Математические структуры и моделирование
2009, вып. 20, с. 153–159

УДК 65.012.810

ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ ПОДХОД В ПОСТРОЕНИИ ДИСКРЕЦИОННОЙ ПОЛИТИКИ БЕЗОПАСНОСТИ

С.В. Белим, С.В. Усов

Проведена модификация модели дискреционного разделения доступа HRU для объектно-ориентированных компьютерных систем.

Введение

Традиционно рассматриваемая модель системы безопасности компьютерных систем строится на основе субъектно-объектного подхода. Однако в последнее время, в связи с распространением объектно-ориентированного подхода в построении программного обеспечения компьютерных систем, складывается двойственная ситуация, когда один и тот же набор данных в одном случае интерпретируется как объект (пассивный), в другом случае как субъект (активный). В связи с этим, для более адекватного описания компьютерных систем, необходимо применение объектно-ориентированного подхода и соответствующая модификация моделей политик безопасности. В частности, объектно-ориентированная модель компьютерной системы построена в работе [5], где даны и необходимые определения.

В случае объектно-ориентированной модели компьютерной системы, как и для субъектно-объектного подхода, возможно определение элементарных операторов в рамках модели безопасности HRU [3]. Поскольку объектно-ориентированный подход предполагает разбиение объектов на классы, задаваемые списком полей и методов, то при построении модели HRU мы должны научиться обращаться не только с представителями классов, но и с самими классами. Во-первых, рассматривая классы в качестве реализации типов из модели типизированной матрицы доступов (TAM), построенной на основе HRU, мы получаем менее жесткие условия в критерии безопасности системы [6]. Во-вторых, возникает два варианта определения как локальных (привязанных к каждому объекту) матриц доступа, так и элементарных операторов, преобразующих эти матрицы.

Первый подход, классово-типизированный, отождествляет класс объекта с его типом, и матрица доступа к каждому объекту каждого класса строится как

Омский государственный университет им. Ф.М. Достоевского.
E-mail: belim@univer.omsk.su

ТАМ с учетом объектно-ориентированного подхода к строению компьютерной системы.

Второй подход, наследственно-типизированный, обращается к наследственной структуре объектов в объектно-ориентированной среде. В этом случае все объекты одного класса, а значит, одного типа, являются эквивалентными с точки зрения набора прав доступа, и матрицу доступа достаточно задать для класса целиком. С другой стороны, множество типов является частично упорядоченным, причем порядок задается наследственностью соответствующих типам классов. Более того, наследственность проявляется и в разграничении прав доступа: объект класса-наследника обладает всеми правами класса-родителя. Также имеет смысл ввести запрет на доступ к полю (методу) класса-наследника для объекта, у которого нет права доступа к соответствующему полю (методу) класса-родителя.

Однако в данной работе мы ограничимся построением объектно-ориентированной модели системы безопасности в рамках классово-типизированного подхода и выявлением случаев, допускающих проверку безопасности компьютерной системы.

1. Элементарные операторы модели HRU в объектно - ориентированной компьютерной системе

Будем рассматривать компьютерную систему в виде множества объектов O, имеющих открытые поля f и скрытые поля p, а также методы обработки полей s.

Все необходимые определения даны в работе [5], там же построена и модель системы безопасности для объектно-ориентированной компьютерной системы.

Матрица доступа M в данном случае - скрытое (private) поле объекта. От объекта к объекту даже одного класса (типа) содержание этого поля может отличаться. Строки матрицы объекта o_i соответствуют объектам компьютерной системы, столбцы – открытым (public) полям и методам объекта o_i, элементом матрицы является некоторое подмножество множества R видов доступа к полю, если столбец элемента соответствует public полю объекта, и принимает значение 0 либо 1, если столбец элемента соответствует методу объекта, где "1" разрешает вызов метода, а "0" - запрещает.

Определим следующие элементарные операции для работы с матрицей доступов:

1) $Create(o_j, \tau)$ – создает объект o_i класса $\tau \in \mathbf{T}$, при этом $O' = O \cup \{o_j\}$, а в матрице доступа объектов o_i $(i \neq j)$ добавляется строка, соответствующая объекту o_j, причем $\forall f \in o_i.F$ $o_i.M'[o_j, o_i.f] = \emptyset$, $\forall s \in o_i.S$ $o_i.M'[o_j, o_i.s] = \emptyset$. Матрица доступа объекта o_j создается полностью пустой. 2) $Destroy(o_j)$ – уничтожает объект o_j, при этом $O' = O \diagdown \{o_j\}$, а в матрице доступа объектов o_i $(i \neq j)$ уничтожается строка, соответствующая объекту o_j. 3) $Enter(r, o_j, o_i.f)$ - вносит право доступа r в $o_i.M[o_j, o_i.f]$, при этом матрица доступа изменяется по правилу: $o_i.M'[o_j, o_i.f] = o_i.M[o_j, o_i.f] \cup \{r\}$. 4) $Delete(r, o_j, o_i.f)$ – удаляет право r доступа из $o_i.M[o_j, o_i.f]$, при этом матрица доступа изменяется по

правилу: $o_i.M'[o_j, o_i.f] = o_i.M[o_j, o_i.f] \setminus \{r\}$. 5) $Grant(o_j, o_i.s)$ – разрешает вызов объекту o_j метода $o_i.s$, при этом матрица доступа изменяется по правилу: $o_i.M'[o_j, o_i.s] = 1$. 6) $Deprive(o_j, o_i.s)$ – запрещает вызов объекту o_j метода $o_i.s$, при этом матрица доступа изменяется по правилу: $o_i.M'[o_j, o_i.s] = \emptyset$.

В дальнейшем будем сокращать запись $o_i.M[o_j, o_i.f]$ до $M_i[j, f]$ и запись $o_i.M[o_j, o_i.s]$ до $M_i[j, s]$.

2. Определения

Определение 1. Под состоянием компьютерной системы в момент времени t будем понимать пару $Q(t) = (O(t), M(t))$, где $O(t) = \{o_j\}$ – множество всех объектов системы в момент времени t, а $M(t) = \{o_j.M[](t)\}$ – семейство всех матриц доступа объектов системы в состоянии на момент t. Начальное состояние системы будем обозначать $Q(0)$.

Состояния компьютерной системы в модели HRU изменяются под воздействием запросов на модификацию матрицы доступа в виде команд следующего формата:

Команда $\gamma(x_1, \ldots, x_\gamma)$:

if — условная часть (представляет собой конъюнкцию значений логических выражений вида $r \in M_i[j, f]$ или $M_i[j, s] = 1$)

$then$ — последовательность элементарных операций.

Здесь аргументы x_k представляют собой либо объекты, либо поля или функции объектов, участвующие в качестве аргументов в условной части либо в элементарных операциях.

Факт перехода системы под действием команды γ из состояния $Q(t)$, в котором она находилась в момент времени t, в состояние $Q(t+1)$, в котором она находилась в следующий момент времени $t+1$, будем записывать в сокращенном виде: $Q(t) \to_\gamma Q(t+1)$.

Определение 2. HRU-модель системы безопасности называется монооперационной, если каждая команда в этой системе содержит только одну элементарную операцию.

Определение 3. HRU-модель системы безопасности называется монотонной, если каждая команда в этой системе содержит только одно условие.

Определение 4. HRU-модель системы безопасности объектно-ориентированной компьютерной системы называется монотонной, если команды этой системы не содержат операций *Delete*, *Deprive* и *Destroy*.

Определение 5. Будем говорить, что состояние системы $Q(t)$ допускает утечку права доступа $r \in R$, если существуют такие команда γ, объект o_j и поле $o_i.f$ объекта o_i, что $r \notin M_i[j, f](t)$, но $r \in M_i[j, f](t+1)$, где $Q(t) \to_\gamma Q(t+1)$.

Определение 6. Будем говорить, что состояние системы $Q(t)$ допускает утечку права вызова, если существуют такая команда γ, объект o_j и метод $o_i.s$ объекта o_i, что $M_i[j, f](t) = 0$, но $M_i[j, f](t+1) = 1$, где $Q(t) \to_\gamma Q(t+1)$.

Определение 7. Будем говорить, что система r-безопасна, если не существует такой последовательности команд $\gamma_1 \dots \gamma_T$, что $Q(t-1) \rightarrow_{\gamma_t} Q(t)$, $t = 1, 2 \dots T$, и $Q(T)$ допускает утечку права r либо утечку права вызова.

3. Случаи, допускающие проверку безопасности модели HRU объектно-ориентированной компьютерной системы

Попытаемся ответить на вопрос, в каких случаях возможно решить проблему r-безопасности системы для объектно-ориентированной системы. Разрешимость аналогичной проблемы для HRU-моделей субъектно-объектных систем уже была доказана для некоторых частных случаев. А именно в [4] - для монооперационных систем и в [2] – для монотонных моноусловных систем. Наша цель – повторить эти результаты в рамках HRU-модели объектно-ориентированной системы.

Теорема 1. *Проблема r-безопасности системы является NP-разрешимой для монооперационной объектно-ориентированной модели.*

Доказательство. Если мы покажем, что для разрешения проблемы r-безопасности достаточно проверить конечное число последовательностей команд конечной длины, теорема будет доказана.

Без ограничения общности считаем, что нам достаточно обнаружить только утечку права доступа к полю объекта, поскольку право вызова метода объекта можно интерпретировать как особое и единственное право доступа к этому методу.

Поскольку система монооперационна, мы можем называть команду по единственной элементарной операции, которую она содержит.

Будем искать цепочку команд наименьшей длины, переводящую систему в состояние, допускающее утечку некоторого права r. Заметим, что такая цепочка не должна содержать команд, не изменяющих ни одну матрицу доступа (как, например, команда, добавляющая право r в ячейку матрицы доступа, в которой право r уже имеется).

Допустим,

$$Q(0) \rightarrow_{\gamma_1} Q(1) \rightarrow \dots \rightarrow_{T-1} Q(T-1) \rightarrow_{\gamma_T} Q(T),$$

где $Q(T)$ допускает утечку права $r \in R$ в ячейку $M_i[j, f](T)$.

Обратим внимание, что условия команд проверяют лишь наличие права в ячейке матрицы доступа, а не его отсутствие. Таким образом, если мы выкинем все команды Delete и Destroy из цепочки, все права доступа, находившиеся в определенной ячейке в состоянии $Q(T)$, останутся там и после удаления указанных команд. Возможно, появятся новые права – и это может привести к появлению состояния, допускающего утечку права r в момент $t < T$, что только укоротит цепочку.

Однако, если для некоторого $t < T$ право r лежало в $M_i[i,f](t)$, удаление команд $Delete(r, o_j, o_i.f)$, $Destroy(o_j)$, $Destroy(o_i)$, выполненных после момента t, приведет к тому что право r останется в $M_i[j,f](T)$ и состояние $Q(T)$ более не будет допускать утечки этого права. В этом случае хотя бы одну из перечисленных команд необходимо оставить в цепочке.

Отыщем первую команду $Create$ в рассматриваемой цепочке.

(а) Такой команды нет. Значит, все команды Destroy можно удалить, поскольку среди них отсутствуют команды $Destroy(o_j)$, $Destroy(o_i)$: если бы таковые присутствовали, ячейки $M_i[i,f](T)$ просто бы не существовало. Возможно, в цепочке присутствует команда $Delete(r, o_j, o_i.f)$. Все остальные команды - это команды $Enter$, и их не более $|R||O|^2N$. Здесь N – максимальное количество полей и методов в классе объектно-ориентированной модели. Заметим, что эту оценку можно улучшить, если различать в доказательстве поля и методы класса до следующей: $(|R|N_f + N_s)|O|^2$, где N_f – максимальное количество полей, а N_s - максимальное количество методов в классе объектно-ориентированной модели. Действительно, в матрицу доступа объекта не удастся внести более $(|R|N_f + N_s)|O|$ прав, а команды, не изменяющие матрицы доступа, мы из цепочки выкидываем.

(б) Нашлась команда $Create(o_k, \tau_k)$, $\tau_k \in \mathbf{T}$. Если $o_k \notin O(0)$, то перед ней нет команды $Destroy(o_k)$, а после нее все такие команды можно удалить.

Если $o_k \in O(0)$, k отлично от i и j, то перед командой $Create(o_k, \tau_k)$ найдется ровно одна команда $Destroy(o_k)$, и мы можем удалить эту пару команд (тем самым сократив цепочку, переводящую систему в состояние, допускающее утечку права), при этом все права в матрице доступов сохранятся, и к ним добавятся новые, ранее удаленные со строкой объекта o_k. После команды $Create(o_k, \tau_k)$ все команды $Destroy(o_k)$ также можно удалить.

Наконец, $o_k \in O(0)$, $k = i$ или $k = j$. По соображениям, изложенным выше, одну пару команд вида $Destroy(o_k)$ - $Create(o_k, \tau_k)$ (если их несколько, то первую) придется оставить. Остальные из цепочки можно удалить: действительно, если в момент выполнения t команды $Destroy(o_l)$, где $l = i$ или $l = j$, $r \in R$ лежит в $M_i[j,f](t)$, то утечка права уже произошла (и цепочку можно сократить), если же $r \notin M_i[j,f](t)$, удаление пары $Destroy(o_l)$ - $Create(o_l, \tau_l)$ не вызовет появления права $r \in R$ в $M_i[j,f]$ вплоть до момента t_1 выполнения команды $Create(o_l, \tau_l)$. По аналогичной причине команду $Delete(r, o_j, o_i.f)$ можно удалить так же.

Далее, не имеет смысла оставлять более одной команды вида $Create(o_k, \tau_k)$ для каждого класса $\tau_k \in \mathbf{T}$. Действительно, второй созданный класс o_l того же типа τ_k можно отождествить с первым, o_k, выкинув из цепочки операцию $Create(o_l, \tau_k)$ и заменив далее в аргументах команд цепочки o_l на o_k. Такое отождествление остается в силе, даже если o_l ранее был разрушен, а теперь воссоздан, а l совпадает с j (или i): действительно, созданная с нуля строка для o_j (матрица доступа для o_i) не может содержать права доступа, отсутствующие в соответствующей строке M_k (матрице M_k).

Итак, в цепочке осталось не более $|\mathbf{T}|$ команд $Create$, не более одной команды $Destroy$ или $Delete$ и конечное число команд $Enter$, а именно не более

$|R|(|O|+|\mathbf{T}|+1)^2N + |R|(|O|+|\mathbf{T}|)N$. Здесь второе слагаемое отвечает за заполнение строк, соответствующих ранее удаленному, а теперь воссозданному объекту o_i (или o_j).

Таким образом, теорема доказана. ■

Теорема 2. *Проблема r-безопасности системы является NP-разрешимой для моноусловной монотонной объектно-ориентированной модели.*

Доказательство. Если мы покажем, что для разрешения проблемы r-безопасности достаточно проверить конечное число последовательностей команд конечной длины, теорема будет доказана.

Без ограничения общности считаем, что нам достаточно обнаружить только утечку права доступа к полю объекта, поскольку право вызова метода объекта можно интерпретировать как особое и единственное право доступа к этому методу.

Будем искать цепочку команд наименьшей длины, переводящую систему в состояние, допускающее утечку некоторого права r. Заметим, что такая цепочка не должна содержать команд, не изменяющих ни одну матрицу доступа (как, например, команда, добавляющая право r в ячейку матрицы доступа, в которой право r уже имеется).

Допустим,

$$Q(0) \to_{\gamma_1} Q(1) \to \cdots \to_{T-1} Q(T-1) \to_{\gamma_T} Q(T),$$

где $Q(T-1)$ допускает утечку права $r \in R$ в ячейку $M_i[j,f](T-1)$ посредством выполнения команды γ_T.

Условие выполнения команды γ_T требует наличия права доступа r_1 в ячейке $M_{i_1}[j_1,f_1](T-1)$, которое добавлено туда некоторой командой γ_t, $t<T$. Таким образом, команды $\gamma_{t+1}\dots\gamma_{T-1}$ можно выкинуть из цепочки, поскольку они заполняют ячейки, содержание которых нам не потребуется для утечки права r. Оставим лишь команду (или две), содержащую элементарную операцию $Create(o_k,\tau_k)$ (k может совпадать с i или с j), если она встретилась на выкидываемом отрезке цепочки. Далее условие команды γ_t требует наличия права доступа r_2 в ячейке $M_{i_2}[j_2,f_2](t-1)$, которое добавлено туда некоторой командой γ'_t, $t'<t$, и мы вновь можем отказаться от промежуточных команд, кроме одной-двух $Create$. Действуя таким образом, доберемся наконец до команды, условие которой выполнялось уже в начальном состоянии системы. Далее считаем, из цепочки

$$Q(0) \to_{\gamma_1} Q(1) \to \cdots \to_{T-1} Q(T-1) \to_{\gamma_T} Q(T)$$

уже выкинуты лишние команды.

Теперь добьемся того, чтобы в условии команд цепочки встречалось не более одного нового (то есть отсутствовавшего в начальном состоянии системы) объекта каждого типа. Допустим, первым был создан объект o_k типа $\tau_k \in \mathbf{T}$, а аргументом некоторых команд γ_t является объект o_l того же типа, созданный позже. Подставив вместо него o_k, получаем команды γ'_t на месте γ_t в цепочке,

причем все права, находившиеся в строчках o_l матриц доступа (и в матрице доступа объекта o_l) теперь находятся в строчках o_k матриц доступа (и в матрице доступа объекта o_k). Учитывая, что там же находятся и права доступа самого объекта o_k, утечка права доступа произойдет в момент T или даже раньше.

Итак, каждая из команд цепочки либо создает один из не более чем $\mathbf{T}$ объектов, которые встретятся в условиях команд цепочки, либо заносит право в ячейку матрицы доступа уже существующего объекта, причем наличие этого права также потребуется в условии одной из команд цепочки. На самом деле команда может выполнять сразу несколько из этих «полезных» элементарных операций, что приведет лишь к сокращению длины цепочки. В любом случае, ее длина не превышает $|R|(|O| + |\mathbf{T}|)^2 N + |\mathbf{T}|$. ■

Теорема 3. *Проблема r-безопасности системы неразрешима для произвольной объектно-ориентированной модели.*

Доказательство данной теоремы практически не отличается от классического доказательства со сведением к проблеме остановки машины Тьюринга, предложенного в работе [3]. Отличие заключается лишь в том, что лента машины Тьюринга собирается из ячеек не одной матрицы доступа, как было в случае стандартной HRU-модели, а нескольких.

Литература

1. Harrison, M.A. Theoretical issues concerning protection in operating systems / M.A. Harrison. – University of California at Berkeley, CSD-84-170, 1984.
2. Harrison, M.A. Monotonic protection systems / M.A. Harrison, W.L. Ruzzo / In R.A. DeMillo, D.P. Dobkin, A.K. Jones, R.J. Lipton, editors. – Foundations of Secure Computation. – Academic Press, Inc., 1978. – P. 461–471.
3. Harrison, M.A. Protection in operating systems / M.A. Harrison, W.L. Ruzzo, J.D. Ullman. – Communications of the ACM, 19(8):461–471, August 1976.
4. Tripunitara, M.V. The Foundational work of Harrison-Ruzzo-Ullman Revisited / M.V. Tripunitara, N. Li. – 2006.
5. Белим, С.В. Объектно-ориентированная модель компьютерной системы / С.В. Белим, С.Ю. Белим // Математические структуры и моделирование. – 2008. – Вып. 18. – С. 98-101.
6. Sandhu, R.S. The Typed Access Matrix Model / R.S. Sandhu // Proceedings of IEEE Symposium on Security and Privacy, Oakland, California, May 4-6, 1992. – P. 122-136.