МАТЕМАТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

DOI 10.17223/20710410/15/6

УДК 004.94

РОЛЕВАЯ ДП-МОДЕЛЬ УПРАВЛЕНИЯ ДОСТУПОМ И ИНФОРМАЦИОННЫМИ ПОТОКАМИ В ОПЕРАЦИОННЫХ СИСТЕМАХ СЕМЕЙСТВА *Linux*

П. Н. Девянин

Институт криптографии, связи и информатики, г. Москва, Россия

E-mail: peter_devyanin@hotmail.com

Базовая ролевая ДП-модель управления доступом и информационными потоками в операционных системах (БРОС ДП-модель) достраивается до ролевой ДП-модели управления доступом и информационными потоками в операционных системах (ОС) семейства *Linux* (РОСЛ ДП-модели). Описываются ее новые элементы: имена сущностей, мандатные атрибуты целостности сущностей-контейнеров, функция де-факто владения. Основное отличие модели — четкое разделение де-юре правил преобразования состояний (требующих реализации в ОС) и де-факто правил (используемых только для анализа условий нарушения безопасности системы). Рассматриваются ограничения, инвариантные относительно немонотонных правил преобразования состояний. С учетом внесенных изменений обосновывается утверждение о возможности использования только монотонных правил преобразования состояний для анализа условий передачи прав доступа ролей, получения доступов или реализации информационных потоков.

Ключевые слова: *компьютерная безопасность, ролевая ДП-модель, операционная система Linux.*

1. Введение. Этапы разработки ролевых ДП-моделей

Одним из направлений развития теории компьютерной безопасности является адаптация формальных моделей логического управления доступом и информационными потоками к условиям функционирования реальных компьютерных систем, в особенности ОС. Существующие ОС включают механизмы дискреционного или мандатного управления доступом, сложность реализации которых существенно превосходит возможности аппарата формальных моделей, ориентированных на анализ безопасности этих механизмов. Классические модели, например *Take-Grant* или Белла — ЛаПадулы [1], в основном устарели и не позволяют учитывать свойства современных ОС, существенно влияющие на их безопасность [2]. Таким образом, целесообразна постепенная детализация формальных моделей, уточнение описания их элементов, отработка в рамках моделей техники строгого теоретического обоснования свойств ОС.

Хотя ролевое управление доступом является достаточно сложным механизмом обеспечения безопасности компьютерных систем как с точки зрения научного исследования его свойств, так и с учетом особенностей его практической реализации, оно является одним из самых перспективных, в том числе для применения в ОС. В связи

с этим разработка семейства моделей ролевого управления доступом и информационными потоками (ролевых ДП-моделей) осуществлялась автором поэтапно.

На первом этапе на базе ролевых моделей семейства *RBAC*, дискреционных и мандатных ДП-моделей была построена базовая ролевая ДП-модель (БР ДП-модель) [1], в рамках которой отработана техника обоснования необходимых и достаточных условий передачи прав доступа ролей и реализации информационных потоков по памяти (для случая, когда на траекториях функционирования системы субъект-сессии не получают доступа владения друг к другу с использованием информационных потоков по памяти к функционально ассоциированным с субъект-сессиями сущностям).

На втором этапе с использованием ДП-модели с функционально или параметрически ассоциированными с субъектами сущностями (ФПАС ДП-модели) [3], моделей мандатной политики целостности информации Биба и мандатного ролевого управления доступом [1] была разработана базовая ролевая ДП-модель управления доступом и информационными потоками в ОС (БРОС ДП-модель) [4]. В нее по сравнению с БР ДП-моделью включены некоторые специфичные современным ОС элементы: учетные записи пользователей, сущности, параметрически ассоциированные с субъект-сессиями или ролями («знание» которых — реализация от которых информационных потоков по памяти — позволяет недоверенным субъект-сессиям получить контроль над доверенными субъект-сессиями), де-факто роли, права доступа, возможности и доступы недоверенных субъект-сессий (получаемые ими за счет контроля над доверенными субъект-сессиями), мандатный контроль целостности. Это позволило смоделировать применяемые в ряде современных ОС соответствующие механизмы защиты (например, механизмы *MIC — Mandatory Integrity Control* и *UAC — User Account Control* в ОС семейства *Microsoft Windows*). Основное внимание в БРОС ДП-модели было уделено уточнению условий и результатов применения правил преобразования состояний с учетом включенных в нее новых элементов. Кроме того, так же, как в предыдущих ДП-моделях, была обоснована возможность использования только монотонных правил преобразования состояний при анализе условий передачи прав доступа, реализации информационных потоков по памяти или по времени.

На третьем реализуемом в настоящее время автором этапе исследований формируется ролевая ДП-модель управления доступом и информационными потоками в ОС семейства *Linux* (РОСЛ ДП-модель), основные элементы которой описаны в [5]. Эта модель не сформирована окончательно, в нее (по сравнению с БРОС ДП-моделью) вносятся изменения, направленные в первую очередь на постепенную адаптацию модели к условиям функционирования реальных ОС, на создание предпосылок для разработки механизма управления доступом в защищенных ОС рассматриваемого семейства на основе формальной модели. Рассмотрим эти изменения подробнее.

2. Состояние системы

При описании состояния системы в рамках РОСЛ ДП-модели скорректированы предположения 1 и 5 БРОС ДП-модели (см. [4]), в которых вместо информационных потоков по памяти к сущностям рассматриваются доступы на чтение к ним субъект-сессий. Это связано с тем, что в реальных ОС возникновение информационных потоков является «де-факто следствием» «де-юре получения» субъект-сессиями доступов к сущностям. Кроме того, дополнительно определены следующие функции:

1) $shared\_container : C \to \{\texttt{true}, \texttt{false}\}$ — функция разделяемых контейнеров: сущность-контейнер $c \in C \setminus S$ является разделяемой, когда $shared\_container(c) = \texttt{true}$, в противном случае $shared\_container(c) = \texttt{false}$;

2) $entity\_name : (C \setminus S) \times (E \setminus S) \to NAMES$ — функция имен сущностей (не являющихся субъект-сессиями) в составе сущностей-контейнеров (также не являющихся субъект-сессиями), где $NAMES$ — некоторое множество допустимых имен сущностей, включающее элемент «» — пустая строка. При этом по определению если $e \notin H_E(c)$, то $entity\_names(c, e) =$ «», иначе $entity\_names(c, e) \neq$ «»;

3) $CCRI : E \setminus S \to \{\texttt{true}, \texttt{false}\}$ — функция, задающая способ доступа к сущностям, не являющимся субъектами, внутри контейнеров (с учетом их меток мандатных уровней целостности). Если сущность $e \in C$ является контейнером и доступ к сущностям, содержащимся внутри контейнера e, разрешен без учета уровня целостности контейнера e, то по определению выполняется равенство $CCRI(e) = \texttt{false}$, в противном случае выполняется равенство $CCRI(e) = \texttt{true}$. При этом по определению для каждой сущности $e \in E$, являющейся объектом, выполняется условие $CCRI(e) = \texttt{false}$;

4) $execute\_container : S \times E \to \{\texttt{true}, \texttt{false}\}$ — функция доступа субъект-сессии к сущностям в контейнерах, такая, что по определению для субъект-сессии $s \in S$ и сущности $e \in E$ справедливо равенство $execute\_container(s, e) = \texttt{true}$ тогда и только тогда, когда либо $e \in S$, либо $e \in E \setminus S$ и существует последовательность сущностей $e_1, \ldots, e_n \in E$, где $n \geqslant 1$, $e = e_n$, удовлетворяющих следующим условиям:

— не существует сущности-контейнера $e_0 \in E \setminus S$, такой, что $e_1 \in H_E(e_0)$;
— $e_i \in H_E(e_{i-1})$, где $1 < i \leqslant n$;
— $(e_i, execute_r) \in PA(roles(s))$ и либо $i_e(e_i) \leqslant i_s(s)$, либо $CCRI(e_i) = \texttt{false}$, где $1 \leqslant i < n$.

Таким образом, для сущностей-контейнеров задана функция разделяемых контейнеров (помечаемых в ОС атрибутом «t»), а для всех сущностей, не являющихся субъект-сессиями, определены их имена в составе сущностей-контейнеров. По аналогии с моделью систем военных сообщений [1] добавлен также мандатный атрибут целостности сущностей-контейнеров — $CCRI$, а с использованием функции $execute\_container$ заданы условия доступа субъект-сессии к сущности с учетом ее текущих прав доступа ко всем сущностям-контейнерам, в которые входит данная сущность.

С учетом дополнительных элементов модели в предположении 6 (см. [4]) уточнено, что в начальном состоянии G_0 любой системы $\Sigma(G^*, OP, G_0)$ выполняются следующие условия:

— функции UA_0, PA_0 и $roles_0$ удовлетворяют соответствующим ограничениям $Constraint_U$, $Constraint_P$, $Constraint_S$;
— для любых субъект-сессии $s \in S_0$ и сущности $e \in E_0$ если $(s, e, \alpha_a) \in A_0$ (т. е. доступ уже имеется в начальном состоянии системы), где $\alpha_a \in R_a$, то справедливо равенство $execute\_container(s, e) = \texttt{true}$ (этот доступ мог быть получен субъект-сессией s только с учетом ее текущих прав доступа ко всем сущностям-контейнерам, в которые входит сущность e).

Предположения 7 и 9 дополнены и частично переформулированы.

Предположение 7. Уровень целостности роли не превосходит уровней целостности ролей, которым она подчинена в иерархии ролей. Уровень целостности сущности, входящей в состав сущности-контейнера и не являющейся субъект-сессией, не превосходит уровня целостности сущности-контейнера. Уровни целостности сущностей, параметрически ассоциированных с учетной записью пользователя, совпадают с ее уровнем целостности. Текущий уровень целостности субъект-сессии не превосходит уровня целостности учетной записи пользователя, от имени которой она функционирует, и текущего уровня субъект-сессии, которой она подчинена в иерархии. При этом

в дополнение к предположению 1 недоверенная субъект-сессия всегда может создать недоверенную субъект-сессию с низким уровнем целостности. Уровень целостности роли не может быть выше уровня целостности учетной записи пользователя, которая на нее может быть авторизована, и текущего уровня целостности субъект-сессии, во множество текущих ролей которой она входит. Права доступа владения own_r и на запись $write_r$ к сущности, не являющейся субъект-сессией, могут принадлежать только ролям, имеющим уровень целостности не ниже, чем уровень целостности сущности. Право доступа владения own_r к субъект-сессии может принадлежать только ролям, имеющим уровень целостности не ниже, чем уровень целостности субъект-сессии. Таким образом, выполняются следующие условия:

1. Для ролей $r, r' \in R \cup AR$, если $r \leqslant r'$, то $i_r(r) \leqslant i_r(r')$.
2. Для сущностей $e, e' \in E \setminus S$, если $e \leqslant e'$, то $i_e(e) \leqslant i_e(e')$.
3. Для субъект-сессий $s, s' \in S$, если $s \leqslant s'$, то $i_s(s) \leqslant i_s(s')$.
4. Для каждой сущности $e \in]u[$, где $u \in U$, справедливо равенство $i_e(e) = i_u(u)$.
5. Для субъект-сессии $s \in S$ верно неравенство $i_s(s) \leqslant i_u(user(s))$.
6. Для учетной записи пользователя $u \in U$ и роли $r \in R$, если $r \in UA(u)$, то $i_r(r) \leqslant i_u(u)$.
7. Для субъект-сессии $s \in S$ и роли $r \in R$, если $r \in roles(s)$, то $i_r(r) \leqslant i_s(s)$.
8. Для права доступа к сущности $(e, \alpha) \in P$, где $\alpha \in \{own_r, write_r\}$, и роли $r \in R$, если $(e, \alpha) \in PA(r)$, то или $e \in E \setminus S$ и $i_e(e) \leqslant i_r(r)$, или $e \in S$, $\alpha = own_r$ и $i_s(e) \leqslant i_r(r)$.

Предположение 9. Субъект-сессии могут иметь друг к другу только доступ владения own_a. Роли могут обладать к субъект-сессиям только правом доступа владения own_r. К сущностям, не являющимся субъект-сессиями, субъект-сессии могут иметь любые виды доступа из множества R_a, а роли могут иметь к таким сущностям любые права доступа из множества R_r. Для управления доступом к сущности субъект-сессия должна получить к ней доступ владения. Для создания, переименования или удаления сущности или «жесткой» ссылки на нее в сущности-контейнере субъект-сессии необходимо иметь к сущности-контейнеру доступ на запись. Для переименования, удаления сущности или «жесткой» ссылки на сущность e в сущности-контейнере c ($e \in H_E(c)$), помеченной как разделяемая ($shared\_container(c) = \texttt{true}$), требуется наличие у субъект-сессии текущей роли, обладающей правом доступа владения own_r к сущности e. При этом для осуществления субъект-сессией любых действий над сущностью (управление к ней доступом, получения доступа, создания, удаления и др.), не являющейся субъект-сессией, требуется существование последовательности непосредственно вложенных друг в друга сущностей-контейнеров, начинающейся с некоторой сущности-«корневой контейнер» (например, корневой контейнер «/» в ОС семейства *Linux*) и заканчивающейся сущностью-контейнером, в состав которой непосредственно входит сама сущность, и наличие у субъект-сессии:

— текущих ролей, обладающих в совокупности правами доступа $execute_r$ ко всем сущностям-контейнерам этой последовательности;
— возможности получения доступа (с учетом уровня целостности субъект-сессии) внутрь всех сущностей-контейнеров этой последовательности с учетом их уровней целостности и мандатных атрибутов целостности $CCRI$.

Таким образом, с учетом предположения 5 всегда справедливо:

— для субъект-сессий s, $s' \in S$, если $(s, s', \alpha_a) \in A$, то $\alpha_a = own_a$;
— для роли $r \in R$, если $(s, \alpha_r) \in PA(r)$, где $s \in S$, то $\alpha_r = own_r$;

— для субъект-сессии $s \in S$ и сущности $e \in E$, если $(s, e, \alpha_a) \in A$, то $execute\_container(s, e) = \texttt{true}$.

Ранее при разработке ДП-моделей для отражения ситуации, когда одна субъект-сессия получает контроль над другой субъект-сессий, использовалось право доступа владения own_a. В то же время в реальных ОС такой доступ не всегда может быть явно задан как ее параметр. Например, когда субъект-сессия осуществляет отладку другой субъект-сессии, можно считать, что этот доступ задан явно, а когда первая субъект-сессия через переполнение буфера памяти (реализацию информационного потока по памяти к сущности, функционально ассоциированной с другой субъект-сессий) получила над ней контроль, доступ владения явно не предоставляется. В связи с этим используем обозначение и следующие формулировки предположений 10 и 11: $de\_facto\_own : S \to S$ — функция де-факто владения субъект-сессиями, т. е. по определению будем говорить, что субъект-сессия s де-факто владеет субъект-сессией s', когда выполняется условие $s' \in de\_facto\_own(s)$. При этом по определению всегда $s \in de\_facto\_own(s)$ (субъект-сессия де-факто владеет сама собой) и если $(s, s', own_a) \in A$, то $s' \in de\_facto\_own(s)$ (если есть де-юре владение, то есть и де-факто владение).

Предположение 10. Если субъект-сессия s реализовала информационный поток по памяти от себя к сущности, функционально ассоциированной с другой субъект-сессией s', или субъект-сессия s реализовала информационный поток по памяти к себе от всех сущностей, параметрически ассоциированных с другой субъект-сессией s', то субъект-сессия s получает де-факто владение субъект-сессией s' ($s' \in de\_facto\_own(s)$).

Предположение 11. Если субъект-сессия s де-факто владеет субъект-сессией s', то субъект-сессия s получает следующие возможности:

— использовать роли из множества текущих ролей субъект-сессии s';
— изменять множество текущих ролей субъект-сессии s';
— использовать текущий уровень целостности субъект-сессии s';
— использовать доступы субъект-сессии s';
— получать де-факто владение субъект-сессиями, которыми де-факто владеет субъект-сессия s';
— использовать административные роли субъект-сессии s' для осуществления действий над ролями и сущностями, которые позволяют ей изменять права доступа ролей субъект-сессии s';
— использовать информационные потоки, в реализации которых участвует субъект-сессия s';
— удалить субъект-сессию s'.

При этом используем обозначения:

1) $de\_facto\_roles : S \to 2^{R \cup AR}$ — функция де-факто текущих ролей субъект-сессий, при этом по определению в каждом состоянии системы G для каждой субъект-сессии $s \in S$ верно равенство $de\_facto\_roles(s) = \{r \in R \cup AR$: существует $s' \in S$, такая, что $s' \in de\_facto\_own(s)$ и $r \in roles(s')\}$;

2) $de\_facto\_rights : S \to 2^P$ — функция де-факто текущих прав доступа субъект-сессий, при этом по определению в каждом состоянии системы G для каждой субъект-сессии $s \in S$ верно равенство $de\_facto\_rights(s) = \{p \in P$: существует $r \in de\_facto\_roles(s)$, такая, что $p \in PA(r)\}$;

3) $de\_facto\_accesses : S \to 2^A$ — функция де-факто доступов субъект-сессий, при этом по определению в каждом состоянии системы G для каждой субъект-сессии $s \in S$ верно равенство $de\_facto\_accesses(s) = \{(s', e, \alpha_a) : s' \in de\_facto\_own(s), (s', e, \alpha_a) \in A\}$.

Заметим, что функция де-факто возможностей субъект-сессий ($de\_facto\_actions$) более в рамках РОСЛ ДП-модели не рассматривается.

3. Правила преобразования состояний

В рамках РОСЛ ДП-модели используются следующие правила преобразования состояний из множества OP (табл. 1 и 2), которые либо добавлены впервые, либо существенно отличаются от правил БРОС ДП-модели. По аналогии с моделью *Take-Grant* их можно неформально классифицировать на де-юре правила (правила, которые требуют реализации в ОС, то есть приводящие к «реальным» изменениям ее параметров: изменению множеств текущих ролей, прав доступа ролей, получению доступов субъект-сессий к сущностям и т. д.) и де-факто правила (правила, которые не требуют реализации в ОС, так как используются в модели для отражения факта получения субъект-сессией де-факто владения субъект-сессиями или факта реализации информационного потока по памяти или по времени).

Таким образом, в рамках РОСЛ ДП-модели заданы 18 де-юре и 11 де-факто правил преобразования состояний. Проанализируем их наиболее существенные отличия от аналогичных правил БРОС ДП-модели.

Во всех де-юре правилах, где ранее требовалось выполнение «де-факто условия» — реализация информационного потока по памяти либо от некоторой субъект-сессии к сущности i_entity, либо от сущности e, параметрически ассоциированной с ролью или учетной записью пользователя, к некоторой субъект-сессии, — теперь требуется выполнение «де-юре условия» — наличие у этой субъект-сессии доступа соответственно либо на запись к сущности i_entity, либо на чтение к сущности e. Фактические возможности также заменены на соответствующие доступы — владения, на запись или на чтение.

Правило создания сущности заменено на три правила — создания сущности-объекта, сущности-контейнера и «жесткой» ссылки на сущность. В эти правила, а также в правило переименования сущности добавлен дополнительный параметр — имя сущности ($name$). Кроме того, в правило создания сущности-контейнера $create\_container$ включены параметры t (указывающий, является ли сущность-контейнер разделяемым) и $ccri$ (мандатный атрибут целостности). В связи с этим в модель добавлено новое правило $set\_container\_attr$, позволяющее субъект-сессиям изменять параметры t и $ccri$ сущностей-контейнеров.

В условиях выполнения правил получения доступов владения, на чтение или на запись к сущности добавлена проверка прав доступа к сущностям-контейнерам, содержащим эту сущность, что отражает порядок предоставления аналогичных доступов в реальных ОС семейства *Linux*.

Де-факто правило $de\_facto\_op(x, op(y, y', \ldots))$ не имеет аналогов в других ДП-моделях и позволяет субъект-сессии x, де-факто владеющей субъект-сессиями y и y', выполнить от имени y любое де-юре правило. Данное правило позволило исключить из рассмотрения в рамках РОСЛ ДП-модели де-факто возможности субъект-сессий и более ясно описать условия применения де-юре правил преобразования состояний.

Таблица 1

Де-юре правила преобразования состояний РОСЛ ДП-модели

Правило	Исходное состояние $G = (PA, user, roles, A, F, H_E)$	Результирующее состояние $G' = (PA', user', roles', A', F', H'_E)$
1	2	3
$take\_role(x, x', \{r_j : 1 \leqslant j \leqslant k\})$	$x, x' \in S$, $r_j \in UA(user(x)) \cup \cup AUA(user(x))$, $\{(x, e, read_a) : e \in]r_j[\} \subset A$, $i_r(r_j) \leqslant i_s(x)$, $Constraint_S(roles') =$ `true`, [если $i_r(r_j) = i\_high$, то $(x', i\_entity, write_a) \in A$], где $1 \leqslant j \leqslant k$	$S' = S$, $E' = E$, $PA' = PA$, $user' = user$, $A' = A$, $H'_E = H_E$, $roles'(x) = roles(x) \cup \{r_j : 1 \leqslant j \leqslant k\}$ и для $s \in S \setminus \setminus \{x\}$ выполняется равенство $roles'(s) = roles(s)$, если $x \in (N_S \cup \cup NF_S) \cap S$ и $\{r_j : 1 \leqslant j \leqslant k\} \setminus roles(x) \neq \varnothing$, то $F' = F \cup \{(x, s, write_t) : s \in (N_S \cup NF_S) \cap S$, $x \neq s$ и $x \in de\_facto\_own(s)\}$, иначе $F' = F$
$remove\_role(x, x', \{r_j : 1 \leqslant j \leqslant k\})$	$x, x' \in S$, $r_j \in roles(x)$, $\{(x, e, read_a) : e \in]r_j[\} \subset A$, $Constraint_S(roles') =$ `true`, [если $i_r(r_j) = i\_high$, то $(x', i\_entity, write_a) \in A$], где $1 \leqslant j \leqslant k$	$S' = S$, $E' = E$, $PA' = PA$, $user' = user$, $A' = A$, $H'_E = H_E$, $roles'(x) = roles(x) \setminus \{r_j : 1 \leqslant j \leqslant k\}$ и для $s \in S \setminus \{x\}$ выполняется равенство $roles'(s) = roles(s)$, если $x \in (N_S \cup NF_S) \cap S$, то $F' = F \cup \cup\{(x, s, write_t)$: $s \in (N_S \cup NF_S)$, $x \neq s$ и $x \in de\_facto\_own(s)\}$, иначе $F' = F$
$grant\_right(x, x', r, \{(y_j, \alpha_{r_j}) : 1 \leqslant j \leqslant k\})$	$x, x' \in S$, $y_j \in E$, $r \in can\_manage\_rights(roles(x) \cap AR)$, $(x, y_j, own_a) \in A$, $i_r(r) \leqslant i_s(x)$, [если $y_j \in S$, то $\alpha_{r_j} = own_r$ и $i_s(y_j) \leqslant i_r(r)$], [если $y_j \in E \setminus S$ и $\alpha_{r_j} \in \{own_r, write_r\}$, то $i_e(y_j) \leqslant i_r(r)$], [$Constraint_P(PA') = true$], [если $i_e(y_j) = i\_high$, то $(x', i\_entity, write_a) \in A$], где $1 \leqslant j \leqslant k$	$S' = S$, $E' = E$, $user' = user$, $roles' = roles$, $A' = A$, $H'_E = H_E$, $PA'(r) = PA(r) \cup \{(y_j, \alpha_{r_j}) : 1 \leqslant j \leqslant k\}$ и для $r' \in R \setminus \{r\}$ выполняется равенство $PA'(r') = PA(r')$, если $x \in (N_S \cup NF_S) \cap S$, то $F' = F \cup \cup\{(x, s, write_t)$: $s \in (N_S \cup NF_S) \cap S$, $x \neq s$, $r \in de\_facto\_roles(s)$, и существует j, такое, что $1 \leqslant j \leqslant k$, $(y_j, \alpha_{r_j}) \notin PA(r)$ и $(y_j, \alpha_a) \in de\_facto\_accesses(s)$, где $\alpha_a \in R_a\}$, иначе $F' = F$
$remove\_right(x, x', r, \{(y_j, \alpha_{r_j}) : 1 \leqslant j \leqslant k\})$	$x, x' \in S$, $y_j \in E$, $(y_j, \alpha_{r_j}) \in PA(r)$, $r \in can\_manage\_rights(roles(x) \cap AR)$, $(x, y_j, own_a) \in A$, $i_r(r) \leqslant i_s(x)$, $Constraint_P(PA') =$ `true`, [если $i_e(y_j) = i\_high$, то $(x', i\_entity, write_a) \in A$], где $1 \leqslant j \leqslant k$	$S' = S$, $E' = E$, $user' = user$, $roles' = roles$, $A' = A$, $H'_E = H_E$, $PA'(r) = PA(r) \setminus \{(y_j, \alpha_{r_j}) : 1 \leqslant j \leqslant k\}$ и для $r' \in R \setminus \{r\}$ выполняется равенство $PA'(r') = PA(r')$, если $x \in (N_S \cup NF_S) \cap S$, то $F' = F \cup \cup\{(x, s, write_t) : s \in (N_S \cup NF_S) \cap S$, $x \neq s$, $r \in de\_facto\_roles(s)$, и существует j, такое, что $1 \leqslant j \leqslant k$ и $(y_j, \alpha_a) \in de\_facto\_accesses(s)$, где $\alpha_a \in R_a\}$, иначе $F' = F$
$create\_object(x, x', r, y, yi, name, z)$	$x, x' \in S$, $y \notin E$, $z \in C \setminus S$, $name \in NAME \setminus \{«»\}$, $r \in \in can\_manage\_rights(roles(x) \cap AR)$, $(x, z, write_a) \in A$, $yi \leqslant \leqslant i_r(r) \leqslant i_s(x)$, $yi \leqslant i_e(z)$, $Constraint_P(PA') =$ `true`, [если $i_e(z) = i\_high$, то $(x', i\_entity, write_a) \in A$]	$S' = S$, $E' = E \cup \{y\}$ ($O' = O \cup \{y\}$, $C' = C$), при этом $y \notin UE \cup RE$, $user' = user$, $roles' = = roles$, $A' = A$, $i'_e(y) = yi$, $CCRI'(y) =$ `false`, $entity\_name(z, y) = name$, $PA'(r) = PA(r) \cup \{(y, ownr)\}$ и для $r' \in R \setminus \{r\}$ выполняется равенство $PA'(r') = PA(r')$, $H'_E(z) = H_E(z) \cup \{y\}$, $H'_E(y) = \varnothing$, для $e \in \in E \setminus \{z\}$ выполняется равенство $H'_E(e) = = H_E(e)$, если $x \in (N_S \cup NF_S) \cap S$, то $F' = F \cup \{(x, e, write_t) : e \in E$ и $y \leqslant e\} \cup \cup\{(x, s, write_t) : s \in (N_S \cup NF_S) \cap S, x \neq \neq s, r \in de\_facto\_roles(s)$ и $(z, \alpha_a) \in de\_facto\_accesses(s)$, где $\alpha_a \in R_a\}$, иначе $F' = F$

Продолжение табл. 1

1	2	3
create_container $(x,\ x',\ r,\ y,\ yi,$ $ccri,\ t,\ name,\ z)$	$x, x' \in S$, $y \notin E$, $z \in C \setminus S$, $ccri, t \in \{\texttt{true}, \texttt{false}\}$, $name \in NAME \setminus \{«»\}$, $r \in can\_manage\_rights(roles(x) \cap AR)$, $(x, z, write_a) \in A$, $yi \leqslant i_r(r) \leqslant i_s(x)$, $yi \leqslant i_e(z)$, $Constraint_P(PA') = \texttt{true}$, [если $i_e(z) = i\_high$, то $(x', i\_entity, write_a) \in A$]	$S' = S$, $E' = E \cup \{y\}$ ($O' = O$, $C' = C \cup \{y\}$), при этом $y \notin UE \cup RE$, $user' = user$, $roles' = roles$, $A' = A$, $i'_e(y) = yi$, $CCRI'(y) = ccri$, $entity\_name(z, y) = name$, $PA'(r) = PA(r) \cup \{(y, ownr)\}$ и для $r' \in R \setminus \{r\}$ выполняется равенство $PA'(r') = PA(r')$, $H'_E(z) = H_E(z) \cup \{y\}$, $H'_E(y) = \varnothing$, для $e \in E \setminus \{z\}$ выполняется равенство $H'_E(e) = H_E(e)$, $shared\_container'(y) = t$, если $x \in (N_S \cup NF_S) \cap S$, то $F' = F \cup \{(x, e, write_t) : e \in E$ и $y \leqslant e\} \cup \{(x, s, write_t) : s \in (N_S \cup \cup NF_S) \cap S$, $x \neq s$, $r \in de\_facto\_roles(s)$ и $(z, \alpha_a) \in de\_facto\_accesses(s)$, где $\alpha_a \in R_a\}$, иначе $F' = F$
create_hard_link$(x, x', y,$ $name, z)$	$x, x' \in S$, $y \in O \setminus S$, $z \in C \setminus S$, $name \in NAME \setminus \{«»\}$, $y \notin UE \cup RE$, $(x, z, write_a) \in A$, $i_e(y) \leqslant i_e(z)$, [если $i_e(z) = i\_high$, то $(x', i\_entity, write_a) \in A$]	$S' = S$, $E' = E$, $PA' = PA$, $user' = user$, $roles' = roles$, $A' = A$, $entity\_name(z, y) = name$, $H'_E(z) = H_E(z) \cup \{y\}$, для $e \in E \setminus \{z\}$ выполняется равенство $H'_E(e) = H_E(e)$, если $x \in (N_S \cup NF_S) \cap S$, то $F' = F \cup \{(x, e, write_t)$: $e \in E$ и $y \leqslant e\}$, иначе $F' = F$
rename_entity $(x, x', y, name, z)$	$x, x' \in S$, $y, z \in E \setminus S$, $y \in H_E(z)$, $name \in NAME \setminus \{«»\}$, $(x, z, write_a) \in A$, [если $i_e(z) = i\_high$, то $(x', i\_entity, write_a) \in A$], [если $shared\_container(z) = \texttt{true}$, то $(y, own_r) \in PA(roles(x))$]	$S' = S$, $E' = E$, $PA' = PA$, $user' = user$, $roles' = roles$, $A' = A$, $H'_E = H_E$, $entity\_name(z, y) = name$, если $x \in (N_S \cup NF_S) \cap S$, то $F' = F \cup \{(x, e, write_t)$: $e \in E$, $x \neq e$, и или $e \leqslant y$, или $e = z\} \cup \{(x, s, write_t)$: $s \in (N_S \cup NF_S) \cap S$, $x \neq s$, $(e, \alpha_a) \in de\_facto\_accesses(s)$, где $e \in E$, $e \leqslant y$, $\alpha_a \in R_a\}$, иначе $F' = F$
set_container_attr$(x, x', y, ccri, t)$	$x, x' \in S$, $y \in C \setminus S$, $ccri, t \in \{\texttt{true}, \texttt{false}\}$, $(x, y, write_a) \in A$, [если $i_e(y) = i\_high$, то $(x', i\_entity, write_a) \in A$]	$S' = S$, $E' = E$, $PA' = PA$, $user' = user$, $roles' = roles$, $A' = A$, $H'_E = H_E$, $CCRI'(y) = ccri$, $shared\_container'(y) = t$, если $x \in (N_S \cup NF_S) \cap S$, то $F' = F \cup \{(x, e, write_t)$: $e \in E$, $x \neq e$ и $e \leqslant y\} \cup \{(x, s, write_t)$: $s \in (N_S \cup NF_S) \cap S$, $x \neq s$, $(e, \alpha_a) \in de\_facto\_accesses(s)$, где $e \in E$, $e \leqslant y$, $\alpha_a \in R_a\}$, иначе $F' = F$
delete_entity$(x,$ $x',\ y,\ z)$	$x, x' \in S$, $y \in E \setminus S$, $z \in C \setminus S$, $y \in H_E(z)$, $H_E(y) = \varnothing$, [не существует $z' \in$, такой, что $z' \neq z$ и $y \in H_E(z')$], [$y \notin (UE \cup RE)$, $(x, z, write_a) \in A$, $Constraint_P(PA') = \texttt{true}$], [если $i_e(z) = i\_high$, то $(x',$ $i\_entity,\ write_a) \in A$], [если $shared\_container(z) = \texttt{true}$, то $(y,\ own_r) \in PA(roles(x))$]	$S' = S$, $E' = E \setminus \{y\}$, $user' = user$, $roles' = roles$, $H'_E(z) = H_E(z) \setminus \{y\}$, для всех $e \in E' \setminus \{z\}$ выполняется равенство $H'_E(e) = H_E(e)$, для $r \in R$ выполняется равенство $PA'(r) = PA(r) \setminus \{(y, \alpha_r)$: $\alpha_r \in R_r\}$, $A' = A \setminus \{(s, y, \alpha_a)$: $s \in S$, $\alpha_a \in R_a\}$, если $x \in (N_S \cup NF_S) \cap S$, то $F' = (F \cup \cup \{(x, e, write_t) : e \in E$, $x \neq e$ и $z \leqslant e\} \cup \cup \{(x, s, write_t) : s \in (N_S \cup NF_S) \cap S'$, $x \neq s$ и $(y, \alpha_a) \in de\_facto\_accesses(s)$, где $\alpha_a \in R_a\}) \setminus (\{(e, y, \alpha_f) : e \in E, \alpha_f \in R_f\} \cup \cup \{(y, e, \alpha_f) : e \in E, \alpha_f \in R_f\})$, иначе $F' = F$

Продолжение табл. 1

1	2	3
$delete\_hard\_link(x, x', y, z)$	$x, x' \in S$, $y \in O \setminus S$, $z \in C \setminus S$, $y \in H_E(z)$, [не существует $z' \in C$, такой, что $z' \neq z$ и $y \in H_E(z')$], [$y \notin (UE \cup RE)$, $(x, z, write_a) \in A$], [если $i_e(z) = i\_high$, то $(x', i\_entity, write_a) \in A$], [если $shared\_container(z) = \texttt{true}$, то $(y, own_r) \in PA(roles(x))$]	$S' = S$, $E' = E$, $PA' = PA$, $user' = user$, $roles' = roles$, $H'_E(z) = H_E(z) \setminus \{y\}$, для всех $e \in E' \setminus \{z\}$ выполняется равенство $H'_E(e) = H_E(e)$, если $x \in (N_S \cup NF_S) \cap S$, то $F' = (F \cup \{(x, e, write_t) : e \in E, x \neq e$ и или $y \leqslant e$, или $z \leqslant e\}$, иначе $F' = F$
$create\_first\_session(x, x', u, r, y, z, zi)$	$x, x' \in S$, $u \in U$, $y \in E$, $z \notin E$, $(y, execute_r) \in PA(roles(x))$, $execute\_container(x, y)=\texttt{true}$, $r \in can\_manage\_rights(roles(x) \cap AR)$, $zi \leqslant i_u(u)$, $zi \leqslant i_r(r)$, $\{(x, e, read_a)$: $e \in]u[\} \subset A$, $Constraint_P(PA') = \texttt{true}$, $Constraint_S(roles') = \texttt{true}$, [если $zi = i\_high$, то $(x', i\_entity, write_a) \in A$]	$S' = S \cup \{z\}$, $E' = E \cup \{z\}$, $A' = A \cup \{(x, z, own_a)\}$, $i'_s(z) = zi$, $user'(z) = u$, для $s \in S$ выполняется равенство $user'(s) = user(s)$, $roles'(z) = \varnothing$, для $s \in S$ выполняется равенство $roles'(s) = roles(s)$, $[z] = fa(u, y)$, $]z[= fp(u, y)$, $PA'(r) = PA(r) \cup \{(z, own_r)\}$ и для $r' \in R \setminus \{r\}$ выполняется равенство $PA'(r') = PA(r')$, $H'_E(z) = \varnothing$, для $e \in E$ выполняется равенство $H'_E(e) = H_E(e)$, если $x \in (NS \cup NF_S) \cap S$, то $F' = F \cup \{(z, x, write_t), (x, z, write_t)\} \cup \{(x, e, write_t) : e \in E$ и $y \leqslant e\} \cup \cup \{(x, s, write_t) : s \in (NS \cup NF_S) \cap S$, $x \neq s$ и $r \in de\_facto\_roles(s)\}$, иначе $F' = F$
$create\_session(x, x', r, y, z, zi)$	$x, x' \in S$, $y \in E$, $z \notin E$, $(y, execute_r) \in PA(roles(x))$, $execute\_container(x, y)=\texttt{true}$, $r \in can\_manage\_rights(roles(x) \cap AR)$, $zi \leqslant i_r(r) \leqslant i_s(x)$, $Constraint_P(PA') = \texttt{true}$, $Constraint_S(roles') = \texttt{true}$, [если $zi = i\_high$, то $(x', i\_entity, write_a) \in A$]	$S' = S \cup \{z\}$, $E' = E \cup \{z\}$, $A' = A \cup \{(x, z, own_a)\}$, $i'_s(z) = zi$, $user'(z) = user(x)$, для $s \in S$ выполняется равенство $user'(s) = user(s)$, $roles'(z) = \varnothing$, для $s \in S$ выполняется равенство $roles'(s) = roles(s)$, $[z] = fa(user(x), y)$, $]z[= fp(user(x), y)$, $H'_E(x) = H_E(x) \cup \{z\}$, $H'_E(z) = \varnothing$, для $e \in E \setminus \{x\}$ выполняется равенство $H'_E(e) = H_E(e)$, $PA'(r) = PA(r) \cup \{(z, own_r)\}$ и для $r' \in R \setminus \{r\}$ выполняется равенство $PA'(r') = PA(r')$, если $x \in (NS \cup NF_S) \cap S$, то $F' = F \cup \{(z, x, write_t), (x, z, write_t)\} \cup \{(x, e, write_t) : e \in E$ и $y \leqslant e\} \cup \{(x, s, write_t) : s \in (NS \cup NF_S) \cap S$, $x \neq s$ и $r \in de\_facto\_roles(s)\}$, иначе $F' = F$
$delete\_session(x, x', z)$	$x, x', z \in S$, $(x, z, own_a) \in A$, $i_s(z) \leqslant i_s(x)$, $Constraint_P(PA') = \texttt{true}$, $Constraint_S(roles') = \texttt{true}$, [если $i_s(z) = i\_high$, то $(x', i\_entity, write_a) \in A$]	$S' = S \setminus \{z\}$, $E' = E \setminus \{z\}$, для $s \in S'$ верно $user'(s) = user(s)$, $roles'(s) = roles(s)$, для $z' \in S$, такой, что $z \in H_E(z')$, верно $H'_E(z') = (H_E(z') \setminus \{z\}) \cup H_E(z)$, при этом выполняется: для $e \in E' \setminus \{z'\}$ верно $H'_E(e) = H_E(e)$, $PA'(r) = PA(r) \setminus \{(z, own_r)\}$, и для $r' \in R \setminus \{r\}$ верно $PA'(r') = PA(r')$, $A' = A \setminus (\{(z, e, \alpha_a) : e \in E, \alpha_a \in R_a\} \cup \{(s, z, own_a) : s \in S\})$, если $x \in (NS \cup NF_S) \cap S$, то $F' = (F \cup \{(x, s, write_t) : e \in E$, $x \neq e$ и $z < e\} \cup \{(x, s, write_t) : s \in (NS \cup NF_S) \cap S$, $x \neq s$ и $z \in de\_facto\_own(s)\}) \setminus (\{(z, e, \alpha_f) : e \in E, \alpha_f \in R_f\} \cup \{(e, z, \alpha_f) : e \in E, \alpha_f \in R_f\})$, иначе $F' = F$

Окончание табл. 1

1	2	3
$access\_own(x, x', y)$	$x, x' \in S$, $y \in E$, $x \neq y$, $(y, own_r) \in PA(roles(x))$, $execute\_container(x, y) =$ `true`, [если $y \in S$, то $i_s(y) \leqslant i_s(x)$], [если $y \in E \setminus S$, то $i_e(y) \leqslant i_s(x)$], [если ($y \in S$ и $i_s(y) = i\_high$) или ($y \in E \setminus S$ и $i_e(y) = i\_high$), то $(x', i\_entity, write_a) \in A$]	$S' = S$, $E' = E$, $PA' = PA$, $user' = user$, $roles' = roles$, $H'_E = H_E$, $A' = A \cup \{(x, y, own_a)\}$, если $x \in (N_S \cup NF_S) \cap S$, то $F' = F \cup \{(x, e, write_t)$: $e \in E$, $x \neq e$ и $y \leqslant e\} \cup \{(x, s, write_t)$: $s \in (N_S \cup NF_S) \cap S$, $x \neq s$ и [если $y \in E \setminus S$, то $(y, own_a) \in de\_facto\_accesses(s)$], [если $y \in S$, то $y \in de\_facto\_own(s)$]$\}$, иначе $F' = F$
$access\_read(x, x', y)$	$x, x' \in S$, $y \in E \setminus S$, $(y, read_r) \in PA(roles(x))$, $execute\_container(x, y) =$ `true`	$S' = S$, $E' = E$, $PA' = PA$, $user' = user$, $roles' = roles$, $H'_E = H_E$, $A' = A \cup \{(x, y, read_a)\}$, если $x \in (N_S \cup NF_S) \cap S$, то $F' = F \cup \{(y, x, write_m)\} \cup \{(x, e, write_t) : e \in E$, $x \neq e$ и $y \leqslant e\}$, иначе $F' = F \cup \{(y, x, write_m)\}$
$access\_write(x, x', y)$	$x, x' \in S$, $y \in E \setminus S$, $(y, write_r) \in PA(roles(x))$, $execute\_container(x, y) =$ `true`, $i_e(y) \leqslant i_s(x)$, [если $i_e(y) = i\_high$, то $(x', i\_entity, write_a) \in A$]	$S' = S$, $E' = E$, $PA' = PA$, $user' = user$, $roles' = roles$, $H'_E = H_E$, $A' = A \cup \{(x, y, write_a)\}$, если $x \in (N_S \cup NF_S) \cap S$, то $F' = F \cup \{(x, y, write_m)\} \cup \{(x, e, write_t)$: $e \in E$, $x \neq e$ и $y \leqslant e\}$, иначе $F' = F \cup \{(x, y, write_m)\}$
$delete\_access(x, x', y, \alpha_a)$	$x, x' \in S$, $y \in E \setminus S$, $(x, y, \alpha_a) \in A$	$S' = S$, $E' = E$, $PA' = PA$, $user' = user$, $roles' = roles$, $H'_E = H_E$, $A' = A \setminus \{(x, y, \alpha_a)\}$, если $x \in (N_S \cup NF_S) \cap S$, то $F' = F \cup \{(x, e, write_t) : e \in E$, $x \neq e$ и $y \leqslant e\}$, иначе $F' = F$

Таблица 2

Де-факто правила преобразования состояний РОСЛ ДП-модели

Правило	Исходное состояние $G = (PA, user, roles, A, F, H_E)$	Результирующее состояние $G' = (PA', user', roles', A', F', H'_E)$
1	2	3
$de\_facto\_op(x, op(y, y', \ldots))$	$x, y, y' \in S$, $y, y' \in de\_facto\_own(x)$, выполняются условия применения де-юре правила преобразования состояний $op(y, y', \ldots)$, заданные в табл. 1	Соответствуют результатам применения правила $op(y, y', \ldots)$
$control(x, y, z)$	$x, y \in S$, $x \neq y$, $z \in [y]$ и или $x = z$, или $(x, z, write_m) \in F$, или $z \in S$ и $z \in de\_facto\_own(x)$	$S' = S$, $E' = E$, $PA' = PA$, $user' = user$, $roles' = roles$, $A' = A$, $H'_E = H_E$, $de\_facto\_own'(x) = de\_facto\_own(x) \cup \{y\}$, если $x \in (N_S \cup NF_S) \cap S$, то $F' = F \cup \{(x, e, write_t)$: $e \in E$, $x \neq e$ и $y \leqslant e\}$, иначе $F' = F$
$know(x, y)$	$x, y \in S$, $x \neq y$, и для каждой $e \in]y[$ существует $(e, x, write_m) \in F$	$S' = S$, $E' = E$, $PA' = PA$, $user' = user$, $roles' = roles$, $A' = A$, $H'_E = H_E$, $de\_facto\_own'(x) = de\_facto\_own(x) \cup \{y\}$, если $x \in (N_S \cup NF_S) \cap S$, то $F' = F \cup \{(x, e, write_t)$: $e \in E$, $x \neq e$ и $y \leqslant e\}$, иначе $F' = F$
$take\_access\_own(x, y, z)$	$x, y, z \in S$, $y \in de\_facto\_own(x)$, $z \in de\_facto\_own(y)$	$S' = S$, $E' = E$, $PA' = PA$, $user' = user$, $roles' = roles$, $A' = A$, $H'_E = H_E$, $de\_facto\_own'(x) = de\_facto\_own(x) \cup \{z\}$, если $x \in (N_S \cup NF_S) \cap S$, то $F' = F \cup \{(x, e, write_t)$: $e \in E$, $x \neq e$ и $z \leqslant e\}$, иначе $F' = F$

Окончание табл. 2

1	2	3
$flow\_memory\_access(x, y, \alpha_a)$	$x \in S$, $y \in E$, $(y, \alpha_a) \in$ $\in de\_facto\_accesses(x)$, где $\alpha_a \in \{read_a, write_a\}$	$S' = S$, $E' = E$, $PA' = PA$, $user' = user$, $roles' = roles$, $A' = A$, $H'_E = H_E$, если $\alpha_a = read_a$, то $F' = F \cup \{(y, x, write_m)\}$, если $\alpha_a = write_a$, то $F' = F \cup \{(x, y, write_m)\}$
$flow\_time\_access(x, y)$	$x \in S$, $y \in E$, $(y, \alpha_a) \in$ $\in de\_facto\_accesses(x)$ или [$y \in S$ и $y \in de\_facto\_own(x)$]	$S' = S$, $E' = E$, $PA' = PA$, $user' = user$, $roles' = roles$, $A' = A$, $H'_E = H_E$, если $x \in (N_S \cup NF_S) \cap S$, то $F' = F \cup \{(x, e, write_t)$: $e \in E$, $x \neq e$ и $y \leqslant e\} \cup \{(e, x, write_t)$: $x \neq e$ и $e \leqslant y\}$, иначе $F' = F$
$flow(x, y, y', z)$	$x, z \in S$, $y, y' \in E$, $x \neq z$, $y \leqslant y'$, [или $x = y$, или $(y, \alpha_a) \in$ $\in de\_facto\_accesses(x)$, или $y \in S$ и $y \in de\_facto\_own(x)$], [или $z = y'$, или $(y', \beta_a) \in de\_facto\_accesses(z)$, или $y' \in S$ и $y' \in de\_facto\_own(z)$], где $\alpha_a, \beta_a \in R_a$	$S' = S$, $E' = E$, $PA' = PA$, $user' = user$, $roles' = roles$, $A' = A$, $H'_E = H_E$, если x, $z \in (N_S \cup NF_S) \cap S$, то $F' = F \cup \{(x, z, write_t)$, $(z, x, write_t)\}$, иначе $F' = F$
$find(x, y, z)$	$x, y \in S$, $z \in E$, $x \neq z$, [$(x, y, \alpha) \in F$, где $\alpha \in \{write_m, write_t\}$], и [или $(z, \beta) \in de\_facto\_accesses(y)$, где $\beta = write_a$, или $(y, z, \beta) \in F$, где $\beta \in \{write_m, write_t\}$]	$S' = S$, $E' = E$, $PA' = PA$, $user' = user$, $roles' = roles$, $A' = A$, $H'_E = H_E$, если $write_t \notin \{\alpha, \beta\}$, то $F' = F \cup \{(x, z, write_m)\}$, если $write_t \in \{\alpha, \beta\}$ и $x, y \in (N_S \cup NF_S) \cap S$, то $F' = F \cup \{(x, z, write_t)\}$, иначе $F' = F$
$post(x, y, z)$	$x, z \in S$, $y \in E$, $x \neq z$, $(y, read_a) \in de\_facto\_accesses(z)$ и [или $(y, \alpha) \in de\_facto\_accesses(x)$, где $\alpha = write_a$, или $(x, y, \alpha) \in F$, где $\alpha \in \{write_m, write_t\}$]	$S' = S$, $E' = E$, $PA' = PA$, $user' = user$, $roles' = roles$, $A' = A$, $H'_E = H_E$, если $\alpha \neq write_t$, то $F' = F \cup \{(x, z, write_m)\}$, если $\alpha = write_t$ и $x, z \in (N_S \cup NF_S) \cap S$, то $F' = F \cup \{(x, z, write_t)\}$, иначе $F' = F$
$pass(x, y, z)$	$y \in S$, $x, z \in E$, $x \neq z$, $(x, read_a) \in de\_facto\_accesses(y)$ и [или $(z, \alpha) \in de\_facto\_accesses(y)$, где $\alpha = write_a$, или $(y, z, \alpha) \in F$, где $\alpha \in \{write_m, write_t\}$]	$S' = S$, $E' = E$, $PA' = PA$, $user' = user$, $roles' = roles$, $A' = A$, $H'_E = H_E$, если $\alpha \neq write_t$, то $F' = F \cup \{(x, z, write_m)\}$, если $\alpha = write_t$ и $y \in (N_S \cup NF_S) \cap S$, то $F' = F \cup \{(x, z, write_t)\}$, иначе $F' = F$
$take\_flow(x, y)$	$x, y \in S$, $x \neq y$, $y \in de\_facto\_own(x)$	$S' = S$, $E' = E$, $PA' = PA$, $user' = user$, $roles' = roles$, $A' = A$, $H'_E = H_E$, если $x \in (N_S \cup NF_S) \cap S$, то $F' = F \cup \{(x, e, \alpha)$: $(y, e, \alpha) \in F$, $e \in E$, $\alpha \in \{write_m, write_t\}\}$, иначе $F' = F \cup \{(x, e, write_m)$: $(y, e, write_m) \in F$, $e \in E\}$

Доступы владения в де-факто правилах $take\_access\_own$, $flow$ и $take\_flow$ заменены на де-факто владение. Новое правило $flow\_memory\_access$ позволяет субъект-сессии, имеющей де-факто доступ на чтение или на запись к сущности, реализовать между ними соответствующий информационный поток по памяти, а правило $flow\_time\_access$ позволяет субъект-сессии, имеющей любой де-факто доступ к сущности или (в случае, когда сущность является субъект-сессией) де-факто владеющей ею, реализовать информационный поток по времени ко всем сущностям и от всех сущностей, которым в иерархии подчинена данная сущность.

Таким образом, зависимости условий и результатов применения де-юре и де-факто правил преобразования состояний РОСЛ ДП-модели существенно изменились. Схе-

ма этих зависимостей показана на рис. 1, на котором сплошными линиями показаны зависимости, возникающие при применении де-юре правил (за исключением информационных потоков), а прерывистыми линиями — зависимости, возникающие при применении де-факто правил или в результате получения информационных потоков при применении де-юре правил.

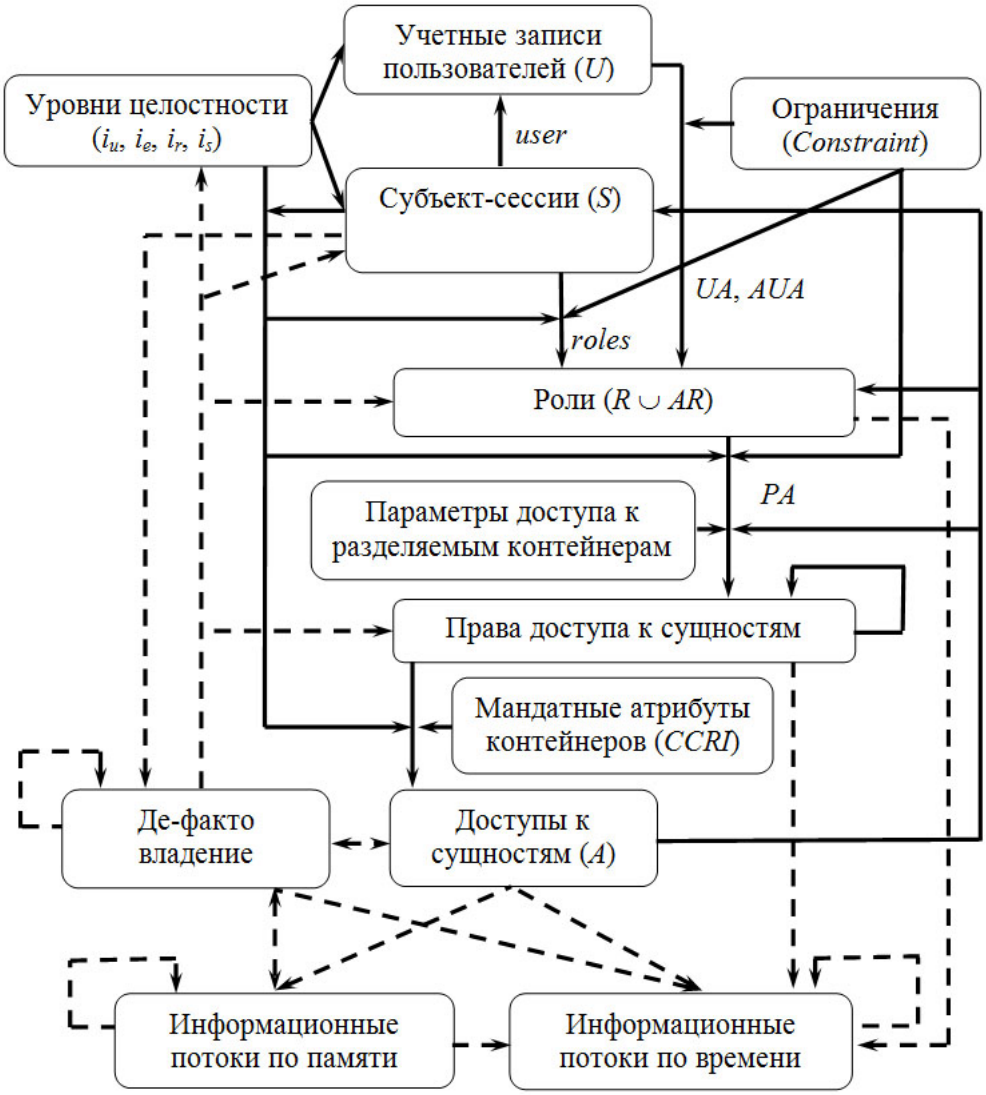

Рис. 1. Зависимости условий и результатов применения правил преобразования состояний РОСЛ ДП-модели

4. Выполнение ограничений и требований мандатного контроля целостности на траекториях системы

Поскольку в рамках РОСЛ ДП-модели, в отличие от других ролевых ДП-моделей, используется механизм ограничений, обоснуем следующее утверждение.

Утверждение 1. Пусть G_0 — начальное состояние системы $\Sigma(G^*, OP, G_0)$, в котором функции $(i_u, i_e, i_r, i_s)_0$ удовлетворяют условиям предположения 7. Тогда в любом состоянии G_N любой траектории $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_N} G_N$, где $op_1, \ldots, op_N$ — правила преобразования состояний и $N \geqslant 0$, функции UA_N, PA_N и $roles_N$ удовлетворяют

соответствующим ограничениям $Constraint_U$, $Constraint_P$ и $Constraint_S$ и функции $(i_u, i_e, i_r, i_s)_N$ удовлетворяют условиям предположения 7.

Доказательство. Докажем утверждение индукцией по длине N траектории функционирования системы.

Пусть $N = 0$, тогда по предположению 6 в состоянии G_0 функции UA_0, PA_0 и $roles_0$ удовлетворяют соответствующим ограничениям $Constraint_U$, $Constraint_P$ и $Constraint_S$ и по условию функции $(i_u, i_e, i_r, i_s)_0$ удовлетворяют условиям предположения 7.

Пусть $N > 0$ и утверждение верно для всех траекторий длины $0 \leqslant L < N$. Пусть $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_N} G_N$ — траектория функционирования системы длины N. По предположению индукции в состоянии G_{N-1} функции UA_{N-1}, PA_{N-1} и $roles_{N-1}$ удовлетворяют соответствующим ограничениям $Constraint_U$, $Constraint_P$ и $Constraint_S$ и функции $(i_u, i_e, i_r, i_s)_{N-1}$ удовлетворяют условиям предположения 7.

Рассмотрим правило преобразования состояний op_N. Если условия его применения не выполняются в состоянии G_{N-1}, то по определению правил преобразования состояний справедливо равенство $G_{N-1} = G_N$ и по предположению индукции в состоянии G_N функции UA_N, PA_N и $roles_N$ удовлетворяют соответствующим ограничениям $Constraint_U$, $Constraint_P$ и $Constraint_S$ и функции $(i_u, i_e, i_r, i_s)_N$ удовлетворяют условиям предположения 7. Пусть условия применения правила преобразования состояний op_N выполняются в состоянии G_{N-1}.

Де-факто правила (за исключением правила $de\_facto\_op(x, op(y, y', \ldots))$ не изменяют значения функций UA, PA, $roles$, (i_u, i_e, i_r, i_s), множества сущностей, субъект-сессий, текущих ролей субъект-сессий, прав доступа ролей, отношения подчиненности в иерархии на множествах сущностей или субъект-сессий. Таким образом, если op_N является де-факто правилом (за исключением правила $de\_facto\_op(x, op(y, y', \ldots))$, по предположению индукции в состоянии G_{N-1} функции UA_{N-1}, PA_{N-1} и $roles_{N-1}$ удовлетворяют соответствующим ограничениям $Constraint_U$, $Constraint_P$ и $Constraint_S$ и функции $(i_u, i_e, i_r, i_s)_{N-1}$ удовлетворяют условиям предположения 7, то в состоянии G_N функции UA_N, PA_N и $roles_N$ также удовлетворяют соответствующим ограничениям $Constraint_U$, $Constraint_P$ и $Constraint_S$ и функции $(i_u, i_e, i_r, i_s)_N$ удовлетворяют условиям предположения 7.

Если op_N является де-факто правилом $de\_facto\_op(x, op(y, y', \ldots))$, то результаты его применения совпадают с результатами применения де-юре правила $op(y, y', \ldots)$, поэтому далее при обосновании шага индукции будем считать, что op_N — де-юре правило.

В рамках РОСЛ ДП-модели не заданы де-юре правила преобразования состояний, изменяющие значение функций UA, i_u, i_r и множества UE. Следовательно, поскольку в состоянии G_{N-1} функция UA_{N-1} удовлетворяет ограничениям $Constraint_U$ и функции $(i_u, i_e, i_r, i_s)_{N-1}$ удовлетворяют условиям предположения 7, то в состоянии G_N функция UA_N удовлетворяет ограничениям $Constraint_U$ и функции $(i_u, i_e, i_r, i_s)_N$ удовлетворяют условиям 1, 4, 6 предположения 7.

Если op_N является де-юре правилом вида $create\_hard\_link(x, x', y, name, z)$, $delete\_hard\_link(x, x', y, z)$, $rename\_entity(x, x', y, name, z)$, $set\_container\_attr(x, x', y, ccri, t)$, $access\_own(x, x', y)$, $access\_read(x, x', y)$, $access\_write(x, x', y)$ или $delete\_access(x, x', y, \alpha_a)$, то оно не изменяет значения функций PA и $roles$. Следовательно, поскольку функции PA_{N-1} и $roles_{N-1}$ соответственно удовлетворяют ограничениям $Constraint_P$ и $Constraint_S$ в состоянии G_{N-1}, то функции PA_N и $roles_N$ соответственно удовлетворяют им в состоянии G_N.

Если op_N является де-юре правилом вида $take\_roles(x, x', \{r_j : 1 \leqslant j \leqslant k\})$ или $remove\_roles(x, x', \{r_j : 1 \leqslant j \leqslant k\})$, то оно не изменяет значение функции PA. Следовательно, поскольку функция PA_{N-1} удовлетворяет ограничениям $Constraint_P$ в состоянии G_{N-1}, то функция PA_N удовлетворяет им в состоянии G_N. При этом соответствие функции $roles_N$ ограничениям $Constraint_S$ в состоянии G_N следует из условий применения правил.

Если op_N является де-юре правилом вида $grant\_rights(x, x', r, \{(y_j, \alpha_{r_j}) : 1 \leqslant j \leqslant k\})$, $remove\_rights(x, x', r, \{(y_j, \alpha_{r_j}) : 1 \leqslant j \leqslant k\})$, $create\_object(x, x', r, y, yi, name, z)$, $create\_container(x, x', r, y, yi, ccri, t, name, z)$ или $delete\_entity(x, x', y, z)$, то оно не изменяет значение функции $roles$. Следовательно, поскольку функция $roles_{N-1}$ удовлетворяет ограничениям $Constraint_S$ в состоянии G_{N-1}, то функция $roles_N$ удовлетворяет им в состоянии G_N. При этом соответствие функции PA_N ограничениям $Constraint_P$ в состоянии G_N следует из условий применения правил.

Если op_N является де-юре правилом вида $create\_first\_session(x, x', u, r, y, z, zi)$, $create\_session(x, x', r, y, z, zi)$ или $delete\_session(x, x', z)$, то из условий применения правил следует, что функции PA_N и $roles_N$ удовлетворяют соответственно ограничениям $Constraint_P$ и $Constraint_S$ в состоянии G_N.

Таким образом, обосновано, что в состоянии G_N функции UA_N, PA_N и $roles_N$ удовлетворяют соответствующим ограничениям $Constraint_U$, $Constraint_P$ и $Constraint_S$.

Если op_N является де-юре правилом вида $rename\_entity(x, x', y, name, z)$, $set\_container\_attr(x, x', y, ccri, t)$, $access\_own(x, x', y)$, $access\_read(x, x', y)$, $access\_write(x, x', y)$ или $delete\_access(x, x', y, \alpha_a)$, то оно не изменяет множества сущностей, субъект-сессий, текущих ролей субъект-сессий, прав доступа ролей, отношения подчиненности в иерархии на множествах сущностей или субъект-сессий. Следовательно, по предположению индукции в состоянии G_N функции $(i_u, i_e, i_r, i_s)_N$ удовлетворяют условиям 2, 3, 5, 7 и 8 предположения 7.

Если op_N является де-юре правилом вида $remove\_roles(x, x', \{r_j : 1 \leqslant j \leqslant k\})$, $remove\_rights(x, x', \{(y_j, \alpha_{r_j}) : 1 \leqslant j \leqslant k\})$, $delete\_entity(x, x', y, z)$, $delete\_hard\_link(x, x', y, z)$ или $delete\_session(x, x', z)$, то оно не добавляет новых элементов во множества сущностей, субъект-сессий, текущих ролей субъект-сессий, прав доступа ролей, не изменяет отношение подчиненности в иерархии на множествах сущностей или субъект-сессий, остающихся в последующем состоянии системы. Следовательно, по предположению индукции в состоянии G_N функции $(i_u, i_e, i_r, i_s)_N$ удовлетворяют условиям 2, 3, 5, 7 и 8 предположения 7.

Если op_N является де-юре правилом вида $take\_roles(x, x', \{r_j : 1 \leqslant j \leqslant k\})$, то оно не добавляет новые элементы во множества сущностей, субъект-сессий, прав доступа ролей, не изменяет отношение подчиненности в иерархии на множествах сущностей или субъект-сессий в последующем состоянии системы. Значит, по предположению индукции в состоянии G_N функции $(i_u, i_e, i_r, i_s)_N$ удовлетворяют условиям 2, 3, 5 и 8 предположения 7. Выполнение условия 7 предположения 7 следует из предположения индукции и из условий и результатов применения правила.

Если op_N является де-юре правилом вида $grant\_rights(x, x', r, \{(y_j, \alpha_{r_j}) : 1 \leqslant j \leqslant k\})$, то оно не добавляет новые элементы во множества сущностей, субъект-сессий, текущих ролей субъект-сессий, не изменяет отношение подчиненности в иерархии на множествах сущностей или субъект-сессий в последующем состоянии системы. Значит, по предположению индукции в состоянии G_N функции $(i_u, i_e, i_r, i_s)_N$ удовлетворяют условиям 2, 3, 5 и 7 предположения 7. Выполнение условия 8 предположения 7 следует из предположения индукции и из условий и результатов применения правила.

Если op_N является де-юре правилом вида $create\_object(x, x', r, y, yi, name, z)$, $create\_container(x, x', r, y, yi, ccri, t, name, z)$ или $create\_hard\_link(x, x', y, name, z)$, то оно не добавляет новые элементы во множества субъект-сессий, текущих ролей субъект-сессий, не изменяет отношение подчиненности в иерархии на множестве субъект-сессий в последующем состоянии системы. Значит, по предположению индукции в состоянии G_N функции $(i_u, i_e, i_r, i_s)_N$ удовлетворяют условиям 3, 5 и 7 предположения 7. Выполнение условий 2 и 8 предположения 7 следует из предположения индукции и из условий и результатов применения правил.

Если op_N является де-юре правилом вида $create\_first\_session(x, x', u, r, y, z, zi)$ или $create\_session(x, x', r, y, z, zi)$, то оно не добавляет новые элементы во множества сущностей (не являющихся субъектами) и не изменяет отношение подчиненности в иерархии на множестве сущностей в последующем состоянии системы. Значит, по предположению индукции в состоянии G_N функции $(i_u, i_e, i_r, i_s)_N$ удовлетворяют условию 2 предположения 7. Выполнение условий 3, 5, 7 и 8 предположения 7 следует из предположения индукции и из условий и результатов применения правила.

Таким образом, обосновано, что в состоянии G_N функции $(i_u, i_e, i_r, i_s)_N$ удовлетворяют условиям 1–8 предположения 7. Шаг индукции доказан. ■

5. Монотонные и немонотонные правила преобразования состояний. Инвариантность ограничений относительно немонотонных правил

По аналогии с существующими ДП-моделями дадим определение.

Определение 1. Монотонное правило преобразования состояний — правило преобразования состояний из множества OP, применение которого не приводит к удалению из состояний:

— ролей из множества текущих ролей субъект-сессий;
— прав доступа ролей к сущностям;
— субъект-сессий, сущностей или «жестких» ссылок на сущности-объекты;
— доступов субъект-сессий к сущностям;
— информационных потоков.

По определению в соответствии с условиями и результатами применения правил преобразования состояний, заданных в табл. 1 и 2, монотонными будут являться следующие правила: $take\_roles(x, x', \{r_j : 1 \leqslant j \leqslant k\})$, $grant\_rights(x, x', r, \{(y_j, r_j) : 1 \leqslant j \leqslant k\})$, $create\_object(x, x', r, y, yi, name, z)$, $create\_container(x, x', r, y, yi, ccri, t, name, z)$, $create\_hard\_link(x, x', y, name, z)$, $rename\_entity(x, x', y, name, z)$, $set\_container\_attr(x, x', y, ccri, t)$, $create\_first\_session(x, x', u, r, y, z, zi)$, $create\_session(x, x', r, y, z, zi)$, $access\_own(x, x', y)$, $access\_read(x, x', y)$, $access\_write(x, x', y)$, $control(x, y, z)$, $know(x, y)$, $take\_access\_own(x, y, z)$, $flow\_memory\_access(x, y, \alpha_a)$, $flow\_time\_access(x, y)$, $flow(x, y, y', z)$, $find(x, y, z)$, $post(x, y, z)$, $pass(x, y, z)$ и $take\_flow(x, y)$. Немонотонными правилами преобразования состояний будут являться де-юре правила $remove\_roles(x, x', \{r_j : 1 \leqslant j \leqslant k\})$, $remove\_rights(x, x', r, \{(y_j, r_j) : 1 \leqslant j \leqslant k\})$, $delete\_entity(x, x', y, z)$, $delete\_hard\_link(x, x', y, z)$, $delete\_session(x, x', z)$, $delete\_access(x, x', y, \alpha_a)$. Де-факто правило $de\_facto\_op(x, op(y, y', \ldots))$ по определению будем считать монотонным или немонотонным в зависимости от монотонности или немонотонности де-юре правила $op(y, y', \ldots)$.

Наличие в РОСЛ ДП-модели механизма ограничений в общем случае может требовать использования монотонных и немонотонных правил при передаче прав доступа ролей или возникновении информационных потоков. Например, пусть в некоторой

системе задано ограничение на значения множеств текущих ролей субъект-сессий, заключающееся в том, что некоторой ролью может одновременно обладать только одна субъект-сессия. Тогда без использования немонотонного правила вида $remove\_roles$ одной субъект-сессией может оказаться невозможным получение данной роли другой субъект-сессией.

Заметим, что в условиях применения правил преобразования состояний, приведенных в табл. 1 и 2, за исключением, возможно, ограничений из множеств $Constraint_P$ и $Constraint_S$, не требуется проверка отсутствия каких-либо элементов (например, сущностей, ролей, прав доступа ролей, информационных потоков), задающих исходное состояние G. По предположению 6 на траекториях системы не изменяются значения функций, задающих уровни целостности сущностей или субъект-сессий, существовавших в предшествующих состояниях системы. Таким образом, в рамках РОСЛ ДП-модели только ограничения обладают свойством, обуславливающим в некоторых случаях необходимость применения немонотонных правил для выполнения в дальнейшем условий применения других правил преобразования состояний. Значит, поиск способов обоснования возможности использования только монотонных правил преобразования состояний при анализе условий передачи прав доступа или возникновения информационных потоков целесообразно осуществлять не в общем случае, а для некоторых заданных для конкретных систем множеств ограничений. Дадим определение и обоснуем утверждение.

Определение 2. Ограничение инвариантно относительно немонотонных правил преобразования состояний в системе $\Sigma(G^*, OP)$, если при условии, что в системе задано только данное ограничение, для любых состояния системы G_0, немонотонного правила преобразования состояний op_1 и правил преобразования состояний op_2, …, op_N, где $N > 1$, справедливо следующее: если $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_{N-1}} G_{N-1}$, $G_0 \vdash_{op_2} G'_2 \vdash_{op_3} \ldots \vdash_{op_{N-1}} G'_{N-1}$ и в состоянии G_{N-1} выполнены ограничения, заданные в условиях применения правила op_N, то эти ограничения выполнены в состоянии G'_{N-1}. Ограничения, заданные в системе $\Sigma(G^*, OP)$, по определению инвариантны относительно немонотонных правил преобразования состояний, когда каждое из ограничений в отдельности инвариантно относительно этих правил.

Так как в правилах преобразования состояний, заданных в рамках РОСЛ ДП-модели, не используются ограничения на значения множеств авторизованных ролей учетных записей пользователей, то по определению 2 в любой системе все ограничения из множества $Constraint_U$ инвариантны относительно немонотонных правил преобразования состояний.

Утверждение 2. Пусть G_0 — начальное состояние системы $\Sigma(G^*, OP, G_0)$, в котором все ограничения инвариантны относительно немонотонных правил преобразования состояний, и функции $(i_u, i_e, i_r, i_s)_0$ удовлетворяют условиям предположения 7. Пусть также существуют состояния системы G_1, …, $G_N = (PA_N, user_N, roles_N, A_N, F_N, H_{E_N})$ и правила преобразования состояний op_1, …, op_N, такие, что $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_N} G_N$, где $N \geqslant 0$. Тогда существуют состояния G'_1, …, $G'_M = (PA'_M, user'_M, roles'_M, A'_M, F'_M, H'_{E_M})$, где $M \geqslant 0$, и монотонные правила преобразования состояний op'_1, …, op'_M, такие, что $G_0 \vdash_{op'_1} G'_1 \vdash_{op'_2} \ldots \vdash_{op'_M} G'_M$ и выполняются следующие условия:

1. Верно включение $S_N \subset S'_M$ и для каждой субъект-сессии $s \in S_N$ выполняются условия $user_N(s) = user'_M(s)$, $roles_N(s) \subset roles'_M(s)$, $de\_facto\_own_N(s) \subset$ $\subset de\_facto\_own'_M(s)$.

2. Верно включение $E_N \subset E'_M$, для каждой сущности $e \in E_N \setminus S_N$, не являющейся субъектом, выполняется условие $H_{E_N}(e) \subset H'_{E_M}(e)$, для любых сущностей $e, e' \in E_N$ если в состоянии G_N выполняется условие $e < e'$, то данное условие выполняется в состоянии G'_M.

3. Для каждой роли $r \in R$ выполняется условие $PA_N(r) \subset PA'_M(r)$.

4. Верно включение $A_N \subset A'_M$.

5. Верно включение $F_N \subset F'_M$.

6. Функции $(i_u, i_e, i_r, i_s)'_M$ удовлетворяют условиям предположения 7.

Доказательство. Пусть существуют состояния G_0, G_1, …, G_N системы $\Sigma(G^*, OP, G_0)$ и правила преобразования состояний op_1, …, op_N, такие, что $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_N} G_N$, где $N \geqslant 0$. Докажем утверждение индукцией по длине N последовательности состояний.

Пусть $N = 0$. Тогда положим $M = 0$, $G'_0 = G_0$, и для состояний G_0 и G'_0 условия 1–6 утверждения выполнены.

Пусть $N > 0$ и условия утверждения выполнены для всех последовательностей состояний длины $0 \leqslant L < N$. Докажем, что условия 1–5 утверждения выполнены для последовательностей состояний длины N.

По предположению индукции для последовательности состояний $G_0, G_1, \ldots, G_{N-1}$ существует последовательность состояний $G_0, G'_1, \ldots, G'_K = (PA'_K, user'_K, roles'_K, A'_K, F'_K, H'_{E_K})$, где $K \geqslant 0$, и для состояний G_{N-1} и G'_K выполнены условия 1–5 утверждения.

Рассмотрим правило преобразования состояний op_N. Если условия его применения не выполняются в состоянии G_{N-1}, то справедливо равенство $G_{N-1} = G_N$. Положим $M = K$, и для состояний G_N и G'_M выполнены условия 1–5 утверждения. Пусть условия применения правила op_N выполняются в состоянии G_{N-1}, тогда возможны два случая.

Первый случай: правило преобразования состояний op_N является немонотонным.

Если в $op_N(x, \ldots)$ субъект-сессия x принадлежит $LF_S \cap S_{N-1}$, то положим $M = K$, и условия 1–5 утверждения выполняются для состояний G_N и G'_K.

Пусть $x \in (N_S \cup NF_S) \cap S_{N-1}$.

Пусть $op_N = remove\_roles(x, x', \{r_j : 1 \leqslant j \leqslant l\})$. Тогда в результате применения правила возникают информационные потоки по времени из множества $STime = \{(x, s, write_t) : s \in (N_S \cup NF_S) \cap S_{N-1},\ x \neq s$ и $x \in de\_facto\_own_{N-1}(s)\}$. Так как для состояний G_{N-1} и G'_K выполнены условия 1–5 утверждения, то для $s \in S_{N-1} \subset S'_K$ справедливо $de\_facto\_own_{N-1}(s) \subset de\_facto\_own'_K(s)$. Положим $M = K + |STime|$ и последовательно для каждой субъект-сессии $s_i \in (N_S \cup NF_S) \cap S_{N-1}$, такой, что $x \neq s_i$ и $x \in de\_facto\_own_{N-1}(s_i)$, положим $op'_i = flow\_time\_access(s_i, x)$, где $K < i \leqslant M$. Пусть состояние G'_M получено из состояния G'_K в результате применения последовательности правил op'_{K+1}, …, op'_M: $G'_K \vdash_{op'_{K+1}} \ldots \vdash_{op'_M} G'_M$. Таким образом, для состояний G_N и G'_M выполнены условия 1–5 утверждения.

Пусть $op_N = remove\_rights(x, x', r, \{(y_j, r_j) : 1 \leqslant j \leqslant k\})$. Тогда в результате применения правила возникают информационные потоки по времени из множества $STime = \{(x, s, write_t) : s \in (N_S \cup NF_S) \cap S_{N-1},\ x \neq s,\ r \in de\_facto\_roles_{N-1}(s)$ и существует j, такое, что $1 \leqslant j \leqslant k$ и $(y_j, \alpha_a) \in de\_facto\_accesses_{N-1}(s)$, где $\alpha_a \in R_a\}$. Положим $Z = |STime|$. Так как для состояний G_{N-1} и G'_K выполнены условия 1–5 утверждения, то выполняются условия $(x, y_j, own_a) \in A'_K$ и для каждого s_i, к которому создается информационный поток из множества $STime$, выполняются условия $(y_j, \alpha_a) \in de\_facto\_accesses_{N-1}(s_i) \subset de\_facto\_accesses'_K(s_i)$, где $1 \leqslant i \leqslant Z$. Положим $op'_{K+i} = flow(x, y_j, y_j, s_i)$, где $1 \leqslant i \leqslant Z$, и $M = K + Z$. Пусть состояние G'_M

получено из состояния G'_K в результате применения последовательности правил op'_{K+1}, $\ldots$, op'_M: $G'_K \vdash_{op'_{K+1}} \ldots \vdash_{op'_M} G'_M$. Таким образом, для состояний G_N и G'_M выполнены условия 1–5 утверждения.

Пусть $op_N = delete\_entity(x, x', y, z)$. Тогда в результате применения правила возникают информационные потоки по времени, в том числе из множества $STime = \{(x, s, write_t) : s \in (N_S \cup NF_S) \cap S_{N-1}, x \neq s$ и $(y, \alpha_a) \in de\_facto\_accesses_{N-1}(s)$, где $\alpha_a \in R_a\}$. Положим $Z = |STime|$. Так как для состояний G_{N-1} и G'_K выполнены условия 1–5 утверждения, то выполняются условия $(x, z, write_a) \in A'_K$, и для каждого s_i, к которому создается информационный поток из множества $STime$, выполняются условия $(y, \alpha_a) \in de\_facto\_accesses_{N-1}(s_i) \subset de\_facto\_accesses'_K(s_i)$, где $1 \leqslant i \leqslant Z$. Положим $op'_{K+i} = flow(x, y, y, s_i)$, где $1 \leqslant i \leqslant Z$, $op'_{K+Z+1} = flow\_time\_access(x, y)$, и $M = K + Z + 1$. Пусть состояние G'_M получено из состояния G'_K в результате применения последовательности правил op'_{K+1}, $\ldots$, op'_M: $G'_K \vdash_{op'_{K+1}} \ldots \vdash_{op'_M} G'_M$. Таким образом, для состояний G_N и G'_M выполнены условия 1–5 утверждения.

Пусть $op_N = delete\_hard\_link(x, x', y, z)$. Так как для состояний G_{N-1} и G'_K выполнены условия 1–5 утверждения, то выполняются условия $(x, z, write_a) \in A'_K$, $i_e(y) \leqslant i_r(r) \leqslant i_s(w)$, $i_e(y) \leqslant i_e(z) \leqslant i_s(w)$, и если $i_e(z) = i\_high$, то $(x', i\_entity, write_a) \in A'_K$. Следовательно, в состоянии G'_K выполнены условия применения правила $create\_hard\_link(x, x', y, name, z)$, где $name \in NAME \setminus \{«»\}$ — некоторое допустимое имя сущности. Тогда положим $M = K + 1$, $op'_M = create\_hard\_link(x, x', y, name, z)$, и пусть состояние G'_M получено из состояния G'_K применением к нему правила op'_M: $G'_K \vdash_{op'_M} G'_M$. Таким образом, для состояний G_N и G'_M выполнены условия 1–5 утверждения.

Пусть $op_N = delete\_session(x, x', z)$. Тогда в результате применения правила возникают информационные потоки по времени, в том числе из множества $STime = \{(x, s, write_t) : s \in (N_S \cup NF_S) \cap S_{N-1}, x \neq s$ и $z \in de\_facto\_own_{N-1}(s)\}$. Положим $Z = |STime|$. Так как для состояний G_{N-1} и G'_K выполнены условия 1–5 утверждения, то выполняются условия $(x, z, write_a) \in A'_K$, и для каждого s_i, к которому создается информационный поток из множества $STime$, выполняются условия $z \in de\_facto\_own_{N-1}(s_i) \subset de\_facto\_own'_K(s_i)$, где $1 \leqslant i \leqslant Z$. Положим $op'_{K+i} = flow(x, z, z, s_i)$, где $1 \leqslant i \leqslant Z$, $op'_{K+Z+1} = flow\_time\_access(x, z)$ и $M = K + Z + 1$. Пусть состояние G'_M получено из состояния G'_K в результате применения последовательности правил op'_{K+1}, $\ldots$, op'_M: $G'_K \vdash_{op'_{K+1}} \ldots \vdash_{op'_M} G'_M$. Таким образом, для состояний G_N и G'_M выполнены условия 1–5 утверждения.

Пусть $op_N = delete\_access(x, x', y, \alpha_a)$. Так как для состояний G_{N-1} и G'_K выполнены условия 1–5 утверждения, то выполняется условие $(x, y, \alpha_a) \in A'_K$. Положим $M = K + 1$, $op'_M = flow\_time\_access(x, y)$, и пусть состояние G'_M получено из состояния G'_K применением к нему правила op'_M: $G'_K \vdash_{op'_M} G'_M$. Таким образом, для состояний G_N и G'_M выполнены условия 1–5 утверждения.

Пусть $op_N = de\_facto\_op(x, op(y, y', \ldots))$, где $op(y, y', \ldots)$ — немонотонное правило преобразования состояний. Так как для состояний G_{N-1} и G'_K выполнены условия 1–5 утверждения, то $de\_facto\_own_{N-1}(x) \subset de\_facto\_own'_K(x)$ и в состоянии G'_K выполняются условия применения правила $op(y, y', \ldots)$, являющегося одним из шести рассмотренных немонотонных правил. Следовательно, повторяя рассуждения для этих правил, получаем состояние G'_M, такое, что для состояний G_N и G'_M выполнены условия 1–5 утверждения.

Второй случай: правило преобразования состояний op_N является монотонным. Положим $M = K + 1$, $op'_M = op_N$, таким образом, монотонные правила исходной после-

довательности $op_1, \ldots, op_N$ без изменений добавляются в последовательность $op'_1, \ldots, op'_M$. Так как для состояний G_{N-1} и G'_K выполнены условия 1–5 утверждения и по предположению 6 на траекториях системы не изменяются значения функций, задающих уровни целостности сущностей или субъект-сессий, существовавших в предшествующих состояниях системы, то в состоянии G'_K выполняются все условия применения правила op_N, за исключением, возможно, ограничений, заданных в op_N. Данные ограничения выполнены в состоянии G_{N-1}, предположим, что они не выполнены в состоянии G'_K.

Если $N = 1$, то $G'_K = G_{N-1} = G_0$, противоречие. Следовательно, выполняется неравенство $N > 1$. Будем считать, что из последовательности $op_1, \ldots, op_{N-1}$ удалены все правила, условия применения которых не выполняются. Если последовательность $op_1, \ldots, op_{N-1}$ состоит только из монотонных правил, то справедливо $K = N-1$, $op_i = op'_i$, где $1 \leqslant i \leqslant N-1$, и $G'_K = G_{N-1}$. Противоречие. Следовательно, в последовательности $op_1, \ldots, op_{N-1}$ есть немонотонные правила. По определению 2, данные немонотонные правила, начиная с конца последовательности $op_1, \ldots, op_{N-1}$, могут быть удалены из нее, и на их место могут быть вставлены соответствующие монотонные правила вида $create\_hard\_link(x, x', y, name, z)$, $flow\_time\_access(x, y)$, $flow(x, y, y', z)$, использованные в доказательстве для первого случая. Эти правила добавляют все информационные потоки, которые создавались соответствующими немонотонными правилами, и не влияют на выполнение ограничений (так как не изменяют значения функций PA и $roles$). В результате будет получена последовательность монотонных правил $op'_1, \ldots, op'_K$, и по определению 2 в состоянии G'_K должно выполняться ограничение, заданное в правиле op_N. Противоречие. Значит, данное ограничение выполнено в состоянии G'_K.

Таким образом, пусть состояние G'_M получено из состояния G'_K с использованием монотонного правила op_N: $G'_K \vdash_{op_N} G'_M$. В соответствии с заданными в табл. 1 и 2 результатами применения правил преобразования состояний для состояний G_N и G'_M выполнены условия 1–5 утверждения.

Выполнение условия 6 утверждения во всех рассмотренных случаях следует из утверждения 1.

Следовательно, условия 1–6 утверждения выполнены для последовательностей состояний длины N, и шаг индукции доказан. ■

Таким образом, в рамках РОСЛ ДП-модели возможно задание ограничений, позволяющих при дальнейшем анализе условий передачи прав доступа ролей, реализации информационных потоков по памяти или по времени использовать только монотонные правила преобразования состояний.

Утверждение 3. Пусть G_0 — начальное состояние системы $\Sigma(G^*, OP, G_0)$, в которой все ограничения инвариантны относительно немонотонных правил преобразования состояний, и $op \in OP$ — правило преобразования состояний. Если в состоянии G_0 выполнены ограничения, заданные в условиях применения правила op, то эти ограничения выполнены в состоянии G_N любой траектории системы $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_N} G_N$, где $op_1, op_2, \ldots, op_N$ — монотонные правила преобразования состояний и $N \geqslant 1$.

Доказательство. Предположим противное. Пусть существуют монотонные правила преобразования состояний $op_1, op_2, \ldots, op_N$, где $N \geqslant 1$, такие, что $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_N} G_N$ — траектория системы, и в состоянии G_N не выполнены ограничения, заданные в условиях применения правила op. Выберем N минимальной длины, тогда в состоянии G_{N-1} выполнены ограничения, заданные в условиях применения правила op. Значит, без ограничения общности можно считать, что существует монотонное

правило преобразования состояний op_1, такое, что $G_0 \vdash_{op_1} G_1$, в состоянии G_1 не выполнены ограничения, заданные в условиях применения правила op, при этом условия применения правила op_1 выполнены в состоянии G_0. Кроме того, по предположению 6 в состоянии G_0 функции UA_0, PA_0 и $roles_0$ удовлетворяют соответствующим ограничениям $Constraint_U$, $Constraint_P$ и $Constraint_S$.

Правило op не может являться правилом вида $create\_hard\_link(x, x', y, name, z)$, $rename\_entity(x, x', y, name, z)$, $set\_container\_attr(x, x', y, ccri, t)$, $access\_own(x, x', y)$, $access\_read(x, x', y)$, $access\_write(x, x', y)$, $control(x, y, z)$, $know(x, y)$, $take\_access\_own(x, y, z)$, $flow\_memory\_access(x, y, \alpha_a)$, $flow\_time\_access(x, y)$, $flow(x, y, y', z)$, $find(x, y, z)$, $post(x, y, z)$, $pass(x, y, z)$ и $take\_flow(x, y)$, а также правилом вида $de\_facto\_op(x, op'_1(y, y', \ldots))$, где $op'_1(y, y', \ldots)$ — де-юре правило одного из перечисленных видов. Эти правила не изменяют значения функций UA, PA и $roles$, поэтому после их применения выполнявшиеся в состоянии G_0 ограничения, заданные в условиях применения правила op, должны выполняться в состоянии G_1. Следовательно, возможны семь случаев.

Первый случай: $op_1 = take\_roles(x, x', \{r_j : 1 \leqslant j \leqslant k\})$. Положим $op_2 = remove\_roles(x, x', \{r_j : 1 \leqslant j \leqslant k\})$.

Второй случай: $op_1 = grant\_rights(x, x', r, \{(y_j, r_j) : 1 \leqslant j \leqslant k\})$. Положим $op_2 = remove\_rights(x, x', r, \{(y_j, r_j) : 1 \leqslant j \leqslant k\})$.

Третий и четвертый случаи: $op_1 = create\_object(x, x', r, y, yi, name, z)$ или $op_1 = create\_container(x, x', r, y, yi, ccri, t, name, z)$. Положим $op_2 = delete\_entity(x, x', y, z)$.

Пятый и шестой случаи: $op_1 = create\_first\_session(x, x', u, r, y, z, zi)$ или $op_1 = create\_session(x, x', r, y, z, zi)$. Положим $op_2 = delete\_session(x, x', z)$.

Седьмой случай: $op_1 = de\_facto\_op(x, op'_1(y, y', \ldots))$, где $op'_1(y, y', \ldots)$ — де-юре правило одного из рассмотренных шести видов. Положим $op_2 = de\_facto\_op(x, op'_2(y, y', \ldots))$, где $op'_2(y, y', \ldots)$— соответствующее для каждого из рассмотренных шести случаев немонотонное де-юре правило.

Рассмотрим траекторию $G_0 \vdash_{op_1} G_1 \vdash_{op_2} G_2$. Во всех семи случаях выполнены условия применения немонотонного правила op_2, в том числе: так как $UA_0 = UA_2$, $PA_0 = PA_2$, $roles_0 = roles_2$, то выполнены ограничения, заданные в условиях применения правила op_2. Следовательно, в состоянии G_2 выполнены ограничения, заданные в условиях применения правила op. Следовательно, по определению 2 эти ограничения выполнены в состоянии G_1. Противоречие.

Утверждение доказано. ■

Таким образом, если в системе заданы ограничения, инвариантные относительно немонотонных правил преобразования состояний, и в ее начальном состоянии выполнены ограничения, заданные в условиях применения некоторого монотонного правила, то эти ограничения будут выполнены в любом состоянии на любой траектории системы, полученной из начального состояния в результате применения монотонных правил преобразования состояний.

6. Способ задания индивидуальных ролей учетных записей пользователей

При реализации в существующих ОС (использующих, как правило, дискреционное управление доступом) ролевого управления доступом, возможно, будет необходимо задание для каждой учетной записи пользователя индивидуальных прав доступа к сущностям. Например, эта необходимость возникает в следующих случаях:

— когда имеются сущности-файлы, принадлежащие только субъект-сессиям, функционирующим от имени учетной записи некоторого пользователя;
— когда субъект-сессия, создавая новую субъект-сессию, должна получить к ней право доступа владения.

Для обеспечения такой возможности в РОСЛ ДП-модели целесообразно для каждой учетной записи пользователя включить во множество ее авторизованных ролей принадлежащие только ей пары (для каждого уровня целостности, не превосходящего уровень целостности учетной записи пользователя) ролей: индивидуальную роль и индивидуальную административную роль, задав соответствующие параметры ограничений. Таким образом, сделаем предположение, дополняющее предположение 1.

Предположение 12. Для каждой учетной записи пользователя $u \in U$, каждого уровня целостности $i \leqslant i_u(u)$ существуют роль $u\_role\_i \in R$ и административная роль $u\_admin\_role\_i \in AR$, которые имеют следующие свойства:

— обладают уровнем целостности i;
— входят во множества авторизованных ролей и авторизованных административных ролей соответственно только учетной записи пользователя u;
— множества параметрически ассоциированных с ними сущностей совпадают с множеством сущностей, параметрически ассоциированных с учетной записью пользователя u.

С использованием административной роли $u\_admin\_role\_i$ разрешено включать или удалять права доступа во множество прав доступа роли u_role_i. При этом не накладываются следующие ограничения:

— из множества $Constraint_U$, требующие наличия во множестве авторизованных ролей учетной записи пользователя u ролей u_role_i и $u\_admin\_role\_i$;
— из множества $Constraint_P$, разрешающие получение любой ролью u_role_i права доступа владения к сущности при ее создании субъект-сессией, функционирующей от имени учетной записи пользователя u, или получение любой ролью u_role_i любого права доступа к сущности при наличии у роли u_role_i права доступа владения к этой сущности;
— из множества $Constraint_S$, разрешающие обладание субъект-сессиями, функционирующими от имени учетной записи пользователя u, текущими ролями u_role_i и $u\_admin\_role\_i$.

Таким образом, для каждой учетной записи пользователя $u \in U$ выполняются условия:

— $u\_role\_i \in UA(u)$, $u\_admin\_role\_i \in AUA(u)$;
— $i_r(u\_role\_i) = i_r(u\_admin\_role\_i) = i$;
— $]u\_role\_i[=]u\_admin\_role\_i[=]u[$;
— $u\_role\_i \in can\_manage\_rights(u\_admin\_role\_i)$.

В результате предположение 12 позволяет задать функции авторизованных ролей учетных записей пользователей, текущих ролей субъект-сессий и прав доступа ролей с учетом одной из особенностей реализации управления доступом в ОС.

Заключение

Основным направлением развития РОСЛ ДП-модели стала ее поэтапная адаптация к условиям функционирования реальных ОС семейства *AltLinux*. При этом в первую очередь элементы модели, правила преобразования состояний были переопределены с целью обеспечения четкого разделения де-юре и де-факто условий функционирования системы. Кроме того, включение в модель механизма ограничений потребова-

ло описания особого их вида — ограничений, инвариантных относительно немонотонных правил преобразования состояний, позволяющих при анализе условий передачи прав доступа ролей, получения доступов и возникновения информационных потоков по памяти или по времени использовать только монотонные правила. Таким образом, были созданы предпосылки для формулирования и обоснования в рамках РОСЛ ДП-модели условий нарушения безопасности системы и практической проверки их корректности на макете реальной ОС.

ЛИТЕРАТУРА

1. *Девянин П. Н.* Модели безопасности компьютерных систем. Управление доступом и информационными потоками. Учеб. пособие для вузов. М.: Горячая линия-Телеком, 2011. 320 с.
2. *Девянин П. Н., Захаренков П. С.* Способ реализации информационного потока по времени в операционных системах с мандатным управлением доступом через clipboard // Методы и технические средства обеспечения безопасности информации: материалы Юбилейной 20-й науч.-техн. конф. 27 июня – 01 июля 2011 г. СПб.: Изд-во Политехн. ун-та, 2011. С. 76–77.
3. *Колегов Д. Н.* ДП-модель компьютерной системы с функционально и параметрически ассоциированными с субъектами сущностями // Вестник Сибирского государственного аэрокосмического университета им. акад. М. Ф. Решетнева. 2009. Вып. 1(22). Ч. 1. С. 49–54.
4. *Девянин П. Н.* Правила преобразования состояний базовой ролевой ДП-модели управления доступом и информационными потоками в операционных системах // Прикладная дискретная математика. 2011. № 1(11). С. 78–95.
5. *Девянин П. Н.* Моделирование ролевого управления доступом в операционных системах семейства Linux // Проблемы информационной безопасности. Компьютерные системы. 2011. № 1. С. 24–43.