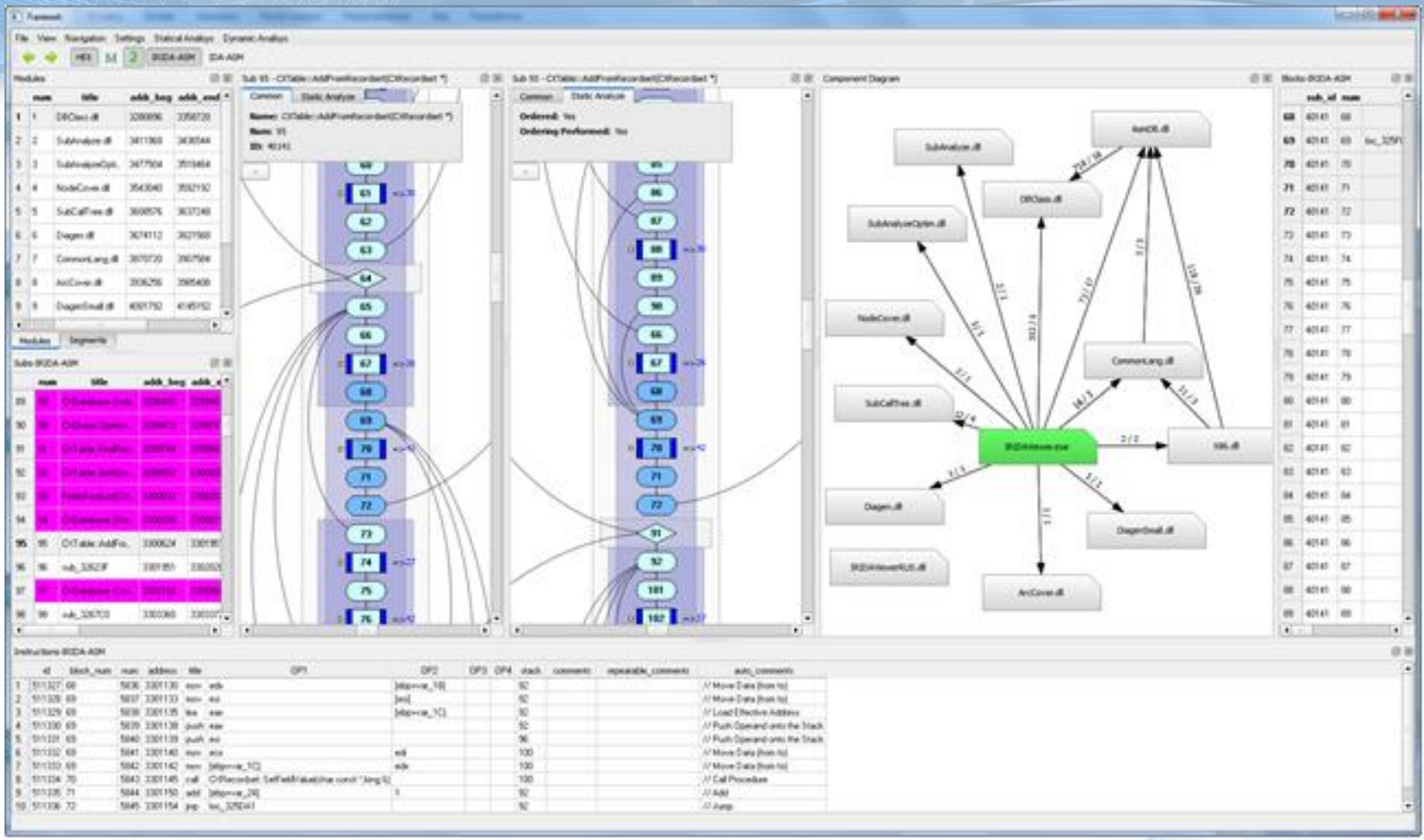
Статический анализ ПП имеет цель выявить **потенциально опасные фрагменты программы (ПОФП)**, последующий углубленный анализ которых (включая выполнение кода) позволяет идентифицировать их функциональность и сделать заключение о наличии (отсутствии) НДВ.

В основе технологии анализа лежит выявление нарушений структурированности управляющей структуры ПП вследствие чего естественной анализируемой моделью ПП становится её управляющий граф (УГ).

К ПОФП относятся следующие нарушения структурированности программ:

- наличие дополнительных точек входа и/или выхода для ПП, циклов и переключателей;
- наличие зацепленных управляющих структур;
- наличие элементов обфускации (преднамеренного усложнения и искажения управляющих и информационных связей в ПП);
- наличие висячих вершин;
- наличие «заплаток» (исправления в объектном коде);
- наличие изолированных вершин (фрагментов кода ПП);
- наличие необоснованной косвенной передачи управления (косвенная передача управления вне текущей ПП – висячая вершина).

Главное окно ИК ИРИДА 2.0



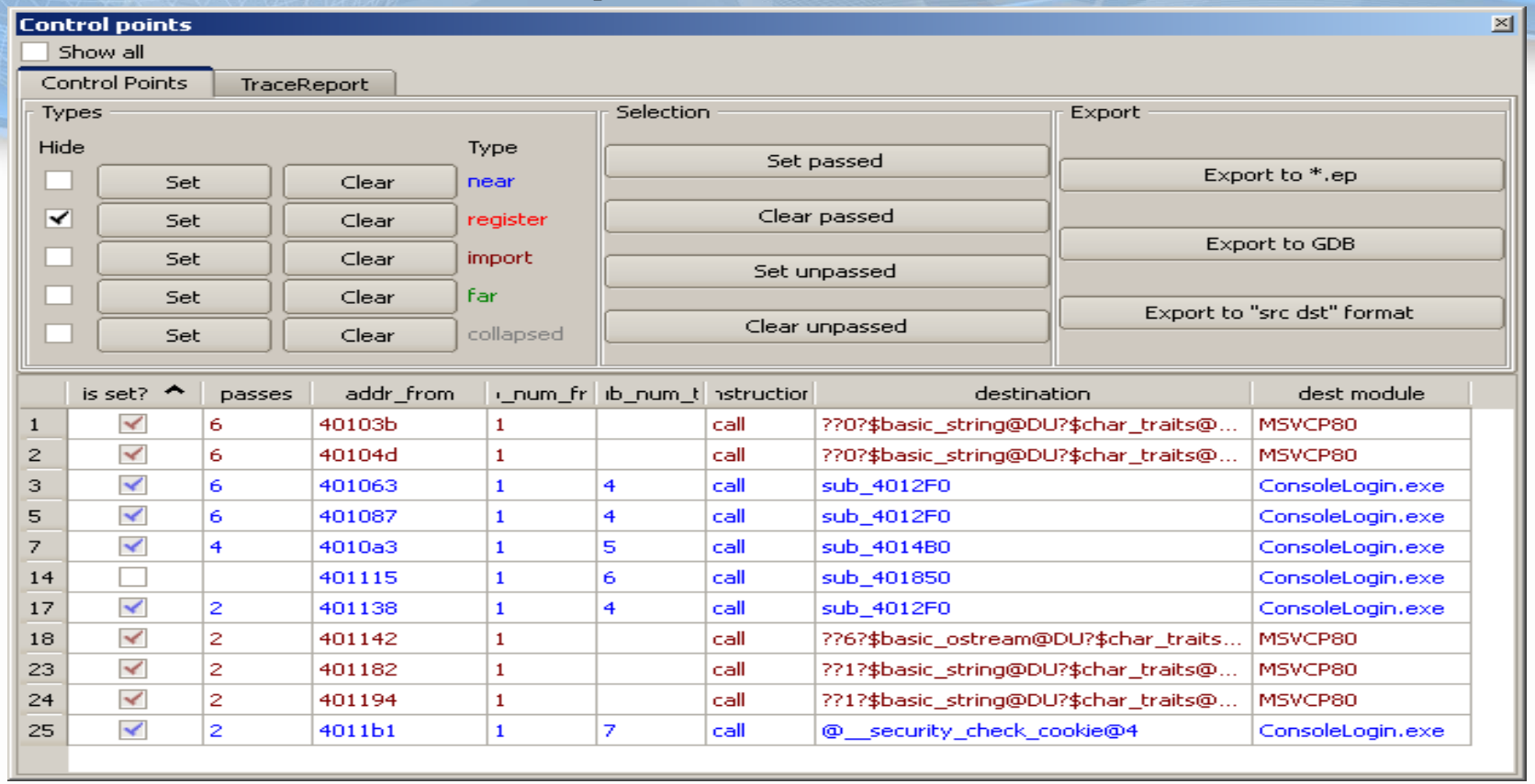
Второй этап контроля и предотвращения недекларированного выполнения ПК

Цель второго этапа - паспортизация выявленных процессов {$П_r$} в терминах возможных путей выполнения составляющих их ПП, что достигается решением следующих задач:

- Выявление возможных путей выполнения задействованных процессом $П_r$ подпрограмм $П_{ri}$.
- **Расстановка контрольных точек** (КТ) и редукция выявленных путей.
- **Построение паспорта процесса** $П_r$.

Под паспортом процесса будем понимать формализованное представление возможных путей его выполнения, выраженных в заданных терминах (вершинах, дугах, вызываемых ПП, КТ и т.д.).

Расстановка контрольных точек ИК «IRIDA» 2.0



	is set?	passes	addr_from	ı_num_fr	ıb_num_t	ıstructior	destination	dest module
1	☑	6	40103b	1		call	??0?$basic_string@DU?$char_traits@...	MSVCP80
2	☑	6	40104d	1		call	??0?$basic_string@DU?$char_traits@...	MSVCP80
3	☑	6	401063	1	4	call	sub_4012F0	ConsoleLogin.exe
5	☑	6	401087	1	4	call	sub_4012F0	ConsoleLogin.exe
7	☑	4	4010a3	1	5	call	sub_4014B0	ConsoleLogin.exe
14	☐		401115	1	6	call	sub_401850	ConsoleLogin.exe
17	☑	2	401138	1	4	call	sub_4012F0	ConsoleLogin.exe
18	☑	2	401142	1		call	??6?$basic_ostream@DU?$char_traits...	MSVCP80
23	☑	2	401182	1		call	??1?$basic_string@DU?$char_traits@...	MSVCP80
24	☑	2	401194	1		call	??1?$basic_string@DU?$char_traits@...	MSVCP80
25	☑	2	4011b1	1	7	call	@__security_check_cookie@4	ConsoleLogin.exe

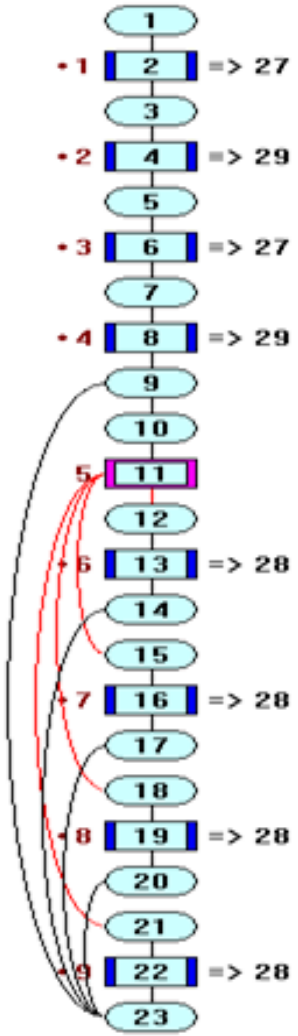
По умолчанию контрольные точки автоматически сохраняются в БД ИК «IRIDA» 2.0. При необходимости их можно экспортировать в один из двух форматов:

- формат .ep, с которым работает ПК «ExeTracer» 2.0;
- формат, принимаемый программой GDB;

Паспорт процесса

Описания управляющего графа в виде ANTLR-грамматики будет следующим:

```
class SmallStructPassportParser extends Parser;
 m0 : m1;          //sub_401000
 m1:               //sub_401000
    KT1   m27  KT1
    KT2   m29  KT2
    KT3   m27  KT3
    KT4   m29  KT4
    (  r0 | ( KT6   m28  KT6  | KT7  m28  KT7   |  KT8   m28  KT8  |  KT9   m28
KT9 ) ) ;
 m27 : ;           //sub_401AA0
 m28 : ;           //sub_401AC0
 m29 : ;           //sub_401AE0
 r0 : ;
 class SmallStructPassportLexer extends Lexer;
KT1    : "000010001"; KT2    : "000020001";
KT3    : "000030001"; KT4    : "000040001";
KT6    : "000060001"; KT7    : "000070001";
KT8    : "000080001"; KT9    : "000090001";
```



Данное описание можно рассматривать как **лингвистический паспорт** $\mathcal{P}_{ri}^{L}$ **ПП**$_{ri}$. В паспорте задекларированы допустимые пути выполнения ПП в терминах КТ. По данному паспорту ПП и создается программа **автомата динамического контроля** потоков управления в ней.

Третий этап контроля и предотвращения недекларированного выполнения ПК

Цель третьего этапа - синтез и встраивание средств динамического контроля, предотвращающих недекларированное выполнение процессов ПК, что достигается решением следующих задач:

- Синтез автомата динамического контроля (АДК) выполнения процесса $П_r$ по его паспорту $\mathcal{P}_r$.
- Встраивание средств предотвращения недекларированного выполнения программ процессов $\{П_r\}$ в ПК.

Программный комплекс «***ExeTracer» 2.0*** предназначен для подготовки исполняемых файлов и библиотек динамической компоновки формата PE к динамическому анализу по контрольным точкам (для ближних вызовов и вызовов через таблицу импорта).

- встраивание функции формирования трассы в исполняемый код исследуемой программы;
- создание лабораторной сборки исследуемого модуля путем встраивания в исполняемый модуль функции формирования трассы прохождения выбранного статического маршрута в исследуемой программе;
- автоматическая генерация программы для анализа соответствия статических и динамических маршрутов выполнения программы.

Результатом выполнения третьего этапа являются:

- Новый исполняемый файл исследуемого программного комплекса с установленными контрольными точками *X_new.exe (лабораторная сборка);*
- Модуль для вывода трассы и проведения динамического контроля потоков управления *X_TrcSheet.dll;*
- Утилита *DiagenAnalizer_X.exe* для отсроченного прохождения трассы контрольных точек.

Для проведения трассировки исследуеммого программного комплекса с одновременным динамическим контролем потоков управления **запускаем на выполнение вновь созданный файл *X_new.exe***.

В ходе его выполнения формируются следующие файлы:
Output.out *– трасса прохождения контрольных точек;*
dynoutput.out *– результаты динамического контроля* модуля динамического контроля *АДК (сравнение статических и динамических маршрутов).*

фрагмент файла Output.out имеет вид:

```
000100002
000100002
000110002
000110002
000120002
000120002
000140002
000140002
000150002
```

фрагмент файла Dynoutput.out имеет вид:

```
Next token : 000100002
Next token : 000100002
Next token : 000110002
Next token : 000110002
Next token : 000120002
Next token : 000120002
Next token : 000140002
ERROR in token 000140002
Next token : 000140002
ERROR in token 000140002
Next token : 000150002
ERROR in token 000150002
```

Результат выполнения третего этапа

Полученные результаты динамической трассировки в соответствии с установленными КТ могут быть наложены на статически установленные КТ в среде **ИК ИРИДА 2.0.**

Это позволяет проводить дополнительное интерактивное исследование хода выполнения процесса, выявить невызываемые ПП, получить статистику вызовов ПП и т.д.

Таким образом, предлагаемые метод защиты, инструментарий и АДК могут использоваться для защиты от проявления и использования скрытых дефектов в программах, а также защиты программного кода от атак.

При этом средством защиты выступает АДК, а Паспорт $\mathcal{P}_r^{\mathrm{L}}$, в общем случае, является описанием множества доверенных путей выполнения процесса $П_r$.

Если в паспорте представлены только доверенные пути, то любое отклонение можно будет зафиксировать, проставив дополнительные КТ на недоверенные компоненты, динамически связываемые ПП, не принадлежащие контролируемому процессу. **Этот метод можно называть методом паспортов**.

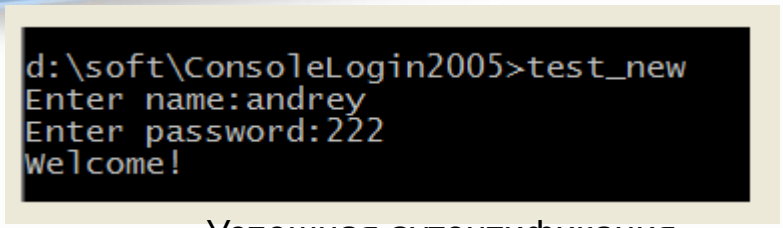
Пример применения метода паспортов защиты ПО

Рассмотрим технологию контроля и предотвращения недекларированного выполнения программ на примере консольного приложения, моделирующего механизм аутентификации, в который преднамеренно вставлена закладка. Закладка реализована в рамках функции **IsSuperUser** и позволяет пользователю **admin** получить доступ в систему при вводе пароля «111».

```
#include <string>
#include <iostream>
#include <fstream>
#include <sstream>
using namespace std;
map<string, string> passwords;
bool IsSuperUser( const string & name, const string & password )
{
        if( name == "admin" && password == "111" ) return true;
        return false;
}
bool IsAuthorizedUser( const string & name, const string & password )
{
        if( passwords.find( name ) != passwords.end() && passwords[name] == password )
                return true;
        return false;
}
void ReadPasswords()
{
        string passwd_file = "password.txt";
        ifstream passwd_stream( passwd_file );
        if( ! passwd_stream.is_open() )
        {
                string err = strerror( errno );
                cerr << "Cannot open file " << passwd_file
                << " :  " << errno << " : " << err << endl;
                exit( -1 );
        }
        string line;
```

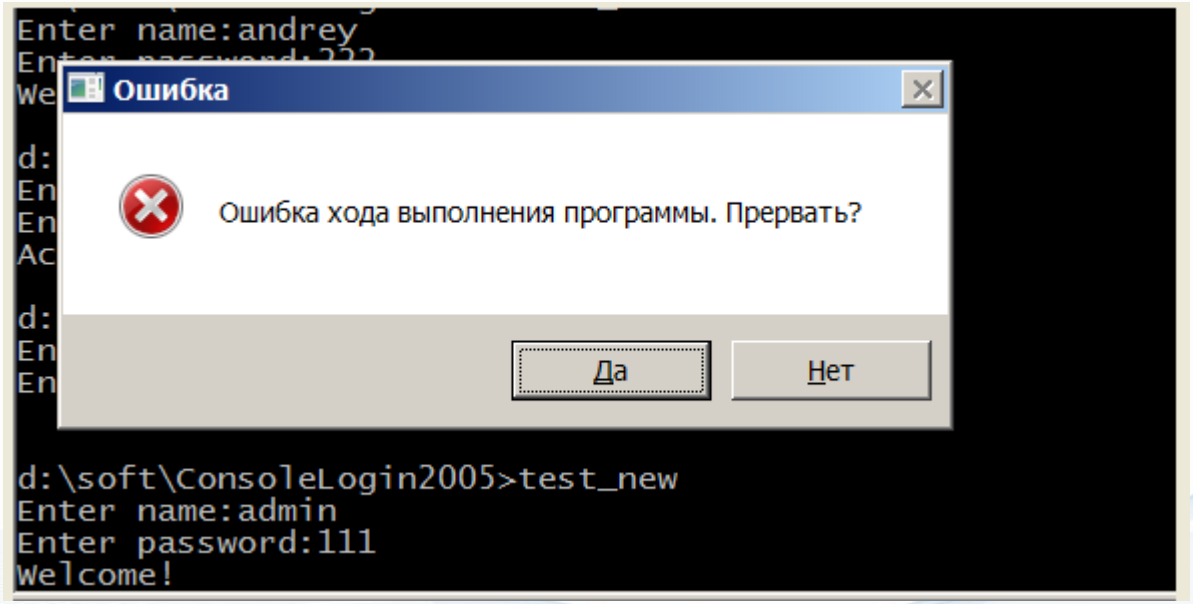
Пример применения метода паспортов защиты ПО

Информация о легальных пользователях и их паролях хранится в базе данных (файле), в данном случае легальный пользователь andrey, пароль «222»



Успешная аутентификация

После применения технологии контроля и предотвращения недекларированного выполнения программ при активизации закладки (пользователь **admin** и пароль «**111**») совместно с исходной программой будет функционировать автомат динамического контроля, который и предотвратит несанкционированный доступ в систему.



Сообщение об ошибке аутентификации

Описанный метод защиты реализован в Инструментальном комплексе для проведения статического и динамического анализа потоков управления в исполняемых кодах программ «IRIDA» 2.0

Компонентный состав ИК «IRIDA» 2.0

ГАЗИНФОРМСЕРВИС

приложение ***IRIDA_Framework_2.0*** предназначено для статического анализа потоков управления в исследуемой программе, подготовки маршрутов для динамического анализа и создания средств их анализа

Плагин ***IRIDA_IDA_plugin_2.0*** предназначен для создания базы данных комплекса, помещения в базу данных дизассемблированного, с использованием дизассемблера IDA Pro, исполняемого кода программы

Программный комплекс «ExeTracer» 2.0 предназначен для подготовки исполняемых файлов и библиотек динамической компоновки формата PE к динамическому анализу по контрольным точкам (для ближних вызовов и вызовов через таблицу импорта).

ИК «IRIDA» 2.0

Особенности ИК «IRIDA» 2.0

ГАЗИНФОРМСЕРВИС

Отличительные особенности ИК «IRIDA» 2.0:

1) расширена номенклатура поддерживаемых платформ - Intel x86/x64/ PowerPC/ARM/MIPS;

2) при помощи плагина IRIDA_IDA_plugin_2.0 автоматизирована операция создания проекта ИК «IRIDA» 2.0.

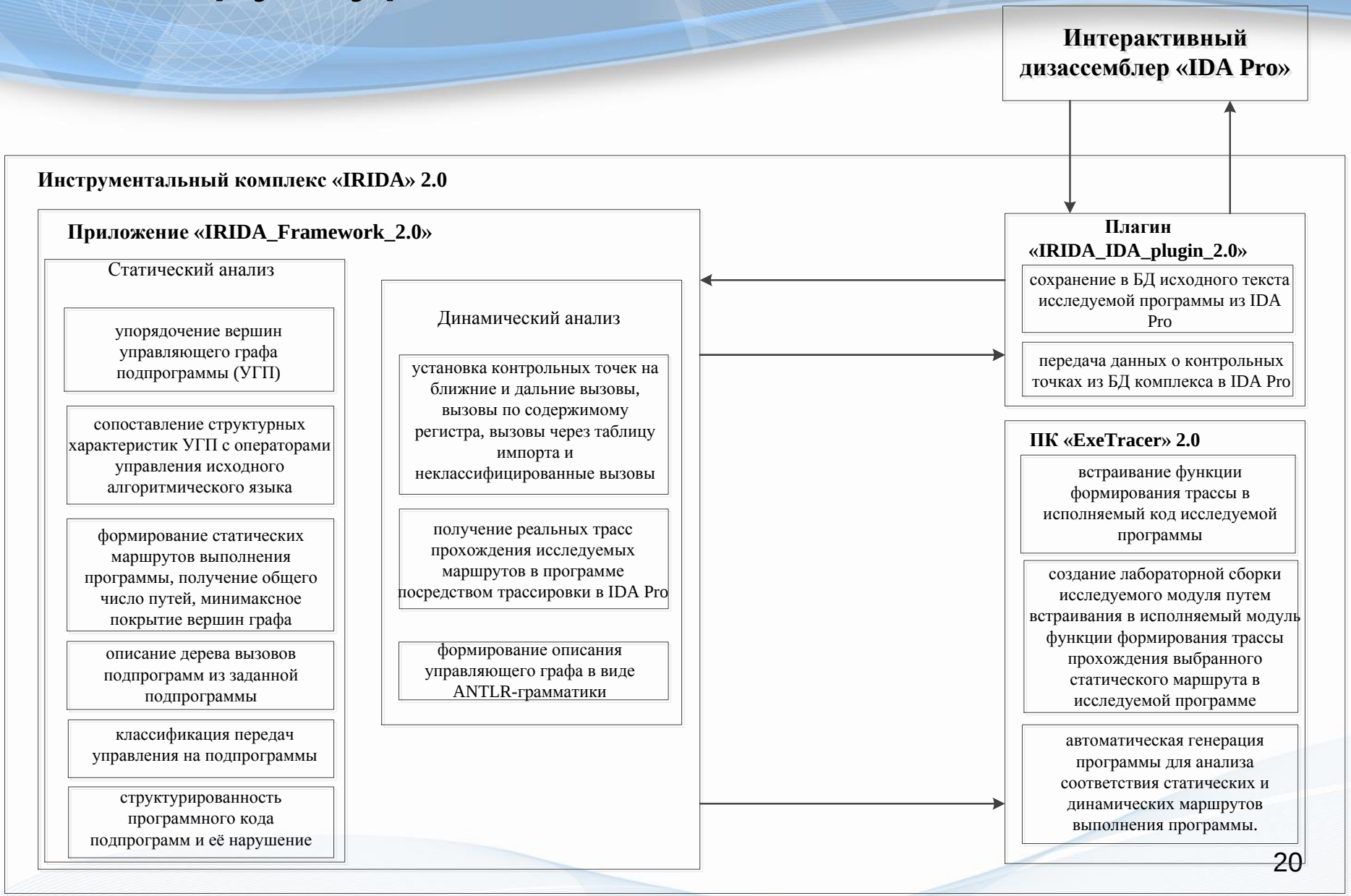
3) функционирование под управлением операционной системы Windows 7.

4) локальная и удаленная трассировка с использованием отладчиков, поддерживаемых IDA Pro.

5) усовершенствованы алгоритмы формирования грамматики для поддержки управляющих графов подпрограмм любой сложности.

6) добавлена возможность экспортирования отчетов по покрытию контрольных точек и подпрограмм в формат CSV.

Структурная схема ИК «IRIDA» 2.0

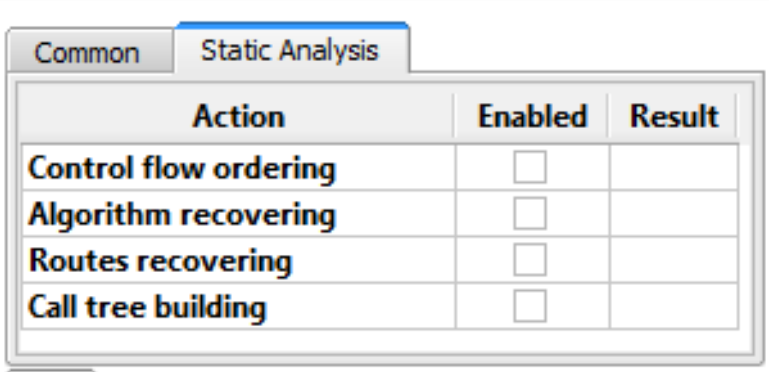


Решаемые задачи статического анализа

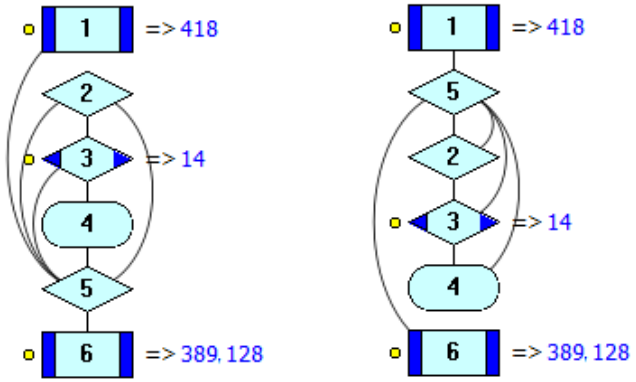
ГАЗИНФОРМСЕРВИС

На этапе статического анализа ИК «IRIDA» 2.0 автоматизирует процесс получения следующих характеристик исполняемых кодов программ на платформе Intel x86/x64, PowerPC, ARM, MIPS:

Для каждой подпрограммы с помощью контекстного меню или кнопок на панели инструментов могут быть активизированы следующие задачи:
Control flow ordering – упорядочение потоков управления;
Algorithm recovering – определение циклов и конструкции switch и отображение их на графе;
Routes recovering – получение списка путей графа;
Call tree building – построение графа вызовов текущей подпрограммы.



1. Изменение исходной упорядоченности вершин УГП



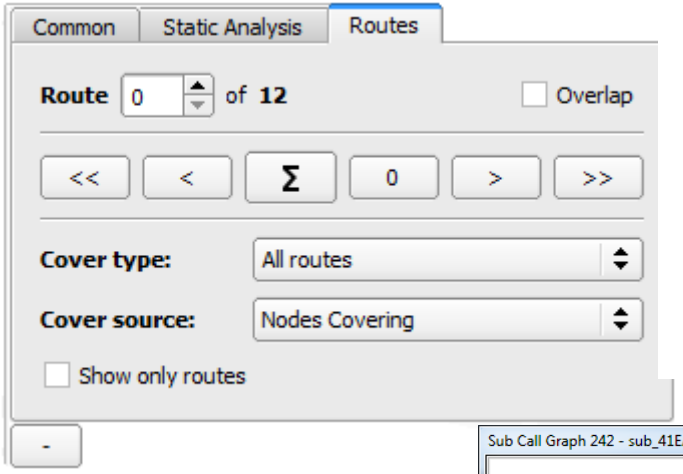
Управляющий граф подпрограммы до упорядочения (слева) и после (справа)

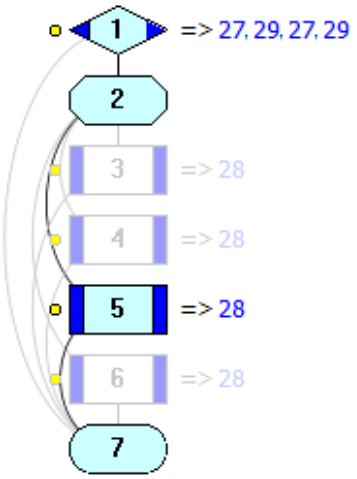
Решаемые задачи статического анализа

ГАЗИНФОРМСЕРВИС

2. Общее число путей в подпрограммах, минимаксное покрытие вершин

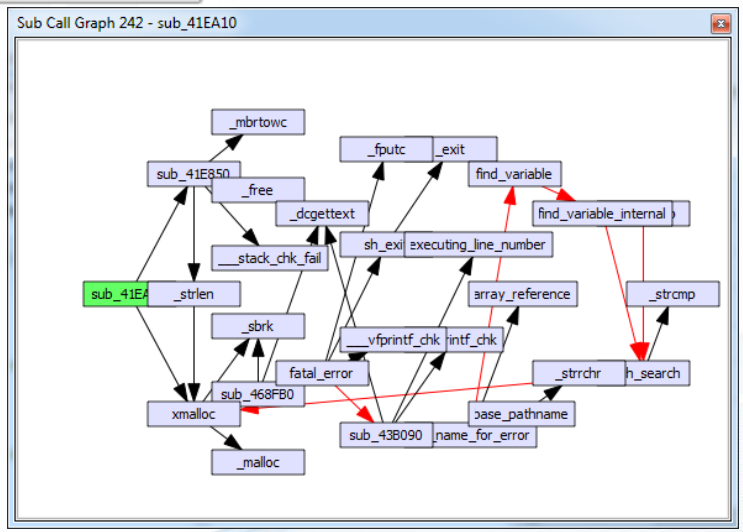
Инструмент ***«Routes recovering»*** позволяет получить общее количество путей в подпрограмме и построить минимаксное покрытие вершин (описание и представление минимального числа путей, покрывающих все вершины (линейные участки) управляющего графа подпрограммы).





3. Описание графа вызовов подпрограмм из заданной подпрограммы.

Инструмент ***«Call tree building»*** позволяет получить граф вызова текущей подпрограммы



Результат работы инструмента ***«Call tree building»***

Решаемые задачи статического анализа

ГАЗИНФОРМСЕРВИС

4. Классификация передач управления на подпрограммы

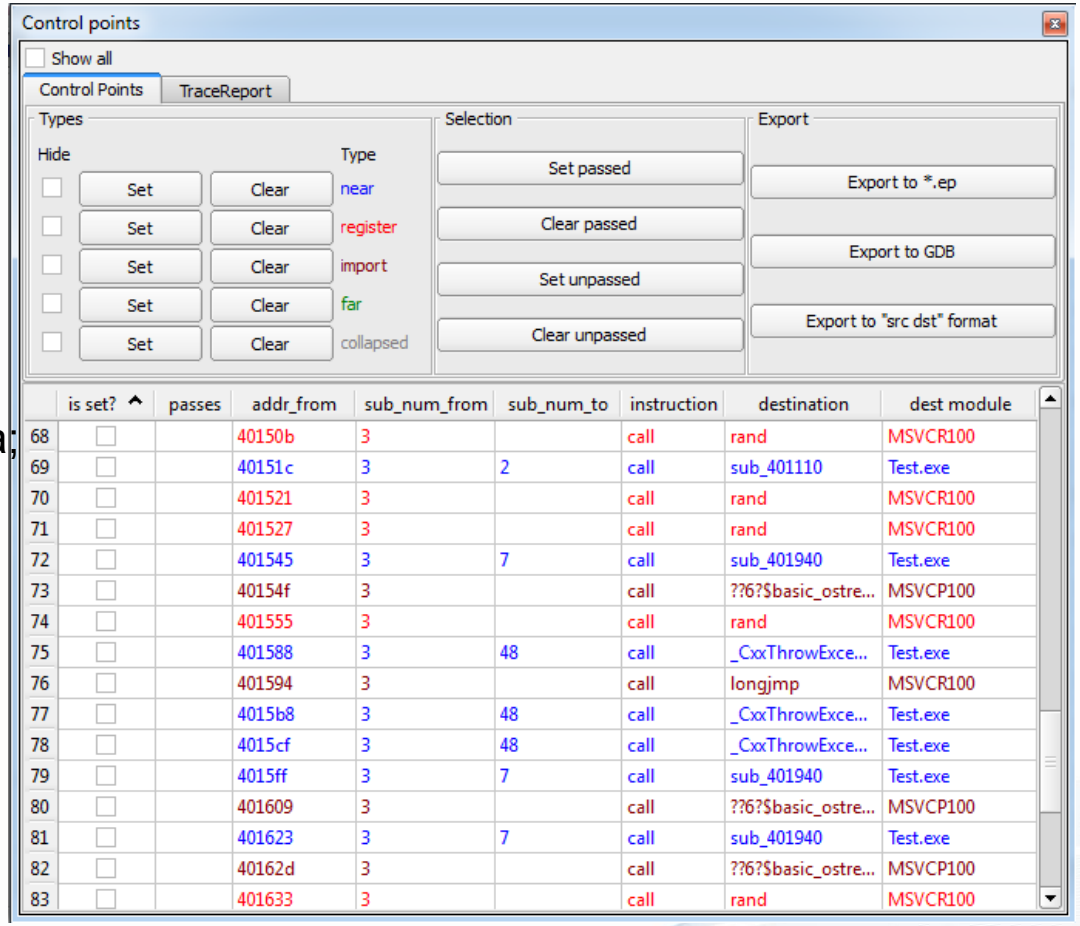
Ближние вызовы;

Дальние вызовы;

Вызовы по содержимому регистра;

Вызов через таблицу импорта;

Неклассифицированный вызов.

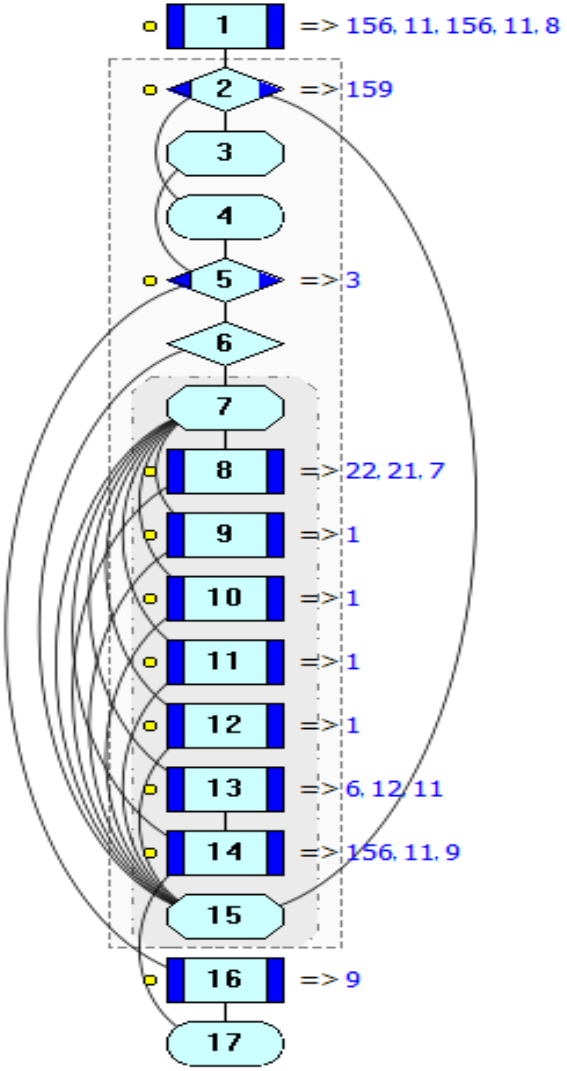


	is set?	passes	addr_from	sub_num_from	sub_num_to	instruction	destination	dest module
68			40150b	3		call	rand	MSVCR100
69			40151c	3	2	call	sub_401110	Test.exe
70			401521	3		call	rand	MSVCR100
71			401527	3		call	rand	MSVCR100
72			401545	3	7	call	sub_401940	Test.exe
73			40154f	3		call	??6?$basic_ostre...	MSVCP100
74			401555	3		call	rand	MSVCR100
75			401588	3	48	call	_CxxThrowExce...	Test.exe
76			401594	3		call	longjmp	MSVCR100
77			4015b8	3	48	call	_CxxThrowExce...	Test.exe
78			4015cf	3	48	call	_CxxThrowExce...	Test.exe
79			4015ff	3	7	call	sub_401940	Test.exe
80			401609	3		call	??6?$basic_ostre...	MSVCP100
81			401623	3	7	call	sub_401940	Test.exe
82			40162d	3		call	??6?$basic_ostre...	MSVCP100
83			401633	3		call	rand	MSVCR100

Решаемые задачи статического анализа

ГАЗИНФОРМСЕРВИС

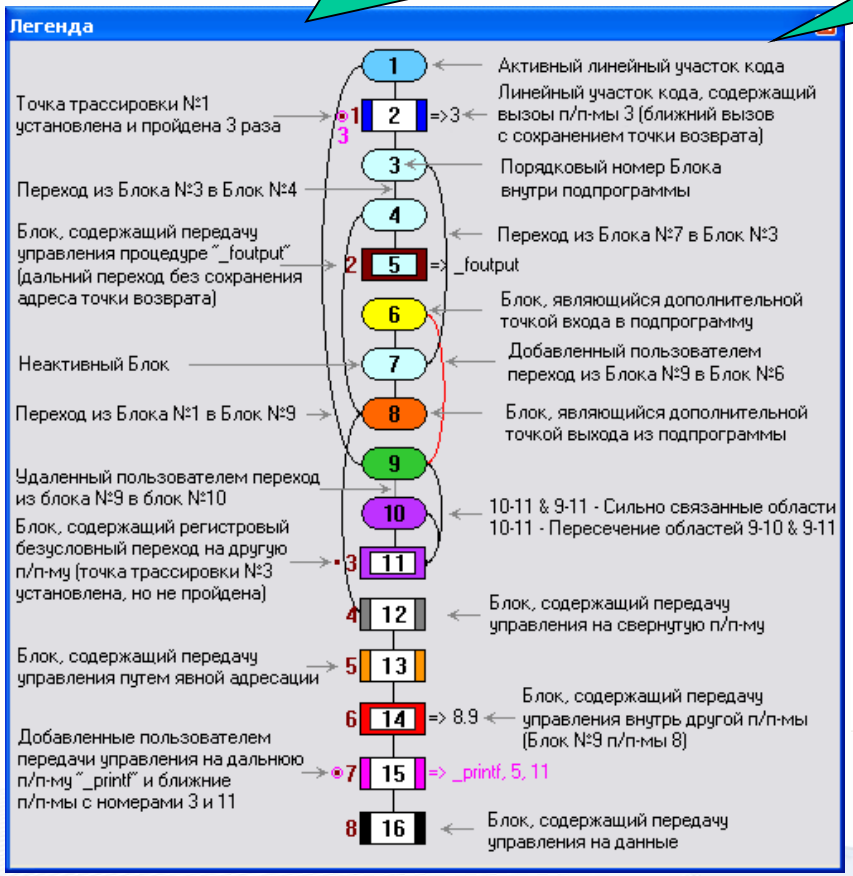
5. Характеристики структурированности программного кода подпрограммы и их нарушения (циклические участки кода, дополнительные точки входа и выхода из подпрограмм и циклов, зацепления циклических участков, операторы switch).

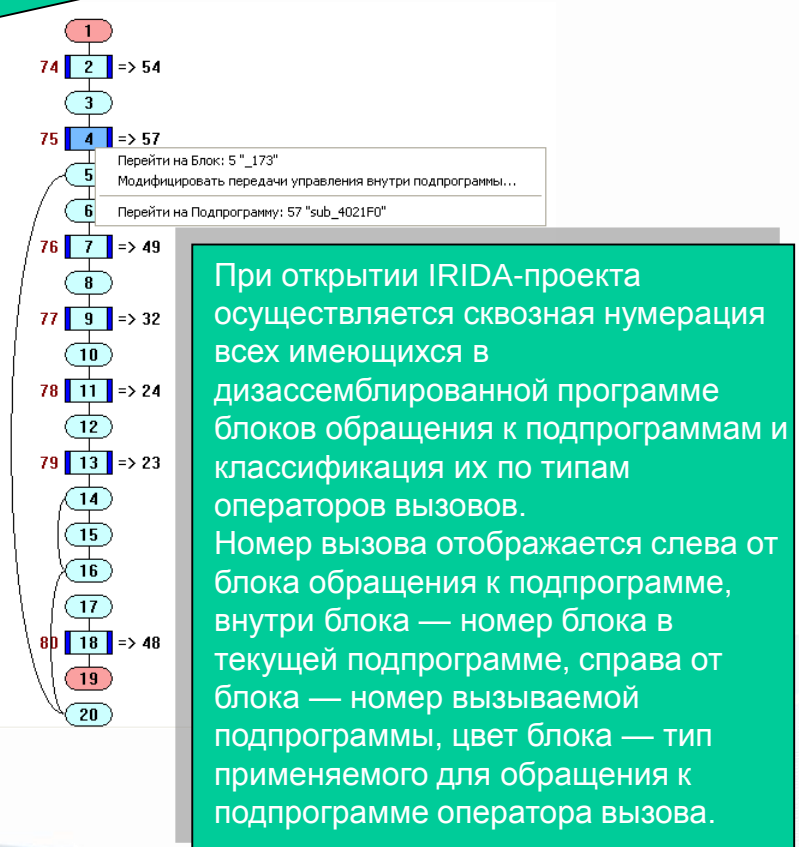


Обозначение блоков и их раскраска на управляющем графе

В рамках IRIDA-проекта применяются различные обозначения блоков и их раскраска на управляющем графе подпрограммы.

Диалога с подсказкой о структуре и обозначениях блоков на графе подпрограммы выводится при активизации кнопки  на панели инструментов.





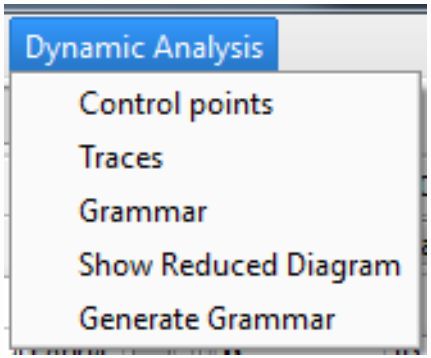
При открытии IRIDA-проекта осуществляется сквозная нумерация всех имеющихся в дизассемблированной программе блоков обращения к подпрограммам и классификация их по типам операторов вызовов.
Номер вызова отображается слева от блока обращения к подпрограмме, внутри блока — номер блока в текущей подпрограмме, справа от блока — номер вызываемой подпрограммы, цвет блока — тип применяемого для обращения к подпрограмме оператора вызова.

Решаемые задачи динамического анализа

ГАЗИНФОРМСЕРВИС

На этапе динамического анализа ИК «IRIDA» 2.0 автоматизирует следующие технологические операции:

- ***для исполняемых кодов программ на платформе Intel x86/x64, PowerPC, ARM, MIPS:***
  - формирование статических маршрутов выполнения программы и их фиксацию путем установки контрольных точек;
  - получение реальных трасс прохождения исследуемых маршрутов в программе посредством трассировки в IDA Pro.



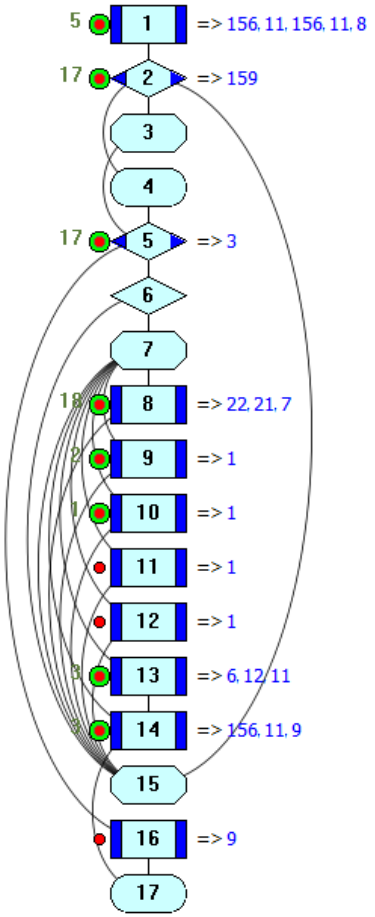
С помощью приложения ***IRIDA_Framework_2.0*** реализованы следующие инструменты динамического анализа:
Control points – расстановка контрольных точек;
Traces – получение трассы;
Grammar – построение грамматики для нескольких подпрограмм;
Show Reduced Diagram – построение графа грамматики;
Generate Grammar – формирование файла грамматики (паспорта программы).

Решаемые задачи динамического анализа

ГАЗИНФОРМСЕРВИС

- ***для исполняемых кодов программ на платформе Intel x86:***

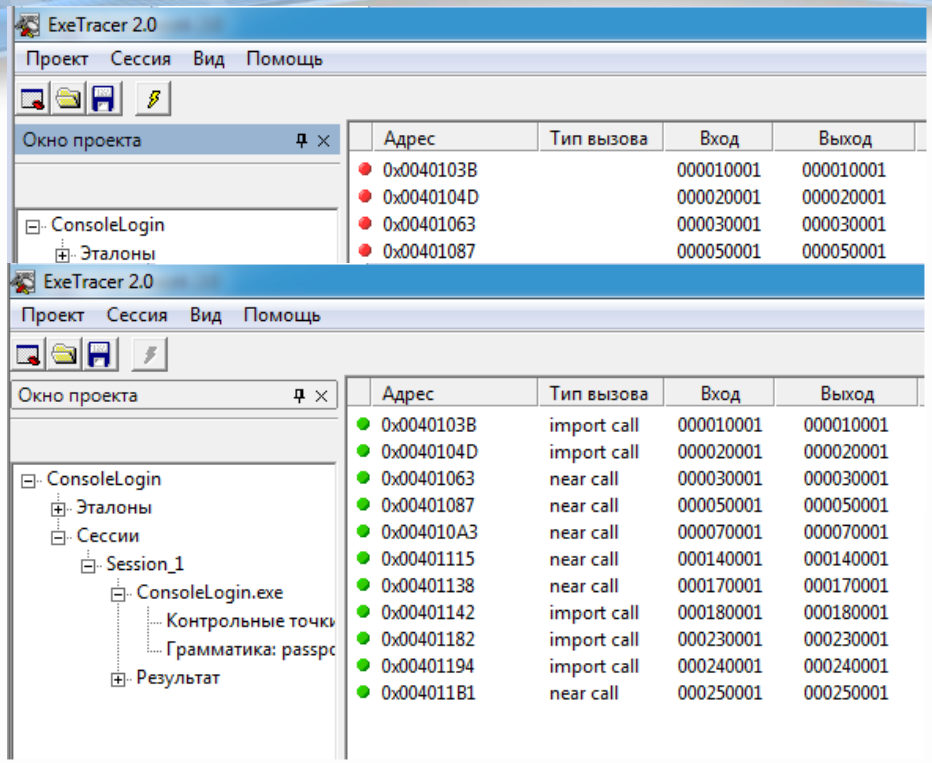
  - формирование описания управляющего графа в виде ANTLR-грамматики;
  - встраивание в исполняемый модуль процедур, реализующих формирование записей контрольных точек;
  - создание лабораторной сборки исследуемого модуля с включенными в него процедурами формирования записей контрольных точек;
  - автоматическая генерация программы для анализа соответствия статических и динамических маршрутов выполнения программы.



Решаемые задачи динамического анализа

С помощью программного комплекса ***«ExeTracer» 2.0*** реализованы следующие задачи динамического анализа:

- формирование нового исполняемого файла исследуемой программы с установленными контрольными точками;
- формирование модуля для вывода трассы и проведения динамического контроля потоков управления;
- формирование утилиты ***DiagenAnalizer***, которая является консольным приложением, предназначенным для отложенного анализа трассы прохождения контрольных точек на соответствие эталонной трассе.



Вид информационной панели установленных контрольных точек после успешной обработки

Результатами обработки проекта являются:

- новый исполняемый файл исследуемой программы с установленными контрольными точками ***ConsoleLogin_new.exe*;**
- модуль для вывода трассы и проведения динамического контроля потоков управления ***ConsoleLogin_TrcSheet.dll;***
- утилита ***DiagenAnalizer_ConsoleLogin.exe*** для отложенного анализа трассы прохождения контрольных точек.

Для проведения трассировки программы с одновременным динамическим контролем потоков управления необходимо запустить на выполнение файл ***ConsoleLogin_new.exe.***

Утилита DiagenAnalizer

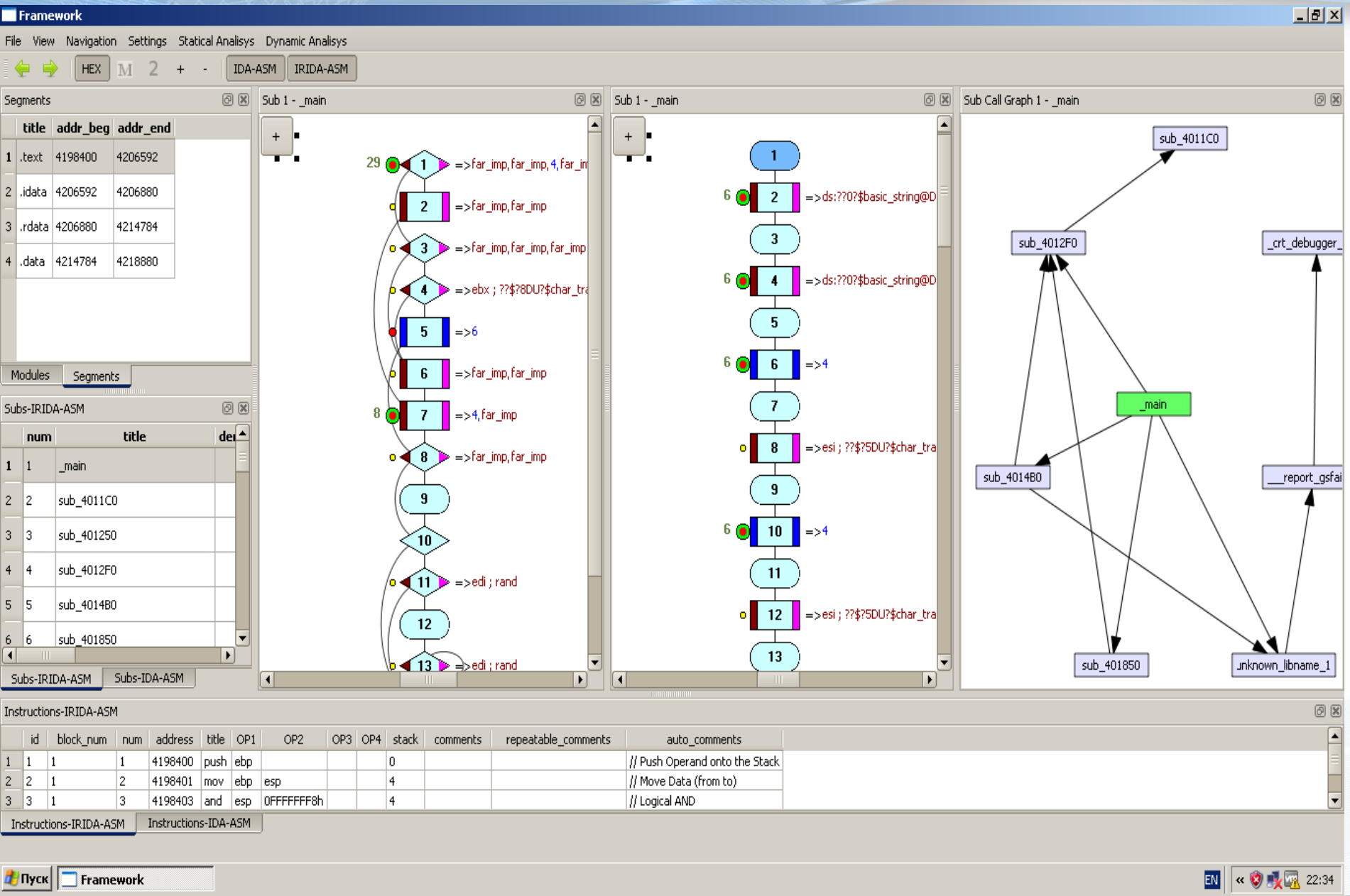
Утилита ***DiagenAnalizer*** является консольным приложением, предназначенным для отсроченного анализа трассы прохождения контрольных точек на соответствие эталонной трассе.

Утилита конфигурируется с помощью параметров командной строки.

DiagenAnalizer.exe	[options]		<input filename>
	-oS	Выводить результаты на консоль	Имя входного файла с трассой контрольных точек.
	-oF <output filename>	Dsdjlbnm htpekmnfns d afqk output filename	
	-eC	Не прерывать анализ при ошибке. При возникновении ошибки выводится диалоговое окно, в котором можно выбрать дальнейшие действия: Прервать-Повторить-Пропустить. Прервать: Завершить выполнение программы; Повторить: Исключить ошибочную КТ из трассы и повторит анализ снова. Пропустить: Пропускать все ошибки и не показывать окно.	
	-eH	Прервать анализ при ошибке	

Результаты работы утилиты будут выдаваться на консоль и в случае возникновения несоответствия анализируемой трассы из файла **Output.out** и модели потоков управления будет выдано сообщение об ошибке.

Инструменты IRIDA 2.0



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

Спасибо за внимание

Компаниец Радион Иванович
kompaniec-r@gaz-is.ru

Ковалев Виктор Васильевич
kovalev-v@gaz-is.ru

ООО «Газинформсервис»
г. Санкт-Петербург
www.gaz-is.ru