Математические структуры и моделирование
2009, вып. 19, с. 97–103

УДК 65.012.810

ИССЛЕДОВАНИЕ БЕЗОПАСНОСТИ КОМПЬЮТЕРНЫХ СИСТЕМ В МОДЕЛИ ДИСКРЕЦИОННОГО РАЗДЕЛЕНИЯ ДОСТУПА HRU

Д.М. Бречка, С.В. Белим

Исследуется возможность расширения классической модели дискреционной безопасности HRU, разделение доступа в которой основывается на матрице доступов. Основной целью является выявление систем, удовлетворяющих требованиям безопасности.

Введение

Фундаментальным понятием в сфере защиты информации компьютерных систем является политика безопасности. Под ней понимают совокупность норм и правил, регламентирующих процесс обработки информации, выполнение которых обеспечивает состояние защищенности информации в заданном пространстве угроз. Формальное выражение политики безопасности называют моделью безопасности.

Среди программно-технических методов защиты информации выделяют в первую очередь разграничение доступа. Разграничение доступа непосредственно обеспечивает конфиденциальность информации, а также снижает вероятность реализации угроз целостности и правомерной доступности.

Под разграничением доступа к информации в компьютерных системах понимают разделение информации, циркулирующей в системе, на части, элементы, компоненты, объекты и т. д., и организацию такой системы работы с информацией, при которой пользователи имеют доступ только к той части (к тем компонентам) информации, которая им необходима для выполнения своих функциональных обязанностей или необходима исходя из иных соображений [3].

Наиболее распространенной и исторически первой является дискреционная политика безопасности. Дискреционная политика описывает отношения между субъектами и объектами компьютерной системы на основе прав, которые субъекты имеют над объектами. При этом под субъектами понимаются активные сущности системы, которые могут изменять состояние системы через порождение процессов над объектами, в том числе порождать новые объекты. Объекты

Омский государственный университет им. Ф. М. Достоевского.
E-mail: dbrechkawork@yandex.ru

— пассивные сущности системы. В дискреционных моделях исследуеются состояния системы на выявление возможности угроз информации [2].

1. Модель HRU

Модель Харисона-Руззо-Ульмана (HRU) является классическим примером дискреционной модели безопасности. Согласно этой модели компьютерная система представляется набором следующих сущностей:

— множество объектов (O) — пассивные сущности системы;

— множество субъектов (S) — активные сущности;

— права доступа (R) — действия, которые субъект может производить над объектом.

Определение 1. Матрица доступа — это таблица, в строках которой расположены субъекты системы, а в столбцах — объекты. Каждая ячейка матрицы доступа специфицирует права доступа субъектов к объектам.

Возможности изменения матрицы доступа определяется шестью примитивными операторами:

$-$*Enter* r *into* $A[s, o]$ — внести право r в ячейку $A[s, o]$;

$-$*Delete* r *from* $A[s, o]$ — удалить право r из ячейки $A[s, o]$;

$-$*Create subject* s — создать субъект s (т. е. новую строку матрицы A);

$-$*Create object* o — создать объект o (т. е. новый столбец матрицы A);

$-$*Destroy subject* s — уничтожить субъект s;

$-$*Destroy object* o — уничтожить объект o.

В результате выполнения примитивного оператора осуществляется переход системы из состояния $Q = (S, O, A)$ в новое состояние $Q' = (S', O', A')$.

Из примитивных операторов сторятся команды, состоящие из двух частей:

— условие, при котором выполняется команда;

— последовательность примитивных операторов.

Таким образом, команда имеет вид:

Command $c(x_1,\ x_2,\ ...,\ x_k)$
 if r_1 *in* $A[x_{s_1},\ x_{o_1}]$ *and*
 r_2 *in* $A[x_{s_2},\ x_{o_2}]$ *and*
 .
 .
 .
 r_m *in* $A[x_{s_m},\ x_{o_m}]$
 then $op_1,\ op_2,\ ...,\ op_n$

Каждое состояние системы Q_i является результатом выполнения некоторой команды c, применимой, согласно ее условиям, к предыдущему состоянию Q_{i-1}, и определяет отношения доступа, которые существуют между сущностями системы.

Безопасность системы определяется некоторыми условиями на начальное состояние системы Q_0, а также особенностями системы команд.

Определение 2. Система является безопасной относительно права r, если для заданного начального состояния $Q_0 = (S_0, O_0, A_0)$ не существует применимой к Q_0 последовательности команд, в результате которой право r будет занесено в ячейку матрицы $A[s, o]$, в которой оно отсутствовало в начальном состоянии Q_0.

Однако задача проверки данного критерия на истинность для произвольной системы алгоритмически неразрешима [1].

2. Примитивный базис модели HRU

Обозначим множество всех возможных матриц доступа через M. Команды HRU будут являться отображениями на этом множестве. Выделим базис системы команд HRU, для этого введем понятие базисного набора операторов.

Определение 3. Базисный набор операторов — это такой набор операторов, что любую команду HRU можно представить в качестве последовательности операторов из данного набора.

Теорема 1. *Примитивные операторы модели HRU образуют базисный набор.*

Доказательство. Покажем полноту системы примитивных операторов.

Так как работа команд HRU направлена исключительно на изменение матрицы доступа, покажем, что при помощи системы примитивных операторов можно построить любую матрицу доступа, то есть построить любую команду.

Рассмотрим матрицы A и A', принадлежащие множеству всех матриц доступа M. Преобразуем A в A' при помощи системы примитивных операторов. Для этого:

1. Применим оператор *Destroy object* к матрице A до тех пор, пока количество объектов в матрице A не станет равным нулю.

For $i = m$ *to* 0

Destroy object o_i,

где m — количество объектов в матрице A.

2. Применим оператор *Destroy subject* к матрице A до тех пор, пока количество субъектов в матрице A не станет равным нулю.

For $i = n$ *to* 0

Destroy subject s_i,

где n — количество субъектов в матрице A.

3. Применим оператор *Create subject* к матрице A до тех пор, пока количество субъектов в матрице A не станет равным количеству субъектов в матрице A'.

For $i = 0$ *to* n'

Create subject s_i,

где n' — количество субъектов в матрице A'.

4. Применим оператор *Create object* к матрице A до тех пор, пока количество объектов в матрице A не станет равным количеству объектов в матрице A'.

For $i = 0$ *to* m'

Create object o_i,

где m' — количество объектов в матрице A'.

5. Для каждого права $r \subseteq R$ применим оператор *Enter* r *into* $A[s, o]$, если право r содержится в соответствующей ячейке $A'[s', o']$.

For $i = 0$ *to* R
For $j = 0$ *to* m
For $k = 0$ *to* n
If r_i *in* $A'[j, k]$
Enter r_i *into* $A[j, k]$,

где R — количество всех возможных прав, m — количество объектов, n — количество субъектов.

Таким образом, при помощи примитивных операторов произвольно выбранная матрица $A \subseteq M$ может быть перобразована в произвольно выбранную матрицу $A' \subseteq M$, что доказывает полноту системы. ■

Как видно из доказательства теоремы, для построения полной системы используется только пять из шести примитивных операторов.

Следствие 1.1. *Оператор delete r можно представить в виде последовательности операторов enter r, create object o, create subject s, destroy object o, destroy subject s.*

Доказательство. Рассмотрим матрицы B и B', принадлежащие множеству всех матриц доступа M, такие, что количество строк и количество столбцов в этих матрицах равны. В какой-либо ячейке матрицы B' отсутствует право r_j, которое присутствует в соответствующей ячейке матрицы B. Все остальные ячейки матриц B и B' полностью идентичны. Преобразуем матрицу B в B' способом, аналогичным описанному в доказательстве теоремы 1. Данный способ позволяет преобразовать друг в друга произвольно выбранные матрицы, но не использует оператор *delete r*. Следовательно, для удаления права из матрицы B можно обойтись без оператора *delete r*. Это доказывает, что оператор *delete r* является избыточным и представляется в виде последовательности операторов *enter r*, *create object o*, *create subject s*, *destroy object o*, *destroy subject s*. ■

Введем понятие минимального набора операторов.

Определение 4. Минимальный базисный набор — это такой набор операторов, что ни один оператор данного набора не может быть представлен в качестве последовательности операторов из данного набора.

Теорема 2. *Набор примитивных операторов enter r, create object o, create subject s, destroy object o, destroy subject s является минимальным.*

Доказательство. Согласно определению, набор операторов не будет минимальным, если какой-либо оператор из набора можно представить в виде набора других операторов, принадлежащих даннму набору. Допустим, что рассматриваемый набор неминимален. Тогда для преобразования произвольно выбранной

матрицы $A \subseteq M$ в произвольно выбранную матрицу $A' \subseteq M$ можно будет обойтись без какого-либо оператора. Однако, как показано в доказателстве теоремы 1, для преобразования матриц необходимы все операторы рассматриваемого набора. Следовательно, рассматриваемый набор будет являться минимальным. ■

3. Построение базиса, отличного от примитивного

Рассмотрим возможность перехода к новому базису: представим какой-либо из операторов, входящих в базисный набор в виде последовательности операторов из другого набора.

Пусть S - множество субъектов системы, O - множество объектов, R - множество прав доступа. A - матрица доступа. Представим каждую ячейку матрицы A в виде последовательности бит таким образом, что каждому биту будет соответствовать какое-либо право из доступного набора прав R. Количество бит будет зависить от количества возможных прав, то есть если $|R| = l$, тогда для кодирования всех прав понадобится l бит. Наличие или отсутствие какого-либо бита означает наличие или отсутствие соответствующего права в данной ячейке.

Представим операцию добавления права в ячейку в виде последовательности логических функций. Будем применять логические функции к матрице доступа поэлементно, то есть для изменения права доступа субъекта s на объект o необходимо изменить ячейку $A[s, o]$.

Рассмотрим операцию добавления права $r_k \in R$ субъекту s на объект o. Для ячейки матрицы $A[s, o]$ необходимо будет выполнить логичесукю операцию «OR» с таким двоичным числом, в котором бит с номером k равен 1, а все остальные биты равны 0.

Логическая операция «OR» может быть выражена через логические операции «AND» и «NOT»: $OR(a, b) = NOT(AND(NOT(a), NOT(b)))$. Следовательно, операцию добавления права в ячейку $A[s, o]$ можно представить в виде двух логических операций «AND» и «NOT».

Таким образом можно получить новый базис, отличный от примитивного, он будет состоять из операторов *create object* o, *create subject* s, *destroy object* o, *destroy subject* s, $AND(a, b)$ и $NOT(a)$.

4. Монооперационные системы

Как уже говорилось ранее, для произвольной системы не существует алгоритма, проверяющего, является ли ее исходное состояние безопасным. Для рассмотрения условий, при которых такой алгоритм существует, вводится понятие монооперационной системы.

Определение 5. Система называется монооперационной, если каждая команда данной системы выполняет один примитивный оператор.

Теорема 3. *Существует алгоритм, проверяющий: является ли исходное состояние монооперационной системы безопасным по отношению к праву* $r \in R$.

Доказательство приводится в [1].

Расширим классическое понятие монооперационной системы с учетом базисного набора операций.

Определение 6. Монооперационная система в базисе B — это такая система, каждая команда которой содержит ровно один оператор из базиса B.

Формулировку теоремы 3 также можно сделать более общей.

Теорема 4. *Существует алгоритм, проверяющий: является ли исходное состояние монооперационной системы в базисе B безопасным по отношению к праву $r \in R$.*

Доказательство. Необходимо показать, что число последовательностей команд монооперационной системы в базисе, которые необходимо проверить, ограничено и сами команды имеют конечную длину. В этом случае алгоритмом проверки безопасности будет являться алгоритм перебора всех последовательностей команд и проверки их конечного состояния на отсутствие утечки права r.

Нет необходимости рассматривать команды, содержащие операторы *delete...* и *destroy...*, так как необходимо проверить наличие права после выполнения команды, а не его отсутствие. Также нет необходимости рассматривать последовательности, содержащие более одного оператора *create...*, так как все последовательности, которые проверяют или вносят права в новые элементы матрицы, могут быть заменой параметров в командах, представленных последовательностями, действующими с существующими субъектами и объектами. Одна команда создания субъекта необходима, если в начальном состоянии системы субъекты отсутствуют. Таким образом, необходимо рассмотреть только последовательности команд, которые содержат операторы *enter* r и один оператор *create subject*. Число различных операторов *enter* r можно вычислить следующим образом:

$n = |R|(|S_0| + 1)(|O_0| + 1)$.

Все команды, содержащие один и тот же оператор, но разные условия, объединяются в одну команду с составным условием.

Таким образом, число последовательностей команд, которые необходимо проверить равняется $n!$, при этом длина каждой команды равна n. ■

Непредсказуемость сложных систем является одним из главных недостатков модели HRU. Наложение условия монооперационности значительно сужает класс безопасных систем. При введении понятия базиса монооперационной системы появляется возможноть рассматривать более широкий класс систем, которые также будут являться безопасными. Для того чтобы проверить, является ли компьютерная система безопасной, достаточно найти такой базис, в котором она будет монооперационна.

Литература

1. Девянин, П.Н. Модели безопасности компьютерных систем / П.Н. Девянин – М.: издательский центр «Академя», 2005.

2. Гайдамакин, Н.А. Разграничение доступа к информации в компьютерных системах / Н.А. Гайдамакин – Уральский университет, 2003.
3. Грушо, А.А. Теоретические основы защиты информации /А.А. Грушо, Е.Е. Тимонина. – Агентство «Яхтсмен», 1996.