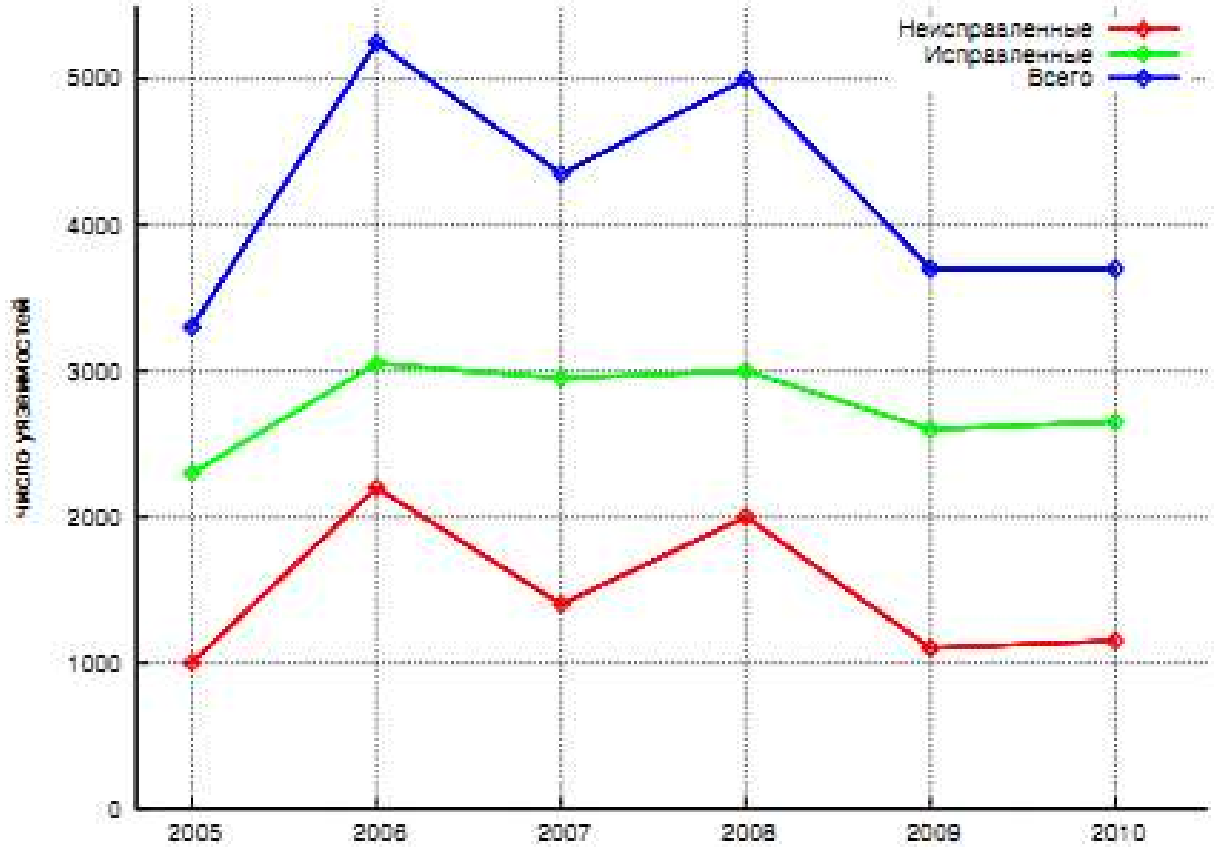
К вопросу о выявлении дефектов безопасности методом статического сигнатурного анализа

Алексей Марков, к.т.н., с.н.с., CISSP, SBCI
Андрей Фадин, CISSP

НПО «Эшелон»
mail@cnpo.ru

ТиПВСИТ - 2012

Проблема безопасности программ



Источник: ЗАО «НПО «Эшелон»

Общая динамика уязвимостей по данным публичных баз данных

Теория `и` практика = нечеткость

- Учебники по *статическому анализу* направлены на выявление «нефункциональных ошибок»
- Нормативные документы определяют требования по выявлению недекларированных возможностей (ПЗ) методами *статического анализа*
- Практика сертификационных испытаний, тематических исследований, аудита *методами статического анализа*

VS

№	Наименование требования	Уровень контроля			
		4	3	2	1
	Требования к документации				
1	**Контроль состава и содержания документации**				
1.1.	Спецификация (ГОСТ 19.202-78)	+	=	=	=
1.2.	Описание программы (ГОСТ 19.402-78)	+	=	=	=
1.3.	Описание применения (ГОСТ 19.502-78)	+	=	=	=
1.4.	Пояснительная записка (ГОСТ 19.404-79)	-	+	=	=
1.5.	Тексты программ, входящих в состав ПО (ГОСТ	+	=	=	=
	Требования к содержани				
2.	**Контроль исходного состояния ПО**	+	=	=	=
3.	**Статический анализ исходных текстов про**				
3.1.	Контроль полноты и отсутствия избыточност	+	+	+	=
3.2.	Контроль соответствия исходных текстов ПО	+	=	=	+
3.3.	Контроль связей функциональных объектов	-	+	=	=
3.4.	Контроль связей функциональных объектов	-	+	=	=
3.5.	Контроль информационных объектов	-	+	=	=
3.6.	Контроль наличия заданных конструкций в исходных текстах	-	-	+	+
3.7.	Формирование перечня маршрутов выполнения функциональных объектов	-	+	+	=
3.8.	Анализ критических маршрутов выполнения функциональных объектов	-	-	+	=
3.9.	Анализ алгоритма работы функциональных объектов на основе блок-схем, диаграмм и т. п., построенных по исходным текстам контролируемого ПО	-	-	+	=
4.	**Динамический анализ исходных текстов программ**				
4.1.	Контроль выполнения функциональных объектов	-	+	+	=
4.2.	Сопоставление фактических маршрутов выполнения функциональных объектов и маршрутов, построенных в процессе проведения статического анализа	-	+	+	=
5.	**Отчетность**	+	+	+	+

РД. Защита от НСД к информации. ПО СЗИ. Классификация по уровню контроля отсутствия недекларированных возможностей (Гостехкомиссия России, 1999)

Выполняемые проверки	Уровень контроля			
	4	3	2	1
1.Контроль состава и содержания документации	+	+	+	+
2.Контроль исходного состояния	+	=	=	=
3.1.Контроль отсутствия избыточности исходных текстов	+	+	+	=
3.2.Контроль соответствия исходных текстов загрузочному коду	+	=	=	+
3.3.Контроль связей функциональных объектов по управлению	-	+	=	=
3.4.Контроль связей функциональных объектов по информации	-	+	=	=
3.5.Контроль информационных объектов	-	+	=	+
3.6.Контроль наличия заданных конструкций	-	-	+	+
3.7.Формирование перечня маршрутов выполнения ФО	-	+	+	=
3.8.Анализ критических маршрутов выполнения ФО	-	-	+	=
3.9.Анализ алгоритма работы на основе блок-схем, построенных по исходным текстам контролируемого ПО	-	-	+	=
4.1.Контроль выполнения функциональных объектов	-	+	+	=
4.2.Сопоставление фактических маршрутов и маршрутов, построенных в процессе проведения статического анализа	-	+	+	=

Статический анализ

Постановка задачи

- Терминология: дефекты и уязвимости
- Методы статического анализа безопасности кода
- Место и роль сигнатурного статического анализа кода
- Реализационные моменты

БЕЗОПАСНОСТЬ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Дефекты и уязвимости

Базовые определения безопасности ПО

- **Дефект** (weakness/bug, стандарт CWE)

- недостаток реализации ПО, потенциально влияющий на степень безопасности информации

- **Уязвимость** (vulnerability, база CVE)

- эксплуатируемый дефект, потенциально составляющий угрозу безопасности

- **Угроза** (threat) …
- **Риск** (risk) …

Отдельные термины (Россия)

- **Дефект** – каждое отдельное несоответствие продукции установленным требованиям (ГОСТ 27.002-89)
- **Недекларированные возможности** - функциональные возможности ПО, не описанные или не соответствующие описанным в документации, при использовании которых возможно нарушение конфиденциальности, доступности или целостности обрабатываемой информации (РД Гостехкомиссии)
- **Программная закладка -** преднамеренно внесенные в ПО функциональные объекты, которые при определенных условиях инициируют реализацию НДВ ПО (ГОСТ 50.1.053-05)
- **Скрытый канал** – непредусмотренный разработчиком системы ИТ и АС коммуникационный канал, который может быть применен для нарушения политики безопасности (ГОСТ 53113.1-08)
- **Уязвимость** ИС; **брешь** - свойство ИС, обусловливающее возможность реализации угроз безопасности обрабатываемой в ней информации (ГОСТ 50922-06)

Международные таксономии

- OWASP Top Ten
  - 10 самых больших классов уязвимостей для веб-приложений
- Fortify Seven Pernicious Kingdoms
  - 7 разрушительных "царств" компании HP Fortify
- MITRE Common Weaknesses Enumeration
  - всесторонняя классификация дефектов ПО

Общие классификационные признаки

- Место в ЖЦ (архитектурные, программирования, эксплуатации)
- Степень преднамеренности («функциональные», «нефункциональные», DOS, ПЗ, СК)
- Подсистема безопасности (I A A A ….)

Признаки локализации

- Модель

SD=<CF; DD>,

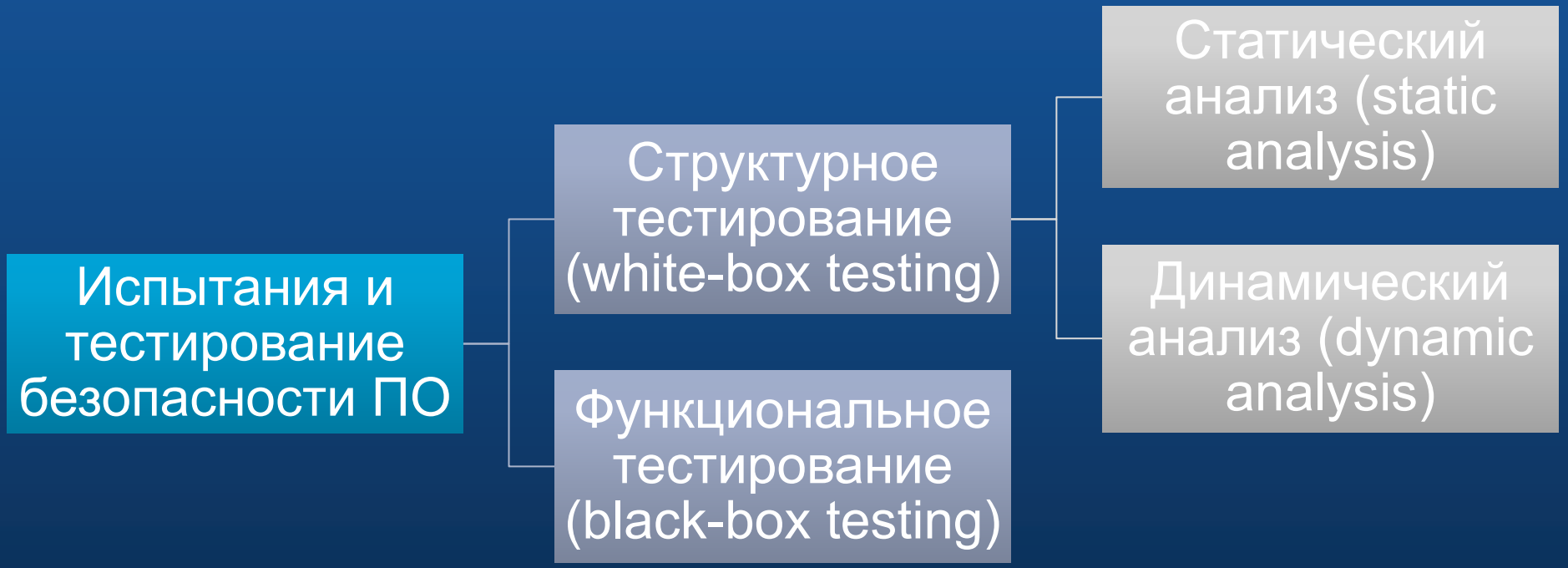
где CF – признаки проектирования и кодирования

(для каждой СПП), DD - область данных

МЕТОДЫ СТАТИЧЕСКОГО АНАЛИЗА БЕЗОПАСНОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Обзор

Подходы к анализу безопасности ПО



Статический анализ

- не требует запуска ПО, сложность статического анализа принципиально ниже сложности динамических методов при выявлении дефектов, связанных с редко используемыми вх.данными
- объекты исследования
  - исходные тексты ПО,
  - исполняемый код ПО,
  - материалы производства ПО (отладочная информация, логи сборки, вывод внешних средств),параметры программного проекта
- ручная инспекция, использование анализаторов кода
- ограниченный набор проверяемых свойств ПО (проблема «останова», теорема Райса)

Области применения статического анализа в широком смысле

- компиляция, интерпретация
- исследование архитектуры и алгоритмов программы, автодокументирование
- структурирование кода
- вычисление программных метрик, оценка сложности и качества
- оптимизация, распараллеливание, профилировка
- преобразования (трансформации) программ, метапрограммирование
- обфускация/деобфускация программ
- обнаружение дефектов

Основные направления статического анализа в области безопасности

Методы статанализа	Выявляемы дефекты
Декомпозиция программы	Выявления нарушений полноты и избыточности
Оценка метрик сложности	Выявление сложных и аномальных фрагментов
Проверка свойств (ограничений) программы на моделях программного кода	Выявление некорректностей программирования (нефункциональных ошибок)
Сигнатурный анализ	Выявление фрагментов и дефектов

Статический анализ корректности программирования по моделям представления программного кода (абстрактная интерпретация)

- Некорректности программирования (нефункциональные ошибки): избыточные ФО и ИО, переполнение буфера, утечки памяти, ошибки типов данных, ошибки указателей и др.
- Алгоритмы выявления: интервальный анализ, анализ указателей, анализ зависимостей по данным и др.
- Преимущество: простота автоматизации
- Недостаток: ограниченность выявления дефектов преднамеренного типа

Сигнатурный статический анализ

- **Сигнатурный анализ программного кода**

  - поиск программных дефектов в исходных текстах или исполняемых модулях путем сопоставления фрагментов с образцами из базы данных шаблонов (сигнатур) дефектов безопасности.

МОДЕЛИ ПРЕДСТАВЛЕНИЯ ПРОГРАММНОГО КОДА

Краткий обзор

Этапы построения модели программы

1) Исходный текст программы

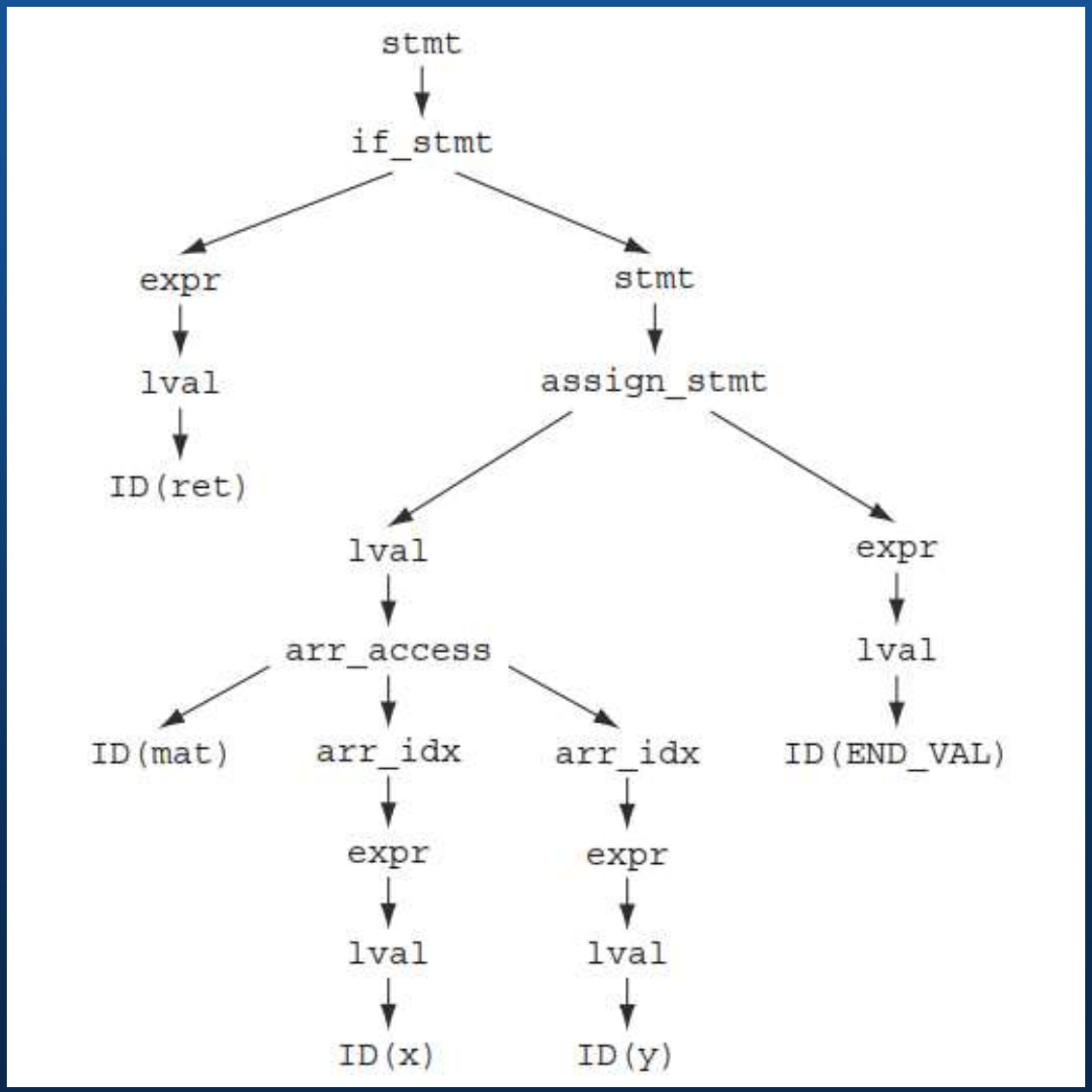
```
if (ret) // probably true
  mat[x][y] = END_VAL;
```

2) Лексический анализ

```
IF LPAREN ID(ret) RPAREN ID(mat) LBRACKET ID(x) RBRACKET LBRACKET
ID(y) RBRACKET EQUAL ID(END_VAL) SEMI
```

Этапы построения модели программы

3) Синтаксический анализ. Дерево разбора



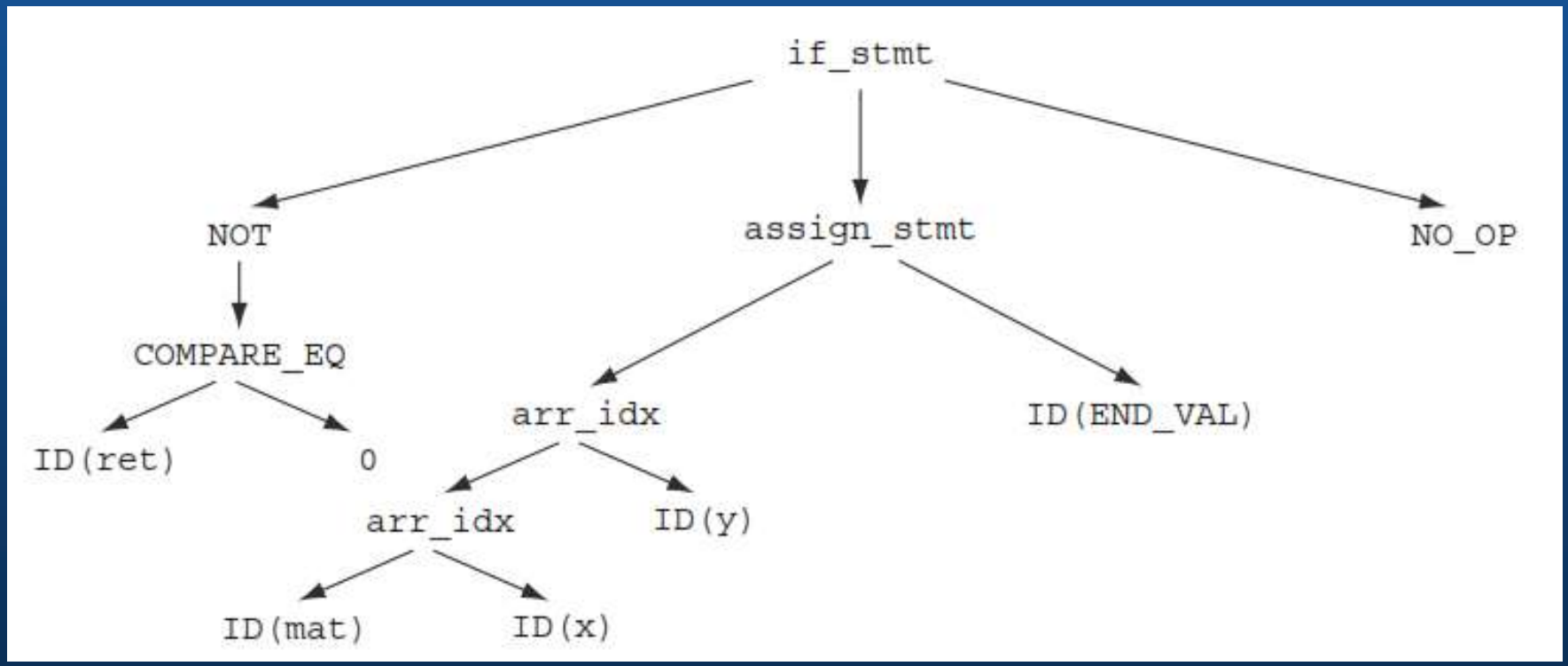
Соответствие РД Гостехкомиссия России

http://www.fstec.ru/_docs/doc_3_3_010.htm

Выполняемые проверки	Уровень контроля			
	4	3	2	1
1.Контроль состава и содержания документации	+	+	+	+
2.Контроль исходного состояния	+	=	=	=
3.1.Контроль отсутствия избыточности исходных текстов	+	+	+	=
3.2.Контроль соответствия исходных текстов загрузочному коду	+	=	=	+
3.3.Контроль связей функциональных объектов по управлению	-	+	=	=
3.4.Контроль связей функциональных объектов по информации	-	+	=	=
3.5.Контроль информационных объектов	-	+	=	+
3.6.Контроль наличия заданных конструкций	-	-	+	+
3.7.Формирование перечня маршрутов выполнения ФО	-	+	+	=
3.8.Анализ критических маршрутов выполнения ФО	-	-	+	=
3.9.Анализ алгоритма работы на основе блок-схем, построенных по исходным текстам контролируемого ПО	-	-	+	=
4.1.Контроль выполнения функциональных объектов	-	+	+	=
4.2.Сопоставление фактических маршрутов и маршрутов, построенных в процессе проведения статического анализа	-	+	+	=

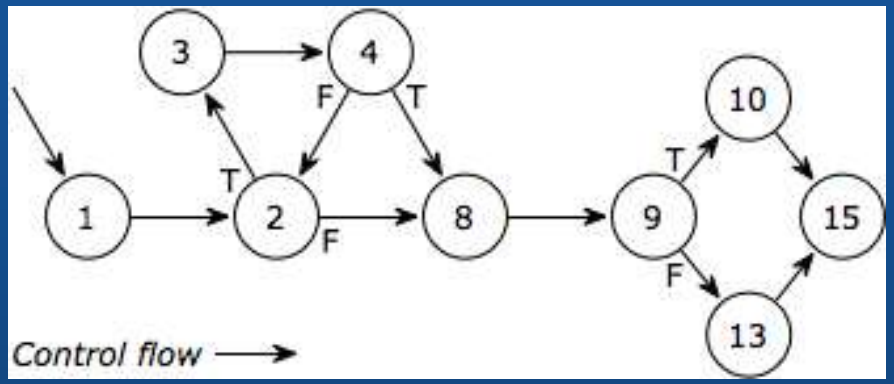
Этапы построения модели программы

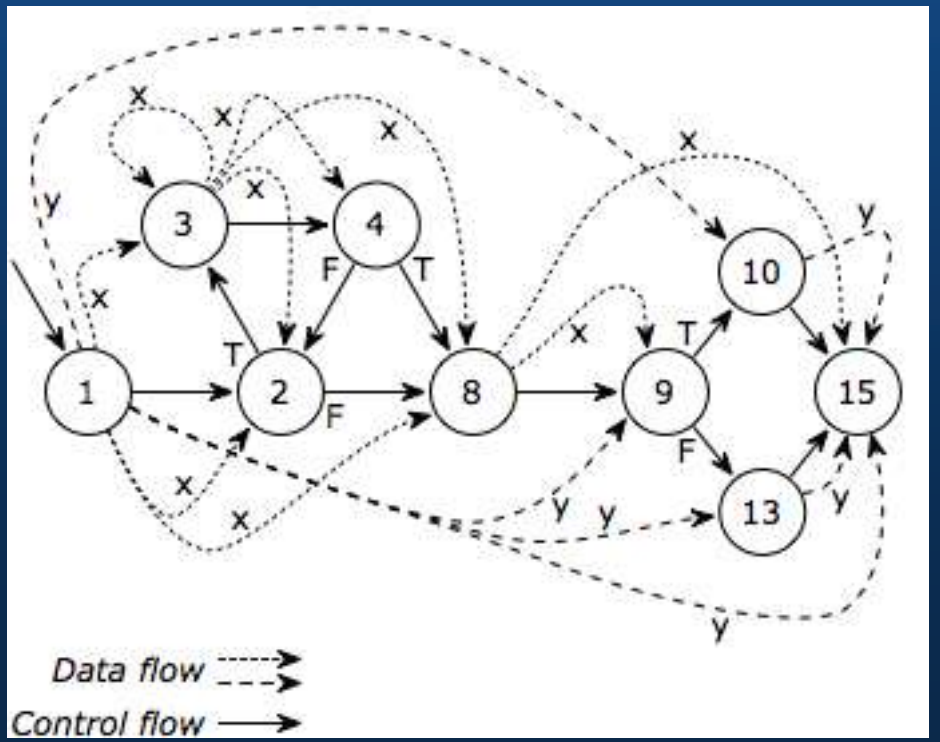
4) Абстрактное дерево разбора (AST)



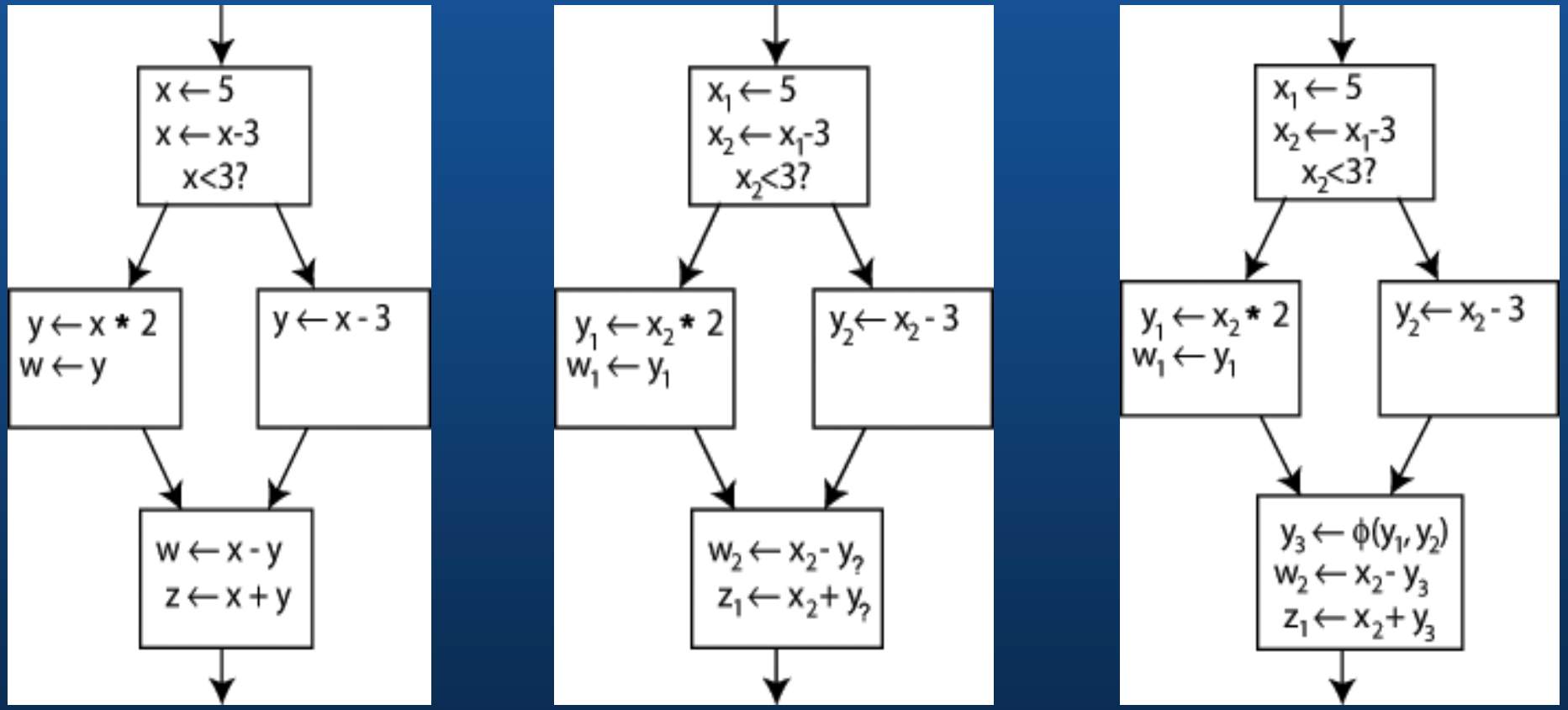
6) Граф потока управления (CFG) и граф зависимостей по данным (DFG)

```
1 int m(int x, int y) {
2 while (x > 10) {
3         x -= 10; // x = x - 10;
4         if (x == 10) {
5         break;
6         }
7 }
8 x = square(x);
9 if (y < 20 && x%2 == 0) {
10        y += 20; // y = y + 20;
11 }
12 else {
13        y -= 20; // y = y - 20; 14 }
15        return 2*x + y;
16 }
```

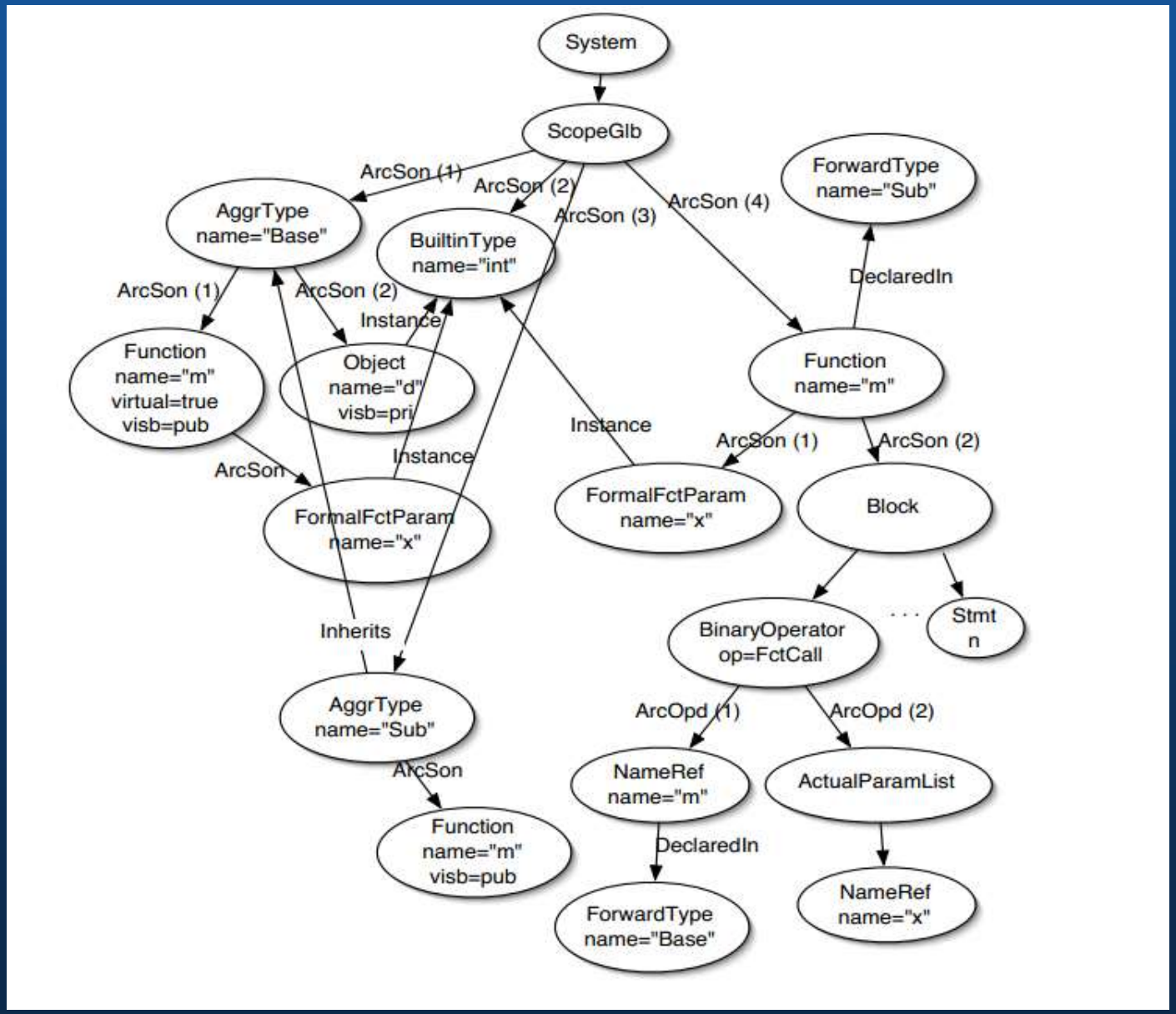




7) Однократное статическое присваивание (SSA)



8) Абстрактный семантический граф



Соответствие РД Гостехкомиссия России

http://www.fstec.ru/_docs/doc_3_3_010.htm

Выполняемые проверки	Уровень контроля			
	4	3	2	1
1.Контроль состава и содержания документации	+	+	+	+
2.Контроль исходного состояния	+	=	=	=
3.1.Контроль отсутствия избыточности исходных текстов	+	+	+	=
3.2.Контроль соответствия исходных текстов загрузочному коду	+	=	=	+
3.3.Контроль связей функциональных объектов по управлению	-	+	=	=
3.4.Контроль связей функциональных объектов по информации	-	+	=	=
3.5.Контроль информационных объектов	-	+	=	+
3.6.Контроль наличия заданных конструкций	-	-	+	+
3.7.Формирование перечня маршрутов выполнения ФО	-	+	+	=
3.8.Анализ критических маршрутов выполнения ФО	-	-	+	=
3.9.Анализ алгоритма работы на основе блок-схем, построенных по исходным текстам контролируемого ПО	-	-	+	=
4.1.Контроль выполнения функциональных объектов	-	+	+	=
4.2.Сопоставление фактических маршрутов и маршрутов, построенных в процессе проведения статического анализа	-	+	+	=

СООТНОШЕНИЕ МЕТОДОВ СТАТИЧЕСКОГО АНАЛИЗА

Обзор

Задачи, представления кода и методы статического анализа

Представление кода/ Решаемые задачи	Декомпозиция и анализ структуры программы	Проверка стиля, подсчет метрик	Проверка свойств программы	Поиск багов/дефектов по шаблону
Исходные тексты программного обеспечения	Контроль зависимостей на уровне компонентов и отдельных файлов проекта	Подсчет базовых метрик (LOC и т.п.)	Свойства по недоступны	**Поиск сигнатур по регулярным выражениям (реализации на основе grep, sed)**
Абстрактное синтаксическое дерево (AST)	Лексический и синтаксический анализ	Лексический и синтаксический анализ	Лексический и синтаксический анализ	Лексический и синтаксический анализ, **Поиск сигнатур по AST**
Абстрактный семантический граф (ASG): Поток управления (control flow)	Определение связей объектов в иерархии	Подсчет метрик связности	Анализ потока выполнения (control flow)	**Поиск сигнатур последовательностей инструкций**
ASG: Поток данных (data flow)	Определение зависимостей по данным	Подсчет метрик связности	Абстрактная интерпретация (abstract interpretation)**:** Интервальный анализ Анализ указателей Анализ зависимостей по данным	Анализ потока данных (data flow) **Поиск внутрипроцедурных сигнатур последовательностей инструкций с учетом передаваемых значений**

Задачи, представления и методы статического анализа

Представление кода/ Решаемые задачи	Декомпозиция и анализ структуры программы	Проверка стиля, подсчет метрик	Проверка свойств программы	Поиск багов/дефектов по шаблону
ASG: Межпроцедурный анализ (interprocedural analysis)	Не требуется	Подсчет метрик связности	Анализ на основе систем уравнений	**Поиск сигнатур с учетом связей между процедурами**
ASG: Семантические описания конструкций (например вызовов внешних API)	Не требуется	Подсчет метрик связности с учетом семантики	Абрстрактная интерпретация Ресурный анализ Темпоральная логика	**Поиск сигнатур на основе семантической базы конструкций**
ASG: Формальные методы верификации	Не требуется	Не требуется	Дедуктивная верификация	Не требуется

РЕАЛИЗАЦИОННЫЕ ОСНОВЫ

Пример

Требования к сигнатурному анализатору исходных текстов

- простота создания новых сигнатур и их анализаторов
- универсальность подхода, поддержка широкого спектра языков программирования
- минимальное число ложных срабатываний

- Пример фрагмента сигнатурного анализатора, способного определять дефекты безопасности, описанные на уровне внутрипроцедурного потока данных (data flow) с применением регулярных выражений

Использование сигнатурного подхода для анализа потоков данных

Представление кода/ Решаемые задачи	Декомпозиция и анализ структуры программы	Проверка стиля, подсчет метрик	Проверка свойств программы	Поиск багов/дефектов по шаблону
Исходные тексты программного обеспечения	Контроль зависимостей на уровне компонентов и отдельных файлов проекта	Подсчет базовых метрик (LOC и т.п.)	Свойста по недоступны	**Поиск сигнатур по регулярным выражениям (реализации на основе grep, sed)**
Абстрактное синтаксическое дерево (AST)	Лексический и синтаксический анализ	Лексический и синтаксический анализ	Лексический и синтаксический анализ	Лексический и синтаксический анализ, **Поиск сигнатур по AST**
Абстрактный семантический граф (ASG): Поток управления (control flow)	Определение связей объектов в иерархии	Подсчет метрик связности	Анализ потока выполнения (control flow)	**Поиск сигнатур последовательностей инструкций**
ASG: Поток данных (data flow)	Определение зависимостей по данным	Подсчет метрик связности	Абстрактная интерпретация (abstract interpretation)**:** Интервальный анализ Анализ указателей Анализ зависимостей по данным	Анализ потока данных (data flow) **Поиск внутрипроцедурных сигнатур последовательностей инструкций с учетом передаваемых значений**

Определение объектов для анализа

- Элементы языка
  - Модуль <module>
  - Макросы <macro>
  - Объект <object>
  - Атом <atom>
  - Ссылка <reference>
  - Выражение <expression>
  - Управление <flow>

Унификация представления конструкций языка

Вектор из трех наборов элементов представления языка

Вызов (C)	• (объекты, ссылки) элементов на которые передается управление
Вход (I)	• Источники информации (объекты, ссылки, атомы)
Выход (O)	• Приемники информации (объекты, ссылки, атомы)

Фрагмент программы на языке PHP с потенциально реализуемой SQL-инъекцией содержимого cookies клиентского браузера в СУБД PostgreSQL

```
<?php
...
$id = $_COOKIE["mid"]; //получение данных
...
$query = "SELECT MessageID FROM messages WHERE
MessageID = '$id'"; //формирование запроса к БД
...
$result = pg_query($conn, $query); //выполнение запроса

?>
```

Виды необходимых сигнатур

- Для выявления типов элементов
  - Типа «ссылка»
  - Типа «атом»
- Для выявления контекста элементов
  - Источник данных
  - Приемник данных
  - Источник перехода
- Для выявления потенциально опасных операций (пример: pg_query)
- Для выявления непроверенных источников данных (пример: _COOKIE)

Выполнение внутрипроцедурного анализа

Шаг 1:

```
1:(0,(”_COOKIE”),”$id”)
```

Шаг 2:

```
2:(0, ”$id”, “$query”)
```

Шаг 3:

```
3:(“pg_query”, “$query” ,“$result”)
```

Шаг 4:

```
4:(“pg_query”, “_COOKIE”,“$result”)
```

После шага 4 у нас срабатывают одновременно две сигнатуры критичных операций (получение данных из COOKIE и выполнение запроса к СУБД), и мы можем сигнализировать о наличии потенциально опасной конструкции: использование данных веденных от пользователя в SQL-запросах без предварительной фильтрации

Фрагмент программы преобразованный в форму CIO

```
<?php
...
$id = $_COOKIE["mid"];
...
$query = "SELECT MessageID FROM messages
WHERE MessageID = '$id'";
...
$result = pg_query($conn, $query);

?>
```

Вызовы (C)	Объекты-источники (I)	Объекты-приемники (O)
	$_COOKIE[“mid”] обращение к Cookies	$id
	$id	$query
pg_query, обращение к СУБД	$query	$result

Выводы по применению сигнатурного методов статического анализа

- Достоинства
  - относительная простота реализации анализаторов и баз сигнатур
  - высокая скорость работы (для уровня исходных текстов и AST-дерева)
  - легкость модификации сигнатур и их портирования на различные платформы и языки программирования
- Недостатки:
  - необходимость учитывать синтактическую «вариативность» различных языков программирования
  - сложность реализации качественного межпроцедурного анализа, сложность обработки конструкций «распределенных» по модулям ПО

Источники:

Выявление уязвимостей в программном коде. / А.С.Марков, С.В.Миронов, В.Л.Цирлов // Открытые системы. СУБД. 2005. № 12. С.64-69.

Методы оценки несоответствия средств защиты информации / Под ред. Маркова А.С. М.: Радио и связь, 2012. 192 с.

СПАСИБО ЗА ВНИМАНИЕ!

НПО «Эшелон»

mail@cnpo.ru

эшелон.рф