— возможность изменения функциональности субъекта при реализации информационного потока по памяти на функционально ассоциированные с ним сущности;
— необходимость в ряде случаев определения различных правил управления доступом и информационными потоками для распределенных компонент КС.

В связи с этим при наличии возможности при преподавании дисциплины «Теоретические основы компьютерной безопасности» целесообразно рассмотреть подходы, примененные при разработке семейства моделей безопасности логического управления доступом и информационными потоками (ДП-моделей) КС с дискреционным, мандатным или ролевым управлением доступом [3]. При этом наиболее существенными моделями семейства являются дискреционная базовая ДП-модель, ДП-модель с функционально ассоциированными с субъектами сущностями (ФАС ДП-модель), мандатная ДП-модель и базовая ролевая ДП-модель.

Все рассмотренные модели войдут в разрабатываемое автором учебное пособие «Модели безопасности компьютерных систем», второе издание которого планируется в 2010 году.

ЛИТЕРАТУРА

1. *Девянин П. Н.* Модели безопасности компьютерных систем: Учеб. пособие для студ. высш. учеб. заведений. М.: Академия, 2005. 144 с.
2. *Bishop M.* Computer Security: art and science. ISBN 0-201-44099-7, 2002. 1084 p.
3. *Девянин П. Н.* Анализ безопасности управления доступом и информационными потоками в компьютерных системах. М.: Радио и связь, 2006. 176 с.

УДК 004.94

ЗАМЫКАНИЕ БАЗОВОЙ РОЛЕВОЙ ДП-МОДЕЛИ

М. А. Качанов

При анализе базовой ролевой (сокращенно: БР) ДП-модели в работе [1] были рассмотрены условия передачи прав доступа ролей в случае взаимодействия только двух субъект-сессий двух пользователей. В рамках обозначений и определений, введенных в указанной работе, рассмотрим случай взаимодействия субъект-сессий произвольного числа пользователей и предложим алгоритм, используя который можно осуществлять проверку истинности предиката $can\_share()$ для всех пользователей, сущностей и прав доступа одновременно, а вместе с ней и возможности получения недоверенным субъектом права доступа владения к доверенному субъекту в компьютерных системах с дискреционным управлением доступом и информационными потоками. Такой алгоритм реализует преобразование начального состояния компьютерной системы (КС) в его замыкание. Дадим определения замыканий базовой ролевой ДП-модели, предложим и обоснуем алгоритмы их построения.

В соответствии с [1], в рамках БР ДП-модели при анализе условий передачи прав доступа, реализации информационных потоков по памяти или по времени возможно использование только монотонных правил преобразования состояний.

Введем подмножество правил преобразования БР ДП-модели $OP_{acs} = \{take\_role(), grant\_right(), create\_first\_session(), control(), access\_own(), take\_access\_own(), access\_write(), access\_append(), post()\}$. Этих правил преобразования, согласно [1], достаточно для анализа условий передачи прав доступа ролей.

Множество информационных потоков F в состоянии БР ДП-модели будем характеризовать лишь множеством информационных потоков по памяти $F_m \subseteq F$. Этого достаточно для проверки истинности указанного предиката.

При создании недоверенным пользователем $u \in N_u$ субъект-сессии множество функционально ассоциированных с этой сессией сущностей определяется только в зависимости от пользователя и сущности, из которой эта субъект-сессия создана. Созданные субъект-сессии различаются лишь функционально ассоциированными сущностями, а также ролью, которая получает право доступа владения к субъект-сессии, поэтому для анализа передачи прав доступа ролей и проверки истинности предиката $can\_share()$ достаточно рассмотреть по одной субъект-сессии, созданной каждым недоверенным пользователем в следующих предположениях.

Предположение 1. Будем считать сущностями, функционально ассоциированными с субъект-сессией, созданной из сущности $y \in E$ недоверенным пользователем $u \in N_u$, такие сущности, функционально ассоциированные с сущностями z, для которых $(z, execute_r) \in PA(UA(u))$ в данном или некотором последующем состоянии КС $\Sigma(G^\star, OP)$.

Предположение 2. После создания недоверенным пользователем $u \in N_u$ с помощью правила $create\_first\_session()$ субъект-сессии z из сущности $y \in E$ с использованием роли $r \in can\_manage\_rights(AUA(u))$ к множеству $PA(t)$ для каждого $t \in can\_manage\_rights(AUA(u))$ добавляется в качестве элемента пара (z, own_r).

Определение 1. Состояние $G' = (PA', user', roles', A', F'_m, H_{E'})$ системы $\Sigma(G^\star, OP, G_0)$ назовем *role*-замыканием, если оно получено из $G_0 = (PA_0, user_0, roles_0, A_0, F_{m0}, H_{E_0})$ применением последовательности правил $create\_first\_session()$ и $take\_role()$ и выполнены следующие условия:

1) $\forall u \in N_u \; \exists! s \in S' (user'(s) = u)$;
2) $\forall x \in S' \; (roles'(x) = UA(user'(x)) \cup AUA(user'(x)))$.

Алгоритм 1. Построение *role*-замыкания системы $\Sigma(G^\star, OP, G_0)$ для $G_0 = (PA_0, user_0, roles_0, A_0, F_{m0}, H_{E_0})$

Для всех $u \in N_u$
  Взять произвольно $r \in can\_manage\_rights(AUA(u))$ и $y \in E$, такие, что $(y, execute_r) \in PA_0(UA(u))$
  Выполнить $create\_first\_session(u, r, y, z)$
  Для всех $y \in E$
    Если $(y, execute_r) \in PA_0(UA(u))$, **то**
      $[z] = [z] \cup fa(u, y)$
  Для всех $t \in can\_manage\_rights(AUA(u))$
    $PA_0(t) \leftarrow (z, own_r)$
Для всех $x \in S$
  Для всех $r \in UA(user_0(x)) \cup AUA(user_0(x))$
    Выполнить $take\_role(x, r)$

Данный алгоритм строит *role*-замыкание непосредственно по определению в рамках предположений 1, 2.

Определение 2. Состояние $G_{acs} = (PA_{acs}, user_{acs}, roles_{acs}, A_{acs}, F_{macs}, H_{E_{acs}})$ системы $\Sigma(G^\star, OP, G_0)$ назовем *access*-замыканием, если оно получено из *role*-замыкания применением последовательности правил из $OP_{acs} \setminus \{create\_first\_session(), take\_role()\}$ и дальнейшее применение указанных правил не приводит к изменению состояния системы.

Алгоритм 2. Построение *access*-замыкания $G_{acs} = (PA_{acs}, user_{acs}, roles_{acs}, A_{acs}, F_{macs}, H_{E_{acs}})$ системы $\Sigma(G^\star, OP, G_0)$ для $G_0 = (PA, user, roles, A, F_m, H_E)$

1: Выполнить алгоритм 1. Получить *role*-замыкание $G = (PA, user, roles, A, F_m, H_E)$ системы $\Sigma(G^\star, OP, G_0)$
2: $G_{acs} = G$, список A_{own} пуст
3: Выбрать произвольно $k, l \in S, k \neq l$, для которых $\exists r \in roles(k)\ [(l, own_r) \in PA(r)]$. Если таких k и l нет, то перейти на п. 34
4: $A_{acs} = A_{acs} \cup (k, l, own_a)$, добавить (k, l, own_a) в конец списка A_{own}
5: Выбрать очередной непомеченный элемент (x, y, own_a) из списка A_{own}
6: **Для всех** $r \in roles_{acs}(x)$
7: **Для всех** $e \in E$
8: **Если** $((e, own_r), r) \in PA_{acs}(r) \times can\_manage\_rights(roles_{acs}(x) \cap AR) \vee ((e, own_r), r) \in (\bigcup_{\tau \in roles_{acs}(y)} PA_{acs}(\tau)) \times can\_manage\_rights(roles_{acs}(y) \cap AR)$, **то**
9: **Для всех** $(e, \alpha_r) \in P_{acs}$
10: $PA_{acs}(r) = PA_{acs}(r) \cup (e, \alpha_r)$
11: **Если** $\alpha_r = execute_r$, **то**
12: $[x] = [x] \cup fa(user_{acs}(x), e)$
13: **Для всех** $e \in E$
14: **Если** $\exists r \in roles_{acs}(x) \cup roles_{acs}(y)\ [(e, write_r) \in PA_{acs}(r)]$, **то**
15: $A_{acs} = A_{acs} \cup \{(x, e, write_a)\}$, $F_{macs} = F_{macs} \cup \{(x, y, write_m)\}$
16: **Если** $\exists r \in roles_{acs}(x) \cup roles_{acs}(y)\ [(e, append_r) \in PA_{acs}(r)]$, **то**
17: $A_{acs} = A_{acs} \cup \{(x, e, append_a)\}$, $F_{macs} = F_{macs} \cup \{(x, y, write_m)\}$
18: Замкнуть множество A_{acs} по транзитивности отношения владения с помощью правила $take\_access\_own()$
19: Замкнуть список A_{own} по транзитивности отношения владения с помощью правила $take\_access\_own()$, добавляя результаты замыкания в конец списка
20: **Для всех** $z \in S, z \neq x$,
21: **Для всех** $e \in E$
22: **Если** $\exists r \in roles_{acs}(z)\ [(e, read_r) \in PA_{acs}(r)] \vee \exists (z, k, own_a) \in A_{acs}\ [\exists r' \in roles_{acs}(k) \wedge (e, read_r) \in PA_{acs}(r')]$, **то**
23: **Если** $\exists r \in roles_{acs}(x) \cup roles_{acs}(y)\ [(e, write_r) \in PA_{acs}(r) \vee (e, append_r) \in PA_{acs}(r)] \vee (x, e, write_m) \in F_{macs}$, **то**
24: $F_{macs} = F_{macs} \cup \{(x, z, write_m)\}$
25: **Для всех** $\eta \in S, \eta \neq x$,
26: **Для всех** $e \in [\eta]$
27: **Если** $x = e \vee (x, e, write_m) \in F_{macs}$, **то**
28: $A_{acs} = A_{acs} \cup \{(x, \eta, own_a)\}$, добавить (x, η, own_a) в конец A_{own}
29: Замкнуть множество A_{acs} по транзитивности отношения владения с помощью правила $take\_access\_own()$
30: Замкнуть список A_{own} по транзитивности отношения владения с помощью правила $take\_access\_own()$, добавляя результаты замыкания в конец списка
31: Пометить (x, y, own_a) в A_{own} как рассмотренный
32: **Если** в списке A_{own} есть непомеченные элементы, **то**
33: Перейти на п. 5
34: **Вернуть** $G_{acs} = (PA_{acs}, user_{acs}, roles_{acs}, A_{acs}, F_{macs}, H_{E_{acs}})$

Поясним работу алгоритма 2. Пусть построено $role$-замыкание системы $\Sigma(G^*, OP, G_0)$. В этом состоянии доступы владения субъект-сессий отсутствуют. Это следует из определения $role$-замыкания и того, что применение правил $grant\_right()$ и $take\_role()$ не приводит к появлению доступов владения субъект-сессий. Новые же права доступа ролей без возникновения доступов владения субъект-сессий появиться не могут. Это следует из определения правила $grant\_right()$ и функции $de\_facto\_actions$. Поэтому на шаге 3 выполняется проверка возможности осуществления доступа владения с помощью правила $access\_own()$. Если этого сделать нельзя, то текущее состояние системы не изменится.

В процессе работы алгоритма формируется список доступов владения субъект-сессий, в котором рассмотренные элементы помечаются. Данные доступы рассматриваются последовательно друг за другом. Алгоритм завершает свою работу, когда в списке не останется непомеченных элементов. Данное условие обязано выполниться в силу конечности множества S и его неизменности в процессе работы алгоритма.

Шаги 6–12 алгоритма соответсвуют выполнению правила $grant\_right()$, причем на шаге 8 выполняется вычисление функции $de\_facto\_actions()$ для рассматриваемых x и y. Шаг 12 соответствует корректировке функционально ассоциированных сущностей в рамках предположения 1. Шаги 13–17 соответствуют применению правил $access\_write$ и $access\_append$, шаги 20–24 — правила $post()$, шаги 25–28 — правила $control()$. Правила преобразований рассматриваются в порядке влияния применения одного правила на возможные аргументы других, причем после шага 30 никакое применение правила из $OP_{acs} \setminus \{create\_first\_session(), take\_role()\}$ с учетом только доступа владения (x, y, own_a) не изменит текущего состояния системы.

ЛИТЕРАТУРА

1. *Девянин П. Н.* Базовая ролевая ДП-модель // Прикладная дискретная математика. 2008. № 1 (1). С. 64–70.

УДК 004.94

ПОДХОДЫ К ОБЕСПЕЧЕНИЮ БЕЗОПАСНОСТИ КОМПЬЮТЕРНЫХ СИСТЕМ С ФУНКЦИОНАЛЬНО ИЛИ ПАРАМЕТРИЧЕСКИ АССОЦИИРОВАННЫМИ СУЩНОСТЯМИ

Д. Н. Колегов

В настоящее время одной из актуальных задач теории компьютерной безопасности является разработка математических моделей безопасности современных компьютерных систем (КС), реализующих управление доступом и информационными потоками. Данная задача возникает как при теоретическом анализе безопасности КС с применением их формальных моделей, так и при тестировании механизмов защиты КС с использованием процедур, методов и средств автоматизации и компьютерного моделирования. На необходимость формализации процедур оценки безопасности, разработки общих методологий и моделей угроз безопасности КС указывается в «Концепции оценки соответствия автоматизированных систем требованиям безопасности информации» [1]. Более того, в соответствии с «Критериями оценки безопасности информационных технологий» [2] для КС с высоким уровнем доверия обязательным является разработка формальной модели их политики безопасности управления доступом и информационными потоками.