УДК 004.94

О ПРОГРАММНОЙ РЕАЛИЗАЦИИ АЛГОРИТМОВ ЗАМЫКАНИЯ БАЗОВОЙ ДП-МОДЕЛИ КОМПЬЮТЕРНОЙ СИСТЕМЫ С ДИСКРЕЦИОННЫМ УПРАВЛЕНИЕМ ДОСТУПОМ

И. В. Смит

В теоретическом анализе безопасности современных компьютерных систем (КС) широкое применение получили субъект-объектные ДП-модели П. Н. Девянина [1]. В работе [2] для них разработаны алгоритмы построения замыканий графов доступа, используемых для нахождения всех путей нарушения безопасности в КС, описываемых ДП-моделями.

С целью компьютерного представления ДП-моделей и программной реализации алгоритмов построения замыкания графа доступов базовой ДП-модели нами разработан класс `GRAPH` на языке программирования C++. В нем граф доступов представляется матрицей смежности M, элементы которой описаны типом данных `short`, и функцией иерархии H в виде класса `HFUNC`. Каждый элемент матрицы представляется набором бит, выставленных в соответствии с принадлежностью ребра множеству рёбер A, R или F, помеченных видами доступа, прав доступа или информационных потоков соответственно. Например, если в графе доступов есть ребро, определяющее наличие доступа на чтение сущности i к сущности j, то в элементе $M[i][j]$ выставлен бит, показывающий доступ на чтение. Принадлежность вершины к множеству контейнеров C, объектов O или субъектов S задается с помощью трёхбитовых элементов начальной строки матрицы. Таким образом, для установления вида того или иного ребра или типа той или иной вершины в графе доступов достаточно наложить заранее определенную маску на элемент матрицы смежности.

Функция иерархии задана таблицей соответствия подмножеств вершин графа (значений функции) индексам вершин (значений аргумента), перечисленных в заданном порядке по убыванию. В памяти это представлено в виде массива указателей на список индексов вершин графа. Использование списка оправдано динамичной мощностью данного множества. Кроме того, для ускорения построения замыкания графа функция иерархии содержит информацию об индексе вершины, стоящей выше по иерархии относительно аргумента. Это изменение в представлении функции из первоисточника сделано исключительно с целью повышения эффективности программы построения замыкания.

Для ввода графа доступов в память компьютера разработан специальный язык, описания на котором задаются построчно, где каждая строка начинается с определяющей множество последовательности, затем, через пробел, перечисляются элементы задаваемого множества. В контексте такого представления данных реализованы функции own-lock(), tgo-lock() и access-lock(), осуществляющие own-, tgo- и access-замыкания графа доступов базовой ДП-модели по алгоритмам 1, 2 и 3 соответственно, описанным в [2].

В реализации алгоритма 1 функцией own-lock() произвольный элемент (x, y, α) списка L имеет: x и y — индексы вершин графа доступов и α — двухбайтовая маска, соответствующая типу ребра, связывающего эти две вершины. В качестве множества N используется список индексов вершин.

Функция tgo-lock() реализует собственную часть алгоритма 2 как последовательность однотипных циклов, пробегающих по матрице смежности графа и применяю-

щих, где это возможно, правила преобразования *take_right* или *grant_right* без явного построения множеств R^{tgo} и F^{tgo}.

Аналогично реализуется собственная часть алгоритма 3 функцией access-lock() с применением в каждом цикле встречающегося правила преобразования *rename_entity*, *access_read*, *access_write* или *access_append*.

Применяемые правила преобразования графа доступов реализованы функциями, изменяющими биты матрицы смежности и значения функции иерархии H в соответствии с определениями этих правил в [1].

С целью проверки корректности работы программы замыкания (ею является функция access-lock()) были проведены тесты на графах размером до 10 вершин. Данная реализация может обработать граф до 30000 вершин на обычных пользовательских компьютерах без дополнительных модификаций исходного кода, однако время ожидания может быть слишком большим.

В продолжение работы планируется разработать средства задания графов доступов, реализовать графическую подсистему представления графа доступов и увеличить эффективность работы программы.

ЛИТЕРАТУРА

1. *Девянин П. Н.* Анализ безопасности управления доступом и информационными потоками в компьютерных системах. М.: Радио и связь, 2006.
2. *Колегов Д. Н.* Применение ДП-моделей для анализа защищенности сетей // Прикладная дискретная математика. 2008. №1. С. 71–88.

УДК 004.432.2

ТЕХНОЛОГИЯ И ИНСТРУМЕНТАЛЬНАЯ СРЕДА СОЗДАНИЯ ЗАЩИЩЁННЫХ СИСТЕМ ОБРАБОТКИ ИНФОРМАЦИИ

Д. А. Стефанцов

Сообщается о разработке и реализации технологии и инструментальной среды создания систем обработки информации (СОИ), защищённых политиками безопасности (ПБ) от угроз нарушения целостности, конфиденциальности и доступности обрабатываемых данных. Результатом применения разработанной технологии является программная система, состоящая из двух подсистем — защищаемой СОИ и модуля, реализующего заданную ПБ. Разработанные инструментальные средства позволяют создавать СОИ и ПБ независимо друг от друга с последующим объединением их с помощью соединительных модулей, обновлять ПБ и вносить в неё изменения и дополнения без изменения защищаемой СОИ.

В основу разработанной технологии положено аспектно-ориентированное программирование (АОП) — способ объединения независимых программных подсистем как аспектов, написанных на базовом языке программирования (C, Pascal, Smalltalk и т. п.), соединённых модулями, написанными на метаязыке программирования [1]. Инструментальная среда состоит из трёх основных частей:

1) языка программирования AspectTalk, обладающего конструкциями базового языка программирования Smalltalk и метаязыковыми конструкциями, основанными на протоколах метаобъектов [2];
2) виртуальной машины, выполняющей операции над данными, описанными на языке AspectTalk;
3) транслятора с языка AspectTalk в язык виртуальной машины.