С. В. Белим, Д. М. Бречка

КЛАССЫ БЕЗОПАСНОСТИ В МОДЕЛИ HRU

Введение

Дискреционная политика безопасности и основанное на ней произвольное разграничение доступа являются минимальным требованием к компьютерной системе при определении класса ее защищенности. Данная политика безопасности строится на явном определении разрешенных и запрещенных доступов администратором безопасности компьютерной системы. Дискреционное разделение доступа задается, как правило, с помощью матрицы доступов, в которой строки соответствуют субъектам, т. е. активным сущностям компьютерной системы, а столбцы — объектам, т. е. хранилищам информации. Следует отметить, что субъекты также являются объектами.

На сегодняшний день разработан ряд математических моделей, позволяющих анализировать системы с дискреционным разделением доступа. Наиболее разработанными моделями являются HRU [1–6] и Take-Grant [4–7]. Эти модели подразумевают наличие матрицы доступа, но в каждой из них по-разному понимается безопасность системы. Модель HRU ставит перед собой задачу отслеживания преобразований матрицы доступов пользователями или процессами от имени пользователей. В рамках модели Take-Grant отслеживается распространение прав доступа в компьютерной системе на основе графа доступов.

В дальнейшем основное внимание в статье уделено модели HRU. Кроме базовой модели, предложенной впервые в работе [1], разработаны усложненные модели [6–9], посвященные типизированным матрицам доступа. В рамках базовой модели доказана теорема о невозможности построения алгоритма проверки произвольной системы на безопасность [2]. Однако там же выявлен класс систем, допускающих проверку на безопасность, эти системы принято называть монооперационными. Таким образом, актуальной является задача поиска других классов систем, для которых возможна проверка на безопасность.

В данной статье проводится дальнейшая формализация модели HRU, что позволяет обобщить класс монооперационных систем и таким образом выявить дополнительные условия, при которых компьютерная система с дискреционным разделением доступа является безопасной.

1. Матрица доступов

Классически матрица доступов определяется как таблица, ячейки которой задают набор разрешенных видов доступа для каждой пары субъект-объект компьютерной системы. Построим более строгую модель матрицы доступов.

Будем считать, что множество всех типов доступов в компьютерной системе определяется множеством $\boldsymbol{A}$. Будем считать, что $\boldsymbol{A}$ конечно: $|\boldsymbol{A}| < \infty$. Данное предположение достаточно правдоподобно и выполняется во всех реальных компьютерных системах. Введем множество всех возможных доступов $\boldsymbol{R}$ как множество всех возможных комбинаций из типов доступа. Легко понять, что $\boldsymbol{R} = 2^{\boldsymbol{A}}$.

Пусть множество $\boldsymbol{A}$ записывается в виде $\{a_1,a_2,\ldots,a_n\}$. Выберем для элементов множества R представление в виде n-битной строки $r = (b_1,b_2,\ldots,b_n)$. Причем $b_i = 1$, если тип доступа a_i входит в доступ r, и $b_i = 0$ в противном случае. Будем считать, что множество субъектов и множество объектов компьютерной системы являются счетными, т. е. может быть проведена нумерация всех возможных объектов и субъектов. Процедура сопоставления идентификационных номеров объектам компьютерной системы существенно зависит от архитектуры самой компьютерной системы. Однако в нашем построении достаточно предположения, что такая нумерация возможна.

Определим множество матриц над множеством $\boldsymbol{A}$ с количеством строк, не превышающим количество столбцов, которое в дальнейшем будем обозначать через $\boldsymbol{M}$. Очевидно, что каждой матрице доступов можно сопоставить матрицу из множества $\boldsymbol{M}$. Верно и обратное утверждение в силу того, что множество объектов включает в себя множество субъектов и, следовательно, для каждой рассматриваемой матрицы можно построить реализацию компьютерной системы.

Изменение защиты компьютерной системы сводится к преобразованию матрицы доступов. Такие преобразования можно записать в виде операций на множестве $\boldsymbol{M}$. Будем считать, что преобразования также зависят и от некоторого набора параметров. Множество всех возможных наборов значений параметров обозначим через $\boldsymbol{P}$. Тогда любое преобразование матрицы доступов будет записано в виде операции:

$o\rho: \boldsymbol{M} \times \boldsymbol{P} \rightarrow \boldsymbol{M}$.

Определим суперпозицию операций $o\rho_1 \bullet o\rho_2 \bullet \ldots \bullet o\rho_n$ как их последовательное применение к некоторой матрице доступов $M \in \boldsymbol{M}$.

Определение. Базисом над множеством множества матриц доступа $\boldsymbol{M}$ будем называть минимальный набор операций, через суперпозицию которых может быть определена любая операция на множестве $\boldsymbol{M}$.

Рассмотрим один из возможных примеров базиса над множеством $\boldsymbol{M}$. При этом введены обозначения: $\boldsymbol{S}$ — множество субъектов компьютерной системы, O — множество объектов компьютерной системы, $M[S,O]$ — элемент матрицы доступов, определяющий права доступа субъекта $S \in \boldsymbol{S}$ к объекту $O \in \boldsymbol{O}$.

1. *AddO(O,M)* — добавить столбец, соответствующий объекту O, в матрицу доступов M. Условие выполнения операции: $O \in \boldsymbol{O}$. Новое состояние компьютерной системы запишется в виде: $\boldsymbol{O'} = \boldsymbol{O} \cup \{O\}$, $\boldsymbol{S'} = \boldsymbol{S}$, $\forall S \in \boldsymbol{S}$: $M[S,O] = \varnothing$.

2. *DelO(O,M)* — удалить столбец, соответствующий объекту O, в матрице доступов M. Условие выполнения операции: $O \in \boldsymbol{O}$. Новое состояние компьютерной системы запишется в виде: $\boldsymbol{O'} = \boldsymbol{O} \setminus \{O\}$, $\boldsymbol{S'} = \boldsymbol{S}$.

3. *AddS(S,M)* — добавить строку и столбец, соответствующие субъекту S, в матрицу доступов M. Условие выполнения операции: $S \notin \boldsymbol{S}$. Новое состояние компьютерной системы запишется в виде: $\boldsymbol{O'} = \boldsymbol{O} \cup \{S\}$, $\boldsymbol{S'} = \boldsymbol{S} \cup \{S\}$, $\forall O \in \boldsymbol{O'}$: $M[S,O] = \varnothing$.

4. *DelS(S,M)* — удалить строку и столбец, соответствующие субъекту S, в матрице доступов M. Условие выполнения операции: $S \in \boldsymbol{S}$. Новое состояние компьютерной системы запишется в виде: $\boldsymbol{O'} = \boldsymbol{O} \setminus \{S\}$, $\boldsymbol{S'} = \boldsymbol{S} \setminus \{S\}$.

5. *Inv(M[S,O],k)* — инвертировать в элементе матрицы доступов $M[S,O]$ бит с номером k. По сути эта операция выдает или отменяет право доступа a_k. Условие выполнения операции: $O \in \boldsymbol{O}$, $S \in \boldsymbol{S}$. Новое состояние компьютерной системы запишется в виде: $\boldsymbol{O'} = \boldsymbol{O}$, $\boldsymbol{S'} = \boldsymbol{S}$, $M'[S,O] = M[S,O] \otimes 2^{k-1}$. Здесь $\otimes$ побитовая логическая операция «Исключающее ИЛИ».

Набор операций { *AddO(O,M)*, *DelO(O,M)*, *AddS(S,M)*, *DelS(S,M)*, *Inv(M[S,O],k)* } в дальнейшем будем обозначать через B_I.

Теорема 1. Набор операций B_I является базисом над множеством матриц доступа $\boldsymbol{M}$.

Доказательство. Для доказательства теоремы необходимо проверить полноту B_I, т. е. возможность представления любой операции не из B_I через суперпозицию операций из B_I, а также минимальность, т. е. невозможность представления любой операции из B_I через другие операции из B_I.

Полнота B_I следует из того, что с помощью операций *AddO(O,M)*, *DelO(O,M)*, *AddS(S,M)*, *DelS(S,M)* можно изменить размер любой матрицы доступа до заданных значений. С помощью операции *Inv(M[S,O],k)* можно поменять значение любой ячейки матрицы доступов.

Следовательно, операции базиса B_1 позволяют преобразовать любую матрицу доступа к заданному виду, т. е. заменить любую операцию.

Минимальность очевидна. ■

2. Модель HRU

Рассмотрим модель HRU и ее представление в терминах операций над множеством матриц доступа. В данной модели ключевую роль играют примитивные операторы, из которых строятся команды преобразования матрицы доступов. В результате выполнения примитивного оператора осуществляется переход системы из состояния $Q = (\boldsymbol{S}, \boldsymbol{O}, \boldsymbol{M})$ в новое состояние $Q' = (\boldsymbol{S'}, \boldsymbol{O'}, \boldsymbol{M'})$. Примитивные операторы модели HRU можно рассматривать как операции над множеством матриц доступа, следовательно, по теореме 1, они могут быть выражены через операции базиса B_1. Проведем соответствующие построения:

1. *Enter* a_k *into* $M[S,O]$ — ввести право a_k в ячейку $M[S,O]$:
Enter a_k *into* $M[S,O] = Inv(M[S,O],k)$.
2. *Delete* a_k *from* $M[S,O]$ — удалить право a_k из ячейки $M[S,O]$:
Delete a_k *from* $M[S,O] = Inv(M[S,O],k)$.
3. *Create subject* S — создать субъект S (т. е. новую строку матрицы M):
Create subject $S = AddS(S,M)$.
4. *Create object* O — создать объект O (т. е. новый столбец матрицы M):
Create object $O = AddO(O,M)$.
5. *Destroy subject* S — уничтожить субъект S:
Destroy subject $S = DelS(S,M)$.
6. *Destroy object* O — уничтожить объект O:
Destroy object $O = DelO(O,M)$.

Первый и второй примитивные операторы HRU представляются одинаково через операцию $Inv(M[S,O],k)$, однако условие выполнения у них разное. В первом случае право a_k первоначально отсутствует в ячейке матрицы доступов, а во втором случае, наоборот, присутствует.

Как видим, примитивных операторов HRU на один больше, чем базисных операций, следовательно, они не являются базисом. Построим базис на основе примитивных операторов HRU.

Из примитивных операторов могут быть построены команды, состоящие из двух частей:

1. условие, при котором выполняется команда;
2. последовательность примитивных операторов.

Общий вид команды:

Command $c(x_1, x_2, ..., x_k)$
<составное условие выполнения команды>
<последовательность примитивных операторов>
end command;

Каждое состояние системы Q_i является результатом выполнения некоторой команды c к предыдущему состоянию Q_{i-1}.

Набор операций { *Enter* a_k *into* $M[S,O]$, *Create subject* S, *Create object* O, *Destroy subject* S, *Destroy object* O} обозначим B_H.

Теорема 2. Набор операций B_H является базисом над множеством матриц доступа.

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

Рассмотрим матрицы M и M', принадлежащие множеству всех матриц доступа $\boldsymbol{M}$. Преобразуем M в M' при помощи системы примитивных операторов.

Алгоритм преобразования можно записать с помощью следующей последовательности шагов:

1. Многократно применим оператор *Destroy object O* к матрице M до тех пор, пока количество объектов в матрице M не станет равным нулю.

2. Многократно применим оператор *Destroy subject S* к матрице M до тех пор, пока количество субъектов в матрице M не станет равным нулю.

3. Многократно применим оператор *Create subject S* к матрице M до тех пор, пока количество субъектов в матрице M не станет равным количеству субъектов в матрице M'.

4. Многократно применим *Create object O* к матрице M до тех пор, пока количество объектов в матрице M не станет равным количеству объектов в матрице M'.

5. Для каждого права a_k применим оператор *Enter* a_k *into* $M[S,O]$, если право a_k содержится в соответствующей ячейке $M'[S,O]$.

Таким образом, при помощи примитивных операторов произвольно выбранная матрица $M \in \boldsymbol{M}$ может быть преобразована в произвольно выбранную матрицу $M' \in \boldsymbol{M}$, что доказывает полноту системы.

Докажем минимальность набора $B_{\text{Н}}$. Согласно определению, набор операторов не будет минимальным, если какой-либо оператор из набора можно представить в виде набора других операторов, принадлежащих данному набору. Допустим, что рассматриваемый набор не минимален. Тогда для преобразования произвольно выбранной матрицы $M \in \boldsymbol{M}$ в произвольно выбранную матрицу $M' \in \boldsymbol{M}$ можно будет обойтись без какого-либо оператора. Однако, как показано в доказательстве полноты набора операторов, для преобразования матриц необходимы все операторы рассматриваемого набора. Следовательно, рассматриваемый набор операторов будет минимальным. ■

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

Следствие. Примитивный оператор *Delete* a_k *from* $M[S,O]$ может быть представлен в виде суперпозиции других примитивных операторов модели HRU.

Доказательство. Рассмотрим матрицы M и M' такие, что количество строк и количество столбцов в этих матрицах равны. В какой-либо ячейке матрицы M' отсутствует право a_k, которое присутствует в соответствующей ячейке матрицы M. Все остальные ячейки матриц M и M' полностью совпадают. Преобразуем матрицу M в M' способом, описанным в доказательстве теоремы 2. С другой стороны, то же преобразование выполняет оператор *Delete* a_k *from* $M[S,O]$. Следовательно, для удаления права из матрицы M можно обойтись без оператора *Delete* a_k *from* $M[S,O]$. Это доказывает, что оператор *Delete* a_k *from* $M[S,O]$ является избыточным и представляется в виде последовательности операторов из набора $B_{\text{Н}}$. ■

3. Монооперационные системы

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

Определение. Система является безопасной относительно права a_k, если для заданного начального состояния Q_0 не существует последовательности команд, в результате которой право a_k будет занесено в ячейку матрицы доступов M, в которой оно отсутствовало в начальном состоянии Q_0.

Задача проверки данного критерия на истинность для произвольной системы алгоритмически неразрешима [2]. Однако можно выделить отдельные классы систем, для которых возможно

построение алгоритма, проверяющего безопасность системы. Важным примером являются монооперационные системы.

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

Доказательство безопасности монооперационных систем приведено в [6]. Расширим понятие монооперационной системы с учетом существования базисного набора операций.

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

Теорема 3. Существует алгоритм, проверяющий, является ли исходное состояние монооперационной системы в базисе B безопасным по отношению к праву a_k.

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 алгоритм перебора всех последовательностей команд и проверки их конечного состояния на отсутствие утечки права a_k.

Нет необходимости рассматривать команды, содержащие операторы *delete* и *destroy*, так как нужно 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* и один оператор *create subject*. Число различных операторов *enter* можно вычислить следующим образом:

$n=|A|(|\boldsymbol{S}_0|+1)(|\boldsymbol{O}_0|+1)$.

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 возможность 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 монооперационной.

4. Примеры построения базисов преобразования матрицы доступов

Рассмотрим еще два базиса преобразований, строящихся с помощью логических операций. Представим операцию добавления права в ячейку в виде последовательности логических функций. Будем применять логические функции к матрице доступа поэлементно, т. е. для изменения права доступа субъекта S на объект O необходимо изменить ячейку $M[S,O]$. Рассмотрим операцию добавления права a_k субъекту S на объект O. Для ячейки матрицы $M[S,O]$ необходимо будет выполнить логическую операцию «OR» (логическое побитовое «ИЛИ») 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 M[S,O] можно представить в виде двух логических операций «AND» и «NOT». Так можно получить новый базис.

Заключение

Таким образом, класс безопасных систем с дискреционным разделением доступа может быть расширен путем построения новых базисов преобразований. В данной статье были рассмотрены только новые операции, заменяющие операцию внесения некоторого права в ячейку матрицы доступов. Однако можно построить преобразования матриц доступа, меняющие размер матрицы и не совпадающие с добавлением нового объекта и нового субъекта.

СПИСОК ЛИТЕРАТУРЫ:

1. *Harrison M., Ruzzo W.* Monotonic Protection System / R. In DeMillo, D. Dobkin, A. Jones, R. Lipton, editors // Foundation of Secure Computation. New York: Academic Press, 1975.
2. *Harrison M. A., Ruzzo W. L. and Ullman J. D.* On Protection in Operating Systems // Communications of the ACM. 1975. P. 14–25.
3. *Грушо А. А., Тимонина Е. Е.* Теоретические основы защиты информации. М.: Яхтсмен, 1996. – 192 с.
4. Теория и практика обеспечения информационной безопасности / Под ред. П. Д. Зегжды. М.: Яхтсмен, 1996. – 302 с.
5. *Зегжда Д. П., Ивашко А. М.* Основы безопасности информационных систем. М.: Горячая линия – Телеком, 2000. – 452 с.
6. *Девянин П. Н.* Модели безопасности компьютерных систем. М.: Издательский центр «Академия», 2005. – 144 с.
7. *Bishop M.* Theft of Information in the Take-Grant Protection System // J. Computer Security. 1994/1995. 3 (4).
8. *Ravi S. Sandhu.* The Typed Access Matrix Model // Proceedings of IEEE Symposium on Security and Privacy. Oakland. California. May 4–6, 1992. P. 122–136.
9. *Jones A., Lipton R., Snyder L.* A Linear Time Algorithm for «Deciding Security» // Proc. 17th Annual Symp. on the Foundations of Computer Science (Oct. 1976).