4.2. Модели безопасности

Формальные модели безопасности являются основным инструментом, используемым для доказательства соответствия системы защиты АС заданной политике безопасности. Кроме того, результаты анализа формальных моделей позволяют систематизировать и направлять научные исследования по вопросам анализа и построения защищенных АС.

В настоящем разделе рассмотрены модели безопасности, используемые для анализа систем защиты АС, в которых реализованы дискреционная политика безопасности (модели *HRU*, *Take-Grant*), мандатная политика безопасности (модель *Bell-LaPadula* и ее интерпретации, *LWM*) и модели безопасности информационных потоков.

Модель матрицы доступов *HRU*

Основные положения модели

Модель *HRU* используется для анализа системы защиты, реализующей дискреционную политику безопасности, и ее основного элемента — матрицы доступов. При этом система защиты представляется конечным автоматом, функционирующим согласно определенным правилам перехода.

Модель *HRU* была впервые предложена *B. Lampson* в 1971 году (см. [13]). В 1976 году *Harrison M.A., Ruzzo W.L., Ullman J.D.* представили формальное описание модели, которым мы будем руководствоваться по [9] и [17].

Обозначим через:

O — множество объектов системы;

S — множество субъектов системы ($S \subseteq O$);

R — множество прав доступа субъектов к объектам, например, права на чтение (*read*), на запись (*write*), владения (*own*);

M — матрица доступа, строки которой соответствуют субъектам, а столбцы соответствуют объектам. $M[s, o] \subseteq R$ — права доступа субъекта s к объекту o.

Отдельный автомат, построенный согласно положениям модели *HRU*, будем называть *системой*. Функционирование системы рассматривается только с точки зрения изменений в матрице доступа. Возможные изменения определяются шестью примитивными операторами:

- «Внести» право $r \in R$ в $M[s, o]$ — добавление субъекту s право доступа r на объект o. При этом в ячейку $M[s, o]$ матрицы доступов добавляется элемент r;
- «Удалить» право $r \in R$ из $M[s, o]$ — удаление у субъекта s права доступа r на объект o. При этом из ячейки $M[s, o]$ матрицы доступов удаляется элемент r;
- «Создать» субъекта s' — добавление в систему нового субъекта s'. При этом в матрицу доступов добавляются новые столбец и строка;
- «Создать» объект o' — добавление в систему нового объекта o'. При этом в матрицу доступов добавляется новый столбец;
- «Уничтожить» субъекта s' — удаление из системы субъекта s'. При этом из матрицы доступов удаляются соответствующие столбец и строка;
- «Уничтожить» объект o' — удаление из системы объекта o'. При этом из матрицы доступов удаляется соответствующий столбец.

В результате выполнения примитивного оператора α осуществляется переход из состояния $Q = (S, O, M)$ в новое состояние $Q' = (S', O', M')$ (см. *таблицу 4.2.1.*). Данный переход обозначим через $Q \vdash_\alpha Q'$.

Таблица 4.2.1. Примитивные операторы модели *HRU*

Оператор — «*op*»	Условия выполнения	Новое состояние

«Внести» право $r \in R$ в $M[s, o]$	$s \in S$ $o \in O$	$S' = S, O' = O, M'[s, o] = M[s, o] \cup \{ r \}$, $(s', o') \neq (s, o) \Rightarrow M'[s', o'] = M[s', o']$
«Удалить» право $r \in R$ из $M[s, o]$	$s \in S$ $o \in O$	$S' = S, O' = O, M'[s, o] = M[s, o] \setminus \{ r \}$, $(s', o') \neq (s, o) \Rightarrow M'[s', o'] = M[s', o']$
«Создать» субъект s'	$s' \notin S$	$S' = S \cup \{ s' \}, O' = O \cup \{ s' \}$, $(s, o) \in S \times O \Rightarrow M'[s, o] = M[s, o]$, $o \in O' \Rightarrow M'[s', o] = \varnothing$, $s \in S' \Rightarrow M'[s, s'] = \varnothing$
«Создать» объект o'	$o' \notin O$	$S' = S, O' = O \cup \{ o' \}$, $(s, o) \in S \times O \Rightarrow M'[s, o] = M[s, o]$, $s \in S' \Rightarrow M'[s, o'] = \varnothing$
«Уничтожить» субъект s'	$s' \in S$	$S' = S / \{ s' \}, O' = O / \{ s' \}$, $(s, o) \in S' \times O' \Rightarrow M'[s, o] = M[s, o]$
«Уничтожить» объект o'	$o' \in O$ $o' \notin S$	$S' = S, O' = O / \{ o' \}$, $(s, o) \in S' \times O' \Rightarrow M'[s, o] = M[s, o]$

Из примитивных операторов могут составляться команды. Каждая команда состоит из двух частей:

- условия, при котором выполняется команда,
- последовательности примитивных операторов.

Таким образом, запись команды имеет вид:

command $C(x_1, \ldots, x_k)$

if $r_1 \in M[x_{s1}, x_{o1}]$ *and* ... *and* $r_m \in M[x_{sm}, x_{om}]$ *then*

α_1 ;

...

α_n ;

end

Где $r_1, \ldots, r_m \in R$ — права доступа, $\alpha_1, \ldots, \alpha_n$ — последовательность примитивных операторов. Следует отметить, что условия в теле команды необязательны.

При выполнении команды $C(x_1, \ldots, x_k)$ система осуществляет переход из состояния Q в новое состояние Q'. Данный переход обозначим через:

$Q \vdash_{c(x1, \ldots, xk)} Q'$.

При этом справедливо:

- $Q' = Q$, если одно из условий команды $C(x_1, \ldots, x_k)$ не выполнено;
- $Q' = Q_n$, если все условия команды $C(x_1, \ldots, x_k)$ и существуют состояния $Q_1, \ldots, Q_n$: $Q = Q_0 \vdash_{\alpha 1} Q_1 \vdash_{\alpha 2} \ldots \vdash_{\alpha n} Q_n$.

Пример 1. Команда создания субъектом s личного файла f.

command «создать файл»(s, f):

«создать» объект f;

«внести» право владения *own* в $M[s, f]$;

«внести» право на чтение *read* в $M[s, f]$;

«внести» право на запись *write* в $M[s, f]$;

end

Пример 2. Команда передачи субъекту s' права *read* на файл f его владельцем субъектом s.

command «передать право чтения» (*s, s', f*) :
if own ∈ *M[s, f] then*
«внести» право *read* в *M[s', f]*;
end

Безопасность системы

Согласно требованиям большинства критериев оценки безопасности, системы защиты должны строится на основе определенных математических моделей. С использованием данных моделей должно быть теоретически обосновано соответствие системы защиты требованиям заданной политики безопасности. Для решения поставленной задачи необходим алгоритм, позволяющий осуществлять данную проверку. Однако, как показывают результаты анализа модели *HRU*, задача построения алгоритма проверки безопасности систем, реализующих дискреционную политику разграничения прав доступа, не может быть решена в общем случае.

Определение 1. Будем считать, что возможна утечка права $r \in R$ в результате выполнения команды *c*, если при переходе системы в состояние *Q'* выполняется примитивный оператор, вносящий *r* в элемент матрицы доступов *M*, до этого *r* не содержавший.

Определение 2. Начальное состояние Q_o называется безопасным по отношению к некоторому праву *r*, если невозможен переход системы в такое состояние *Q*, в котором возможна утечка права *r*.

Определение 3. Система называется монооперационной, если каждая команда выполняет один примитивный оператор.

Теорема 1. Существует алгоритм, который проверяет, является монооперационная система и ее исходное состояние безопасным для данного права *r*.

Доказательство. Для доказательства достаточно показать, что число последовательностей команд системы, которые необходимо проверить, ограничено. В этом случае алгоритм проверки безопасности — есть алгоритм тотального перебора всех последовательностей и проверки конечного состояния для каждой из них на отсутствие утечки права *r*.

Заметим, что нет необходимости рассматривать в последовательностях команды «удалить» и «уничтожить», т.к. нам необходимо проверить наличие права доступа, а не его отсутствие. Далее заметим, что нет необходимости рассматривать последовательности команд, содержащих более одного оператора «создать». Это связано с тем, что все последовательности команд, которые проверяют или вносят права доступа в новые элементы матрицы доступов *M*, могут быть простой заменой параметров представлены последовательностями, которые действуют с существующими субъектами и объектами. Тем не менее необходимо сохранить одну команду создания субъекта на случай, если $Q_o = (S, O, M)$ и $S = \varnothing$.

Таким образом, надо рассматривать только те последовательности команд, которые содержат операторы «внести» и максимум один оператор «создать» субъект.

Число различных операторов «внести» равно $n = |R| \cdot (|S_0| + 1) \cdot (|O_0| + 1)$. Так как порядок операций «внести» в последовательности команд не важен, то с учетом одной операции «создать» число последовательностей команд ограничено величиной 2^{n+1} .

■

Теорема 2. Задача проверки безопасности произвольных систем алгоритмически неразрешима.

Доказательство. Для доказательства теоремы воспользуемся фактом, доказанным в теории машин Тьюринга: *Не существует алгоритма проверки для произвольной машины Тьюринга и произвольного начального слова остановится ли машина Тьюринга в конечном состоянии или нет*.

Под машиной Тьюринга понимается способ переработки слов в конечных алфавитах. Слова записываются на бесконечную в обе стороны ленту, разбитую на ячейки.

Элементы и команды машины Тьюринга обозначим через:

$A = \{ a_0, a_1, \dots, a_m \}$ — внешний алфавит, где $a_0 = \wedge$ — пустой символ;

$Q = \{ q_0, q_1, \dots, q_k \}$ — внутренний алфавит, где q_1 — начальное состояние, q_0 — конечное состояние;

$D = \{ r, l, e \}$ — множество действий, где r — шаг вправо управляющей головки, l — шаг влево, e — управляющая головка не перемещается;

$C: Q \times A \to Q \times A \times D$ — функция, задающая команды машины Тьюринга. Если $C(q_{i0}, a_{i0}) = (q_{i1}, a_{i1}, d)$, то это означает, что когда машина, находится в состоянии q_{i0} и управляющая головка указывает на ячейку ленты, содержащую символ a_{i0}, тогда выполняется шаг машины, в результате которого в эту ячейку записывается символ a_{i1}, машина переходит в состояние q_{i1}, а управляющая головка смещается по ленте согласно действию d.

Для того чтобы воспользоваться указанным выше фактом, представим все элементы и команды машины Тьюринга в виде элементов и команд модели *HRU*.

Пусть машина Тьюринга выполнила какое-то количество шагов. Занумеруем все ячейки, пройденные считывающей головкой (включая те, в которые была изначально занесена информация) числами от 1 до n. Тогда $(a_{s1}, \dots, a_{sn})$ — заполнение ленты, где $a_{si} \in A$, $si \in \{0, 1, \dots, m\}$ для $i = 1, \dots, n$. Пусть считывающая головка указывает на ячейку с номером $i \in \{1, \dots, n\}$, содержащую символ $a_{it} \in A$, где $it \in \{0, 1, \dots, m\}$, состояние машины $q_{ij} \in Q$, где $ij \in \{1, \dots, k\}$, и должна быть выполнена команда $C(q_{ij}, a_{it}) = (q_{ij'}, a_{it'}, d)$, где $q_{ij'} \in Q$, $a_{it'} \in A$, $it' \in \{0, 1, \dots, m\}$, $ij' \in \{1, \dots, k\}$, $d \in D$.

Каждой ячейке ленты поставим в соответствие субъекта модели *HRU*, при этом будем считать, что $O = S = \{ s_1, \dots, s_n \}$. Зададим матрицу доступов M в текущем состоянии. Пусть множество прав доступа $R = Q \cup A \cup \{own\}$ и в матрицу доступов внесены права:

- $a_{sj} \in M[s_j, s_j]$ для $j = 1, \dots, n$ — заполнение ленты;
- $own \in M[s_j, s_{j+1}]$ для $j = 1, \dots, n\text{-}1$ — упорядочивание субъектов, соответствующих ячейкам ленты;
- $q_{ij} \in M[s_i, s_i]$ — управляющая головка указывает на ячейку с номером i.

Таким образом, текущему состоянию машины Тьюринга соответствует матрица доступов M модели *HRU* следующего вида:

	s_1	s_2	s_3	…	s_i	…	s_{n-1}	s_n
s_1	a_{s1}	*own*						
s_2		a_{s2}	*own*					
s_3			a_{s3}					
…								
s_i					a_{si}, q_{ij}			
…								
s_{n-1}							$a_{s\,n-1}$	*own*
s_n								$a_{s\,n}$

Для каждой команды машины Тьюринга $C(q_{ij}, a_{it}) = (q_{ij'}, a_{it'}, d)$ зададим соответствующую ей команду $cd(q_{ij}, a_{it}, q_{ij'}, a_{it'})$ модели *HRU*:

- Если $d = e$, то

command $ce(q_{ij}, a_{it}, q_{ij'}, a_{it'})$:
  if $(q_{ij} \in M[s, s])$ *and* $(a_{it} \in M[s, s])$ *then*
    «удалить» право q_{ij} из $M[s, s]$;
    «удалить» право a_{it} из $M[s, s]$;
    «внести» право $q_{ij'}$ в $M[s, s]$;
    «внести» право $a_{it'}$ в $M[s, s]$;
  end;

- Если $d = r$, то необходимо указать две команды $cr1(q_{ij}, a_{it}, q_{ij'}, a_{it'})$ и $cr2(q_{ij}, a_{it}, q_{ij'}, a_{it'})$ для случаев соответственно, когда считывающая головка не указывает на самую правую ячейку ленты и когда это так.

  command $cr1$ $(q_{ij}, a_{it}, q_{ij'}, a_{it'})$:
  if $(q_{ij} \in M[s, s])$ *and* $(a_{it} \in M[s, s])$ *and* $(own \in M[s, s'])$ *then*
  «удалить» право q_{ij} из $M[s, s]$;
  «удалить» право a_{it} из $M[s, s]$;
  «внести» право $a_{it'}$ в $M[s, s]$;
  «внести» право $q_{ij'}$ в $M[s', s']$;
  end;
  command $cr2$ $(q_{ij}, a_{it}, q_{ij'}, a_{it'})$:
  if $(q_{ij} \in M[s, s])$ *and* $(a_{it} \in M[s, s])$ *and* $(not\ (\exists s' \in S : own \in M[s, s']))$ *then*
  «удалить» право q_{ij} из $M[s, s]$;
  «удалить» право a_{it} из $M[s, s]$;
  «внести» право $a_{it'}$ в $M[s, s]$;
  «создать» субъект s';
  «внести» право *own* в $M[s, s']$;
  «внести» право a_0 в $M[s', s']$;
  «внести» право $q_{ij'}$ в $M[s', s']$;
  end;
- Если $d = l$, то команды $cl1(q_{ij}, a_{it}, q_{ij'}, a_{it'})$, $cl2(q_{ij}, a_{it}, q_{ij'}, a_{it'})$ задаются аналогично командам $cr1(q_{ij}, a_{it}, q_{ij'}, a_{it'})$, $cr2(q_{ij}, a_{it}, q_{ij'}, a_{it'})$.

Если машина Тьюринга останавливается в своем конечном состоянии q_0, то в соответствующей системе *HRU* происходит утечка права доступа q_0. Из алгоритмической неразрешимости задачи проверки остановится ли машина Тьюринга в конечном состоянии, следует аналогичный вывод для задачи проверки безопасности соответствующих им систем *HRU*. Таким образом, в общем случае для систем дискреционного разграничения доступа, построенных на основе модели *HRU*, задача проверки безопасности алгоритмически неразрешима.

■

Приведенные выше *теорема 1* и *теорема 2* представляют проблему выбора перед разработчиками систем защиты. С одной стороны, общая модель *HRU* может выражать большое разнообразие политик дискреционного разграничения доступа, но при этом не существует алгоритма проверки их безопасности. С другой стороны, можно использовать монооперационные системы, для которых алгоритм проверки безопасности существует, но данный класс систем является слишком узким. Например, монооперационные системы не могут выразить политику, дающую субъектам права на созданные ими объекты, т.к. не существует одной операции, которая и создает объект, и помечает его как принадлежащий создающему субъекту одновременно.

Дальнейшие исследования модели *HRU* велись в основном в направлении определения условий, которым должна удовлетворять система, чтобы для нее задача проверки безопасности была алгоритмически разрешима. Так в 1976 году *Harrison M.A., Ruzzo W.L., Ullman J.D.* в работе [10] доказали, что указанная задача разрешима для систем, в которых нет операции «создать». В 1978 году (см. [11]) они показали, что таковыми могут быть системы *монотонные* и *моноусловные*, т.е. не содержащие операторов «уничтожить» или «удалить» и имеющие только команды, чьи части условия имеют, не более одного предложения. В том же году *R. Lipton, L. Snyder* (см. [14]) показали, что задача безопасности для систем с конечным множеством субъектов разрешима, но вычислительно сложна.

Модель распространения прав доступа *Take-Grant*

Основные положения модели

Модель распространения прав доступа *Take-Grant*, предложенная в 1976 году (см. [12]), используется для анализа систем дискреционного разграничения доступа, в первую очередь для анализа путей распространения прав доступа в таких системах. В качестве основных элементов модели используются граф доступов и правила его преобразования. Цель модели в первую очередь дать ответ на вопрос о возможности получения прав доступа субъектом системы на объект в состоянии, описываемом графом доступов. В настоящее время, модель *Take-Grant* получила продолжение как расширенная модель *Take-Grant* (см. [8]), в которой рассматриваются пути возникновения информационных потоков в системах с дискреционным разграничением доступа.

Перейдем к формальному описанию модели *Take-Grant*, которое приведем по работам [2] и [7].

Обозначим через:

O — множество объектов (например, файлов или сегментов памяти);

$S \subseteq O$ — множество активных объектов — субъектов (например, пользователей или процессов);

$R = \{ r_1, r_2, \ldots, r_m \} \cup \{ t, g \}$ — множество прав доступа, где t (*take*) — право брать права доступа, g (*grant*) — право давать права доступа;

$G = (S, O, E)$ — конечный помеченный ориентированный граф, без петель, представляющий текущие доступы в системе. Множества S, O представляют вершины графа, которые будем обозначать, соответственно, объекты (элементы множества $O \setminus S$) через — ⊗, субъекты (элементы множества S) через — • . Элементы множества $E \subseteq O \times O \times R$ представляют дуги графа, помеченные непустыми подмножествами из множества прав доступа R.

Состояние системы описывается его графом доступов. Переход системы из состояния в состояние определяется операциями или правилами преобразования графа доступов. Преобразование графа G в граф G' в результате выполнения правила op обозначим через

$$G \vdash_{op} G'.$$

В классической модели *Take-Grant* правило преобразования графа может быть одним из четырех, перечисленных ниже:

Правило *«Брать» — take(* α, x, y, z *)* . Пусть $x \in S$, $y, z \in O$ — различные вершины графа G, $\beta \subseteq R$, $\alpha \subseteq \beta$. Правило определяет порядок получения нового графа доступов G' из графа G.

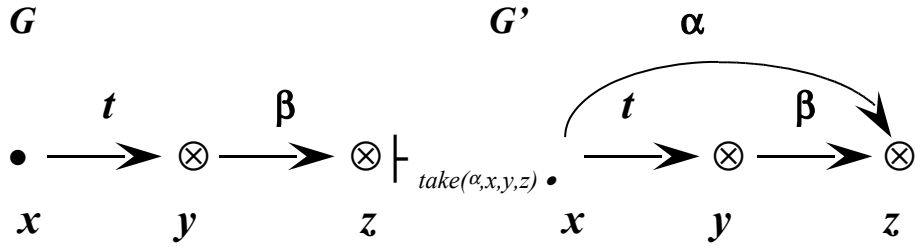


Рисунок 4.2.1. Субъект x берет у объекта y права $\alpha \subseteq \beta$ на объект z

Правило *«Давать» — grant(* α, x, y, z *)*. Пусть $x \in S$, $y, z \in O$ — различные вершины графа G, $\beta \subseteq R$, $\alpha \subseteq \beta$. Правило определяет порядок получения нового графа G' из графа G.

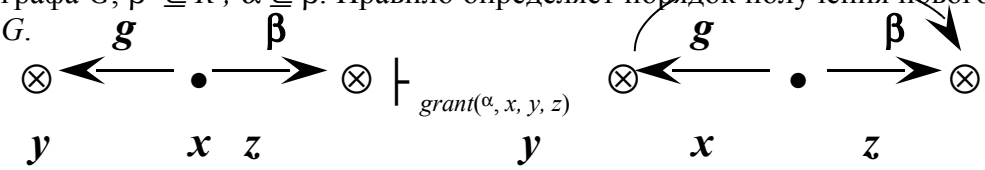


Рисунок 4.2.2. Субъект x дает объекту y права $\alpha \subseteq \beta$ на объект z

Правило *«Создать» — create(β, x, y)*. Пусть $x \in S$, $\beta \subseteq R$, $\beta \neq \varnothing$. Правило определяет порядок получения нового графа *G'* из графа *G*. Где $y \in O$ — новый объект или субъект.

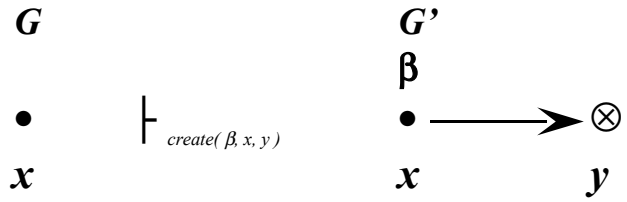


Рисунок 4.2.3. Субъект ***x*** создает новый **β -** доступный объект ***y***

Правило *«Удалить» — remove(α, x, y)*. Пусть $x \in S$, $y \in O$ — различные вершины графа *G*. $\beta \subseteq R$, $\alpha \subseteq \beta$. Правило определяет порядок получения нового графа *G'* из графа *G*.

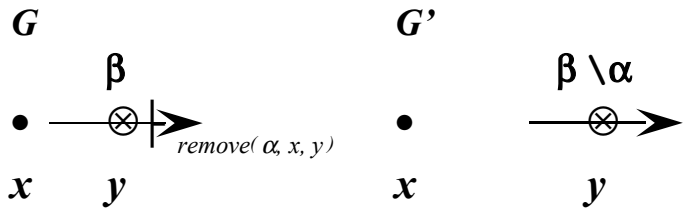


Рисунок 4.2.4. Субъект ***x*** удаляет права доступа **α** на объект ***y***

Перечисленные правила «Брать», «Давать», «Создать», «Удалить» для отличия от правил расширенной модели *Take-Grant* будем называть *де-юре* правилами.

Таблица. *Де-юре* правила модели *Take-Grant*

Правила	Условия	Результирующее состояние $G' = (S', O', E')$
«брать» *take*(α, *x, y, z*)	$x \in S$, $(x, y, t) \in E$, $(y, z, \beta) \in E$, $x \neq z$, $\alpha \subseteq \beta$	$S' = S$, $O' = O$, $E' = E \cup \{ (x, z, \alpha) \}$
«давать» *grant*(α, *x, y, z*)	$x \in S$, $(x, y, g) \in E$, $(x, z, \beta) \in E$, $y \neq z$, $\alpha \subseteq \beta$	$S' = S$, $O' = O$, $E' = E \cup \{ (y, z, \alpha) \}$
«создать» *create*(β, *x, y*)	$x \in S$, $y \notin O$	$O' = O \cup \{ y \}$, $S' = S \cup \{ y \}$, если *y* субъект, $E' = E \cup \{ (x, y, \beta) \}$

«удалить» *remove*(α, *x*, *y*)	$x \in S$, $y \in O$, $(x , z, \beta) \in E$, $\alpha \subseteq \beta$	$S' = S$, $O' = O$, $E' = E \setminus \{ (x , y, \alpha) \}$

В модели *Take-Grant* основное внимание уделяется определению условий, при которых в системе возможно распространение прав доступа определенным способом. Мы рассмотрим условия реализации:

- способа санкционированного получения прав доступа,
- способа похищения прав доступа.

Рассмотрим их подробнее.

Санкционированное получение прав доступа

Данный способ характеризуется тем, что при передачи прав доступа не накладывается ограничений на кооперацию субъектов системы, участвующих в этом процессе.

Пусть $x, y \in O$ — различные объекты графа доступа $G_0 = (S_0, O_0, E_0)$, $\alpha \subseteq R$. Определим предикат ***«возможен доступ»*(α, *x, y,* G_0)** , который будет истинным тогда и только тогда, когда существуют графы $G_1 = (S_1 , O_1 , E_1), \ldots , G_N = (S_N , O_N , E_N)$ такие, что:

$G_0 \vdash_{op1} G_1 \vdash_{op2} \ldots \vdash_{opN} G_N$ и $(x , y , \alpha) \in E_N$.

Определение 1. Говорят, что вершины графа доступов являются *tg* - связными или, что они соединены *tg* -путем, если, без учета направления дуг, в графе между ними существует путь такой, что каждая дуга этого пути помечена *t* или *g*. Будем говорить, что вершины непосредственно *tg* - связны, если *tg* -путь между ними состоит из единственной дуги.

Теорема 1. Пусть $G_0 = (S_0, O_0, E_0)$ граф доступов, содержащий только вершины субъекты. Тогда предикат «возможен доступ» (α, *x, y,* G_0) истинен тогда и только тогда, когда выполняются *условия 1* и *2*:

Условие 1. Существуют субъекты $s_1, \ldots , s_m$:

$(s_i , y, \gamma_i) \in E_0$ для $i = 1, \ldots , m$ и $\alpha = \gamma_1 \cup \ldots \cup \gamma_m$.

Условие 2. Субъект *x* соединен в графе G_0 *tg* - путем с каждым субъектом s_i для $i = 1, \ldots , m$.

Доказательство. Проведем доказательство теоремы для $m = 1$, т.к. схему доказательства для этого случая легко продолжить на случай $m > 1$.

При $m = 1$ *условия 1* и *2* формулируются следующим образом:

Условие 1. Существует субъект *s* такой, что справедливо $(s , y, \alpha) \in E_0$.

Условие 2. Субъекты *x* и *s* соединены *tg* - путем в графе G_0.

Необходимость. Пусть истинен предикат *«возможен доступ»* (α, *x, y,* G_0).

По определению истинности предиката существует последовательность графов доступов $G_1 = (S_1 , O_1 , E_1), \ldots , G_N = (S_N , O_N , E_N)$ такая, что: $G_0 \vdash_{op1} G_1 \vdash_{op2} \ldots \vdash_{opN} G_N$ и $(x , y , \alpha) \in E_N$, при этом *N* является минимальным, т.е. $(x, y, \alpha) \notin E_{N-1}$. Докажем необходимость *условий 1* и *2* индукцией по *N*.

При $N = 0$, очевидно $(x, y, \alpha) \in E_0$. Следовательно, *условия 1, 2* выполнены.

Пусть $N > 0$, и утверждение теоремы истинно для $\forall k < N$. Тогда $(x, y, \alpha) \notin E_0$ и дуга (x, y, α) появляется в графе доступов G_N в результате применения к графу G_{N-1} некоторого правила *opN* . Очевидно, это не правила «Создать» или «Удалить». Если *opN* правило «Брать» («Давать»), то по его определению $\exists\, s' \in S_{N-1} : (x, s', t) \in E_{N-1}$ ($(s', x, g) \in E_{N-1}$), $(s', y, \alpha) \in E_{N-1}$ и $opN = take(\alpha, x, s', y)$ ($opN = grant(\alpha, s', x, y)$).

Возможны два случая $s' \in E_0$ и $s' \notin E_0$.

Пусть $s' \in E_0$, тогда истинен предикат «возможен доступ» (α, s', y, G_0), при этом число преобразований графов $< N$. Следовательно, по предположению индукции $\exists\, s \in S_0$:

(s, y, α) $\in E_0$ и s' соединен с s tg-путем в графе G_0. Кроме этого, истинен предикат «возможен доступ» (t , x, s', G_0) («возможен доступ» (g, s', x, G_0)), при этом число преобразований графов $< N$. Следовательно, по предположению индукции $s'' \in S_0$: (s'', s', t) $\in E_0$ и s'' соединен с x tg-путем в графе G_0 ((s'', x, g) $\in E_0$ и s'' соединен с s' tg-путем в графе G_0). Таким образом, $\exists$ $s \in S_0$: (s, y, α) $\in E_0$ и x, s соединены tg - путем в графе G_0. Выполнение *условий 1* и *2* для случая $s' \in E_0$ доказано.

Пусть $s' \notin E_0$. Заметим, что число преобразований графов N минимально, поэтому новые субъекты создаются только в тех случаях, когда без этого невозможна передача прав доступа. Следовательно, преобразования графов отвечают следующим требованиям:

1. Субъект создатель берет на созданный субъект максимально необходимый набор прав — {t, g};
2. Каждый имеющийся в графе G_0 субъект не создает более одного субъекта;
3. Созданный субъект не создает новых субъектов;
4. Созданный субъект не использует правило «Брать» для получения прав доступа на другие субъекты.

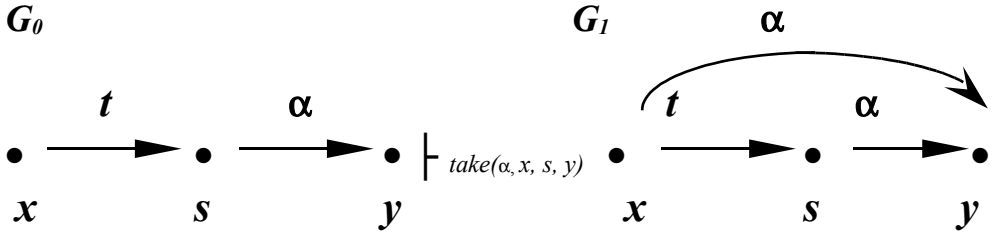
Из перечисленных требований следует, что $\exists M < N$ - 1, $\exists$ $s'' \in S_0$: opM = *create*({ g , t } , s'', s'), opN = *take*(α, x, s', y) и истинен предикат «возможен доступ» (α, s'', y, G_0). Из этого, истинен предикат «возможен доступ»(t, x, s', G_M), а, т.к. s'' единственный субъект в графе G_M, имеющим права на субъект s', то, по предположению индукции, s'' соединен с x tg-путем в графе G_0. Из истинности предиката «возможен доступ» (α, s'', y, G_0) и по предположению индукции, $\exists$ $s \in S_0$: (s, y, α) $\in E_0$ и s'', s соединены tg - путем в графе G_0. Следовательно, $\exists$ $s \in S_0$: (s, y, α) $\in E_0$ и x, s соединены tg - путем в графе G_0. Выполнение *условий 1* и *2* для случая $s' \notin E_0$ доказано. Индуктивный шаг доказан.

Достаточность. Пусть выполнены *условия 1* и *2*. Доказательство проведем индукцией по длине tg - пути, соединяющего субъекты x и s.

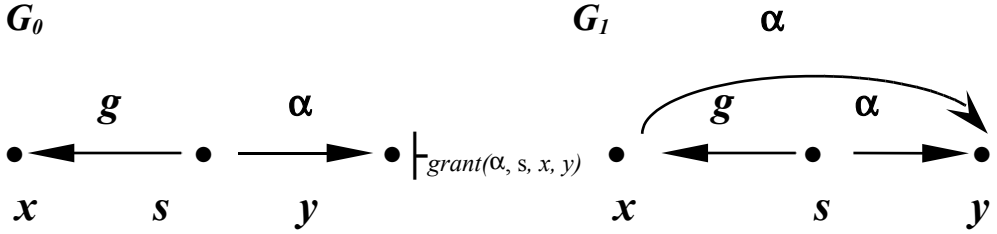
Пусть N = 0. Следовательно, $x = s$, (x, y, α) $\in E_0$ и предикат «возможен доступ» (α, x, y, G_0) истинен.

Пусть N = 1. Т.е. существует $\exists$ (s, y, α) $\in E_0$, и x, s непосредственно tg - связны. Возможны четыре случая такого соединения x и s (см. *рисунок 4.2.5*), для каждого из которых, указана последовательность преобразований графа, требуемая для передачи прав доступа.

1)



2)



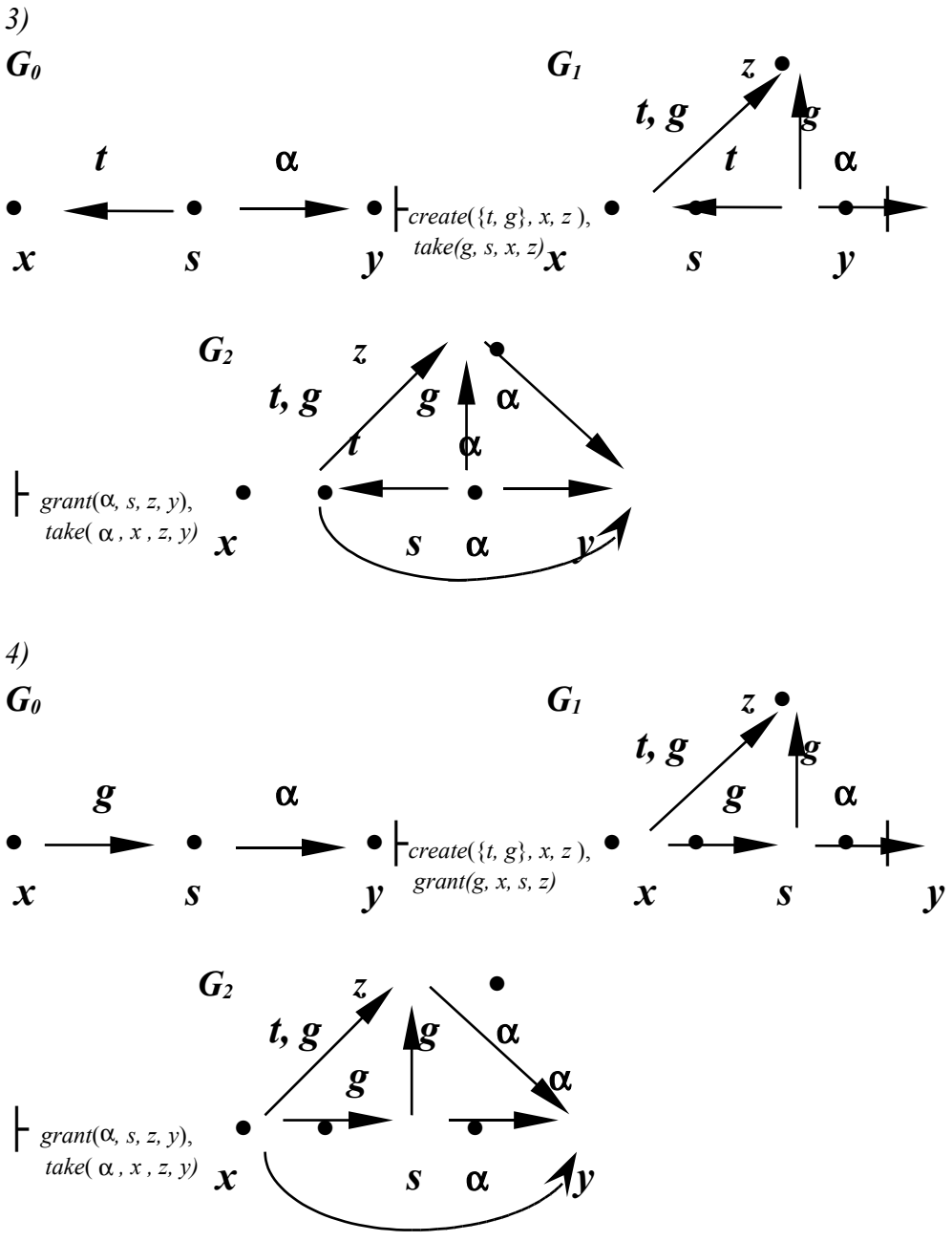


Рисунок 4.2.5. Возможные случаи непосредственной *tg* - связности *x* и *s*

Если длина пути $N > 1$. Пусть вершина *z* находится на *tg* - пути между *x* и *s* и является смежной с *s* в графе G_0. Тогда по доказанному для случая $N = 1$ существует последовательность преобразований графов доступов $G_0 \vdash_{op1} G_1 \vdash_{op2} \ldots \vdash_{opK} G_K : (z, y, \alpha) \in E_k$ и длина *tg* -пути между *z* и *x* равна $N - 1$, что позволяет применить предположение индукции.

Теорема доказана. ■

Для определения истинности предиката «возможен доступ» в произвольном графе необходимо ввести ряд дополнительных понятий.

Определение 2. Островом в произвольном графе доступов G_0 называется его максимальный *tg* -связный подграф, состоящий только из вершин субъектов.

Определение 3. Мостом в графе доступов G_0 называется *tg* - путь, концами которого являются вершины субъекты, словарная запись которого имеет вид:

$\vec{t}^{\,*}$, $\overleftarrow{t}^{\,*}$, $\vec{t}^{\,*}\vec{g}\ \overleftarrow{t}^{\,*}$, $\vec{t}^{\,*}\overleftarrow{g}\ \overleftarrow{t}^{\,*}$, где символ * означает многократное (в том числе нулевое) повторение.

Определение 4. Начальным пролетом моста в графе доступов G_0 называется *tg* - путь, началом которого является вершина субъект, словарная запись которого имеет вид:

$\vec{t}^{\,*}\vec{g}$.

Определение 5. Конечным пролетом моста в графе доступов G_0 называется *tg* - путь, началом которого является вершина субъект, словарная запись которого имеет вид:

$\vec{t}^{\,*}$.

Теорема 2. Пусть $G_0 = (S_0, O_0, E_0)$ произвольный граф доступов. Предикат «возможен доступ»(α, x, y, G_0) истинен тогда и только тогда, когда выполняются *условия 3, 4* и *5* (см. *рисунок 4.2.6.*):

Условие 3. Существуют объекты $s_1, \ldots, s_m$:

$(s_i , y, \gamma_i) \in E_0$ для $i = 1, \ldots, m$ и $\alpha = \gamma_1 \cup \ldots \cup \gamma_m$.

Условие 4. Существуют вершины субъекты $x_1', \ldots, x_m'$ и $s_1', \ldots, s_m'$:

$x = x_i'$ или x_i' соединен с x начальным пролетом моста для $i = 1, \ldots, m$.

$s_i = s_i'$ или s_i' соединен с s_i конечным пролетом моста для $i = 1, \ldots, m$.

Условие 5. Для каждой пары (x_i' , s_i') для $i = 1, \ldots, m$ существуют острова $I_{i,1}, \ldots, I_{i,ui}$, $u_i \geq 1$, такие, что $x_i' \in I_{i,1}$, $s_i' \in I_{i,ui}$, и существуют мосты между островами $I_{i,j}$ и $I_{i,j+1}$, $1 \leq j < u_i$.

Доказательство. Проведем доказательство теоремы для $m = 1$, т.к. схему доказательства для этого случая легко продолжить на случай $m > 1$.

При $m = 1$ *условия 3, 4*, *5* формулируются следующим образом:

Условие 3. $\exists\, s \in O_0 : (s , y, \alpha) \in E_0$.

Условие 4. $\exists\, x' , s' \in E_0$:

1. $x = x'$ или x' соединен с x начальным пролетом моста.
2. $s = s'$ или s' соединен с s конечным пролетом моста.

Условие 5. Существуют острова $I_1 , \ldots , I_u$, $u \geq 1$, такие, что $x' \in I_1$, $s' \in I_u$, и существуют мосты между островами I_j и I_{j+1} для $j = 1, \ldots , u\text{-}1$.

Необходимость. Пусть истинен предикат «возможен доступ» (α, *x, y,* G_0). По определению истинности предиката существует последовательность графов доступов $G_1 = (S_1 , O_1 , E_1), \ldots , G_N = (S_N , O_N , E_N)$ такая, что: $G_0 \vdash_{op1} G_1 \vdash_{op2} \ldots \vdash_{opN} G_N$ и $(x , y , \alpha) \in E_N$, при этом N является минимальным, т.е. $(x, y, \alpha) \notin E_{N-1}$. Докажем необходимость *условий 3, 4, 5* индукцией по N.

При $N = 0$, очевидно $(x, y, \alpha) \in E_0$. Следовательно, *условия 3, 4, 5* выполнены.

Пусть $N > 0$, и утверждение теоремы истинно для $\forall k < N$. Тогда $(x, y, \alpha) \notin E_0$ и дуга (x, y, α) появляется в графе доступов G_N в результате применения к графу G_{N-1} некоторого правила *opN* . Возможны два случая $x \notin S_0$ и $x \in S_0$.

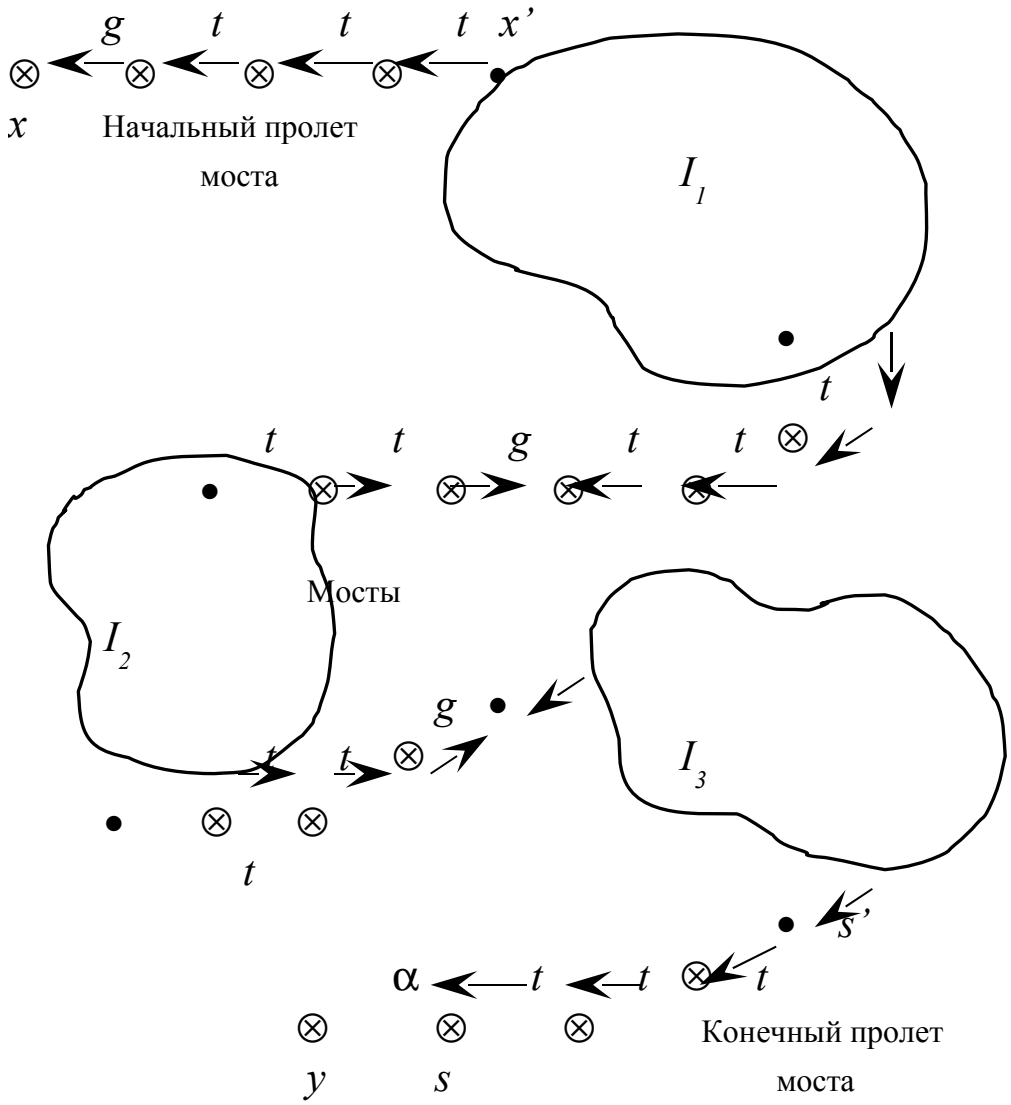
Если $x \notin S_0$, то $\exists\, x_1 \in S_{N-1} : opN = grant(\alpha, x_1, x, y)$. С учетом минимальности N и замечаний, сделанных при доказательстве *теоремы 1*, можно считать, что $x_1 \in S_0$. Следовательно:

- Истинен предикат «возможен доступ» (g, x_1, x, G_0) с числом преобразований графов $< N$. Тогда по предположению индукции выполнены *условия 3, 4, 5*:
  1. $\exists\, x_2 \in O_0 : (x_2 , x, \mathrm{g}) \in E_0$;
  2. $\exists\, x' \in E_0$, соединенный с x_2 конечным пролетом моста;
  3. Существуют острова $I_1 , \ldots , I_t$, $t \geq 1$, такие, что $x_1 \in I_t$, $x' \in I_1$, и существуют мосты между островами I_j и I_{j+1} для $j = 1, \ldots , t\text{-}1$;
- Истинен предикат «возможен доступ» (α, x_1, y, G_0) с числом преобразований графов $< N$. Тогда по предположению индукции выполнены *условия 3, 4, 5*:

1. $s \in O_0 : (s, y, \alpha) \in E_0$;
2. $\exists\, s' \in E_0 : s = s'$ или s' соединен с s конечным пролетом моста;
3. Существуют острова I_t , ... , I_u , t-$u \geq 1$, такие, что $x_1 \in I_t$, $s' \in I_u$, и существуют мосты между островами I_j и I_{j+1} для $j = t, \ldots, u\text{-}1$.

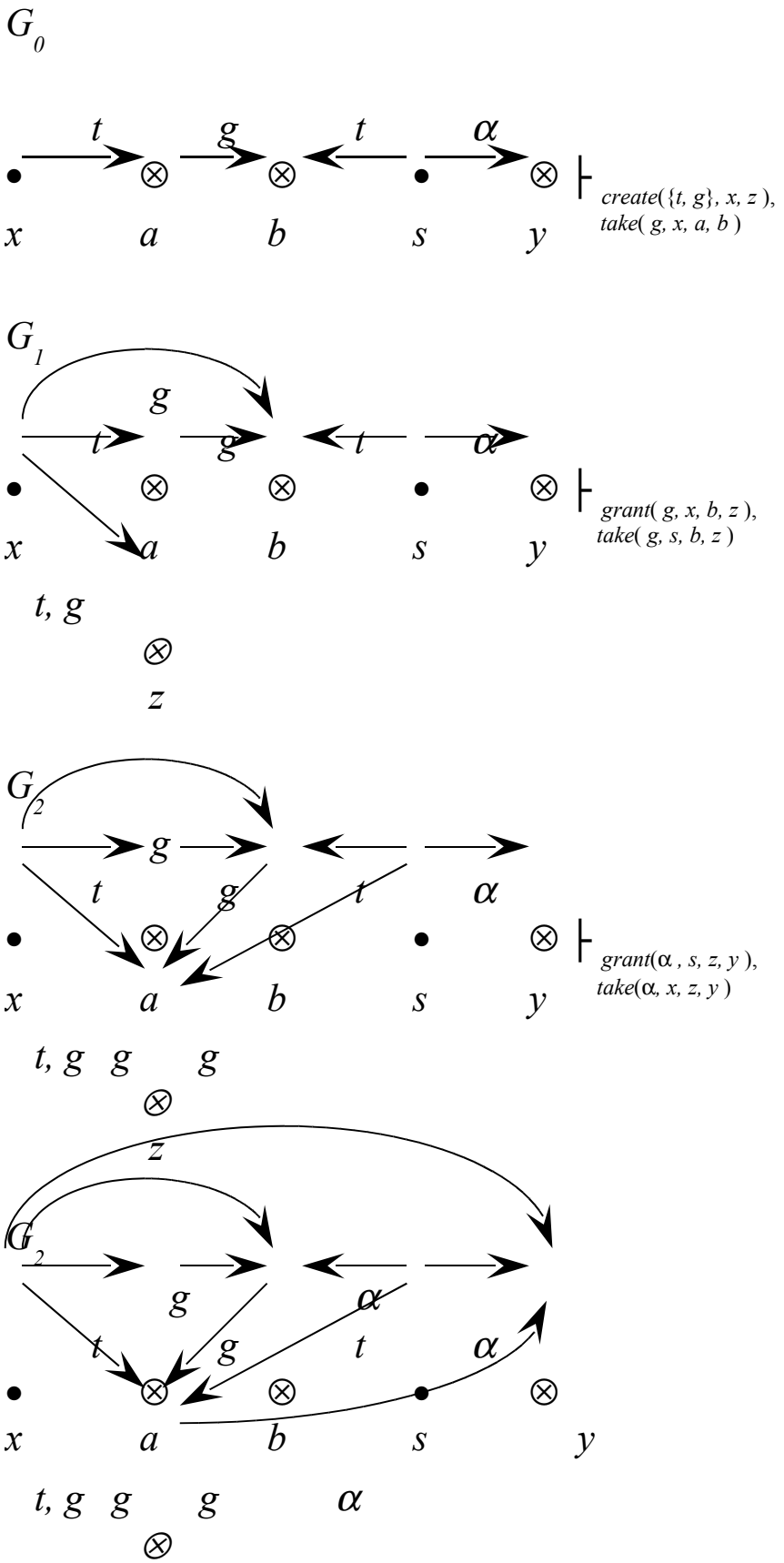
Заметим, что путь, соединяющий вершины x', x_2, x, есть начальный пролет моста. Таким образом, для случая $x \notin S_0$ *условия 3, 4, 5* выполняются, и индуктивный шаг доказан.

Если $x \in S_0$, то *п.1 условия 4* очевидно выполняется. Многократно применяя технику доказательства, использованную выше, можно доказать индуктивный шаг и в данном случае.



Достаточность. Условия 3, 4, 5 конструктивны.

По *условию 3* существует объект *s*, который обладает правами α на объект *y*. По *п. 2 условия 4* существует субъект *s'*, который, либо совпадает с *s*, либо по конечному пролету моста может забрать у субъекта *s* права α на объект *y*.



По *теореме 1* права доступа, полученные одним субъектом, принадлежащим острову, могут быть переданы любому другому субъекту острова. По *условию 5* между островами существуют мосты, по которым возможна передача прав доступа. В качестве примера на *рисунке 4.2.7* разобрана последовательность преобразований графа доступов при передаче прав по мосту вида $\vec{t}\ \vec{g}\ \overleftarrow{t}$. По *n*.1 *условия 4* существует субъект *x'*, который или совпадает с *x*, или, получив права доступа, может передать их *x* по начальному пролету моста.

Теорема доказана. ■

Возможность похищения прав доступа

Способ передачи прав доступа, рассмотренный в предыдущей главе, предполагает идеальное сотрудничество субъектов. В случае похищения прав доступа предполагается, что передача прав доступа на объект осуществляется без содействия субъекта, изначально обладавшего передаваемыми правами.

Пусть $x, y \in O$ — различные объекты графа доступа $G_0 = (S_0, O_0, E_0)$, $\alpha \subseteq R$. Определим предикат ***«возможно похищение»*(α, *x, y, G₀*)** , который будет истинным тогда и только тогда, когда $(x, y, \alpha) \notin E_0$ и существуют графы $G_1 = (S_1, O_1, E_1), \ldots, G_N = (S_N, O_N, E_N)$ такие, что:

$G_0 \vdash_{op1} G_1 \vdash_{op2} \ldots \vdash_{opN} G_N$ и $(x, y, \alpha) \in E_N$, при этом, если $\exists(s, y, \alpha) \in E_0$, то $\forall z \in S_j, j = 0, 1, \ldots, N$ выполняется $opK \neq grant(\alpha, s, z, y)$, $K = 1, \ldots, N$.

Теорема 3. Пусть $G_0 = (S_0, O_0, E_0)$ произвольный граф доступов. Предикат «возможно похищение»(α, x, y, G_0) истинен тогда и только тогда, когда выполняются *условия 6, 7, 8*.

Условие 6. $(x, y, \alpha) \notin E_0$.

Условие 7. Существуют объекты $s_1, \ldots, s_m$:

$(s_i, y, \gamma_i) \in E_0$ для $i = 1, \ldots, m$ и $\alpha = \gamma_1 \cup \ldots \cup \gamma_m$.

Условие 8. Являются истинными предикаты «возможен доступ» (t, x, s, G_0) для $i = 1, \ldots, m$.

Доказательство. Аналогично доказательству *теоремы 2*. ■

Расширенная модель *Take-Grant*

В расширенной модели *Take-Grant* (см. [8]) рассматриваются пути и стоимости возникновения информационных потоков в системах с дискреционным разграничением доступа.

В классической модели *Take-Grant* по существу рассматриваются два права доступа: *t, g,* и четыре правила (правила *де-юре*) преобразования графа доступов: *take, grant, create, remove*. В расширенной модели дополнительно рассматриваются два права доступа на чтение *r* (*read*), на запись *w* (*write*) и шесть правил (правила *де-факто*) преобразования графа доступов: *post, spy, find, pass* и два правила без названия.

Де-факто правила служат для поиска путей возникновения возможных информационных потоков в системе. Эти правила являются следствием уже имеющихся у объектов системы прав доступа и могут явиться причиной возникновения информационного потока от одного объекта к другому без их непосредственного взаимодействия.

В результате применения к графу доступов *де-факто* правил в него добавляются мнимые дуги, помечаемые *r* или *w* и изображаемые пунктиром (см. *рисунок 4.2.8*). Вмести с дугами графа, соответствующими правам доступа *r* и *w*, мнимые дуги указывают на направления информационных каналов в системе.

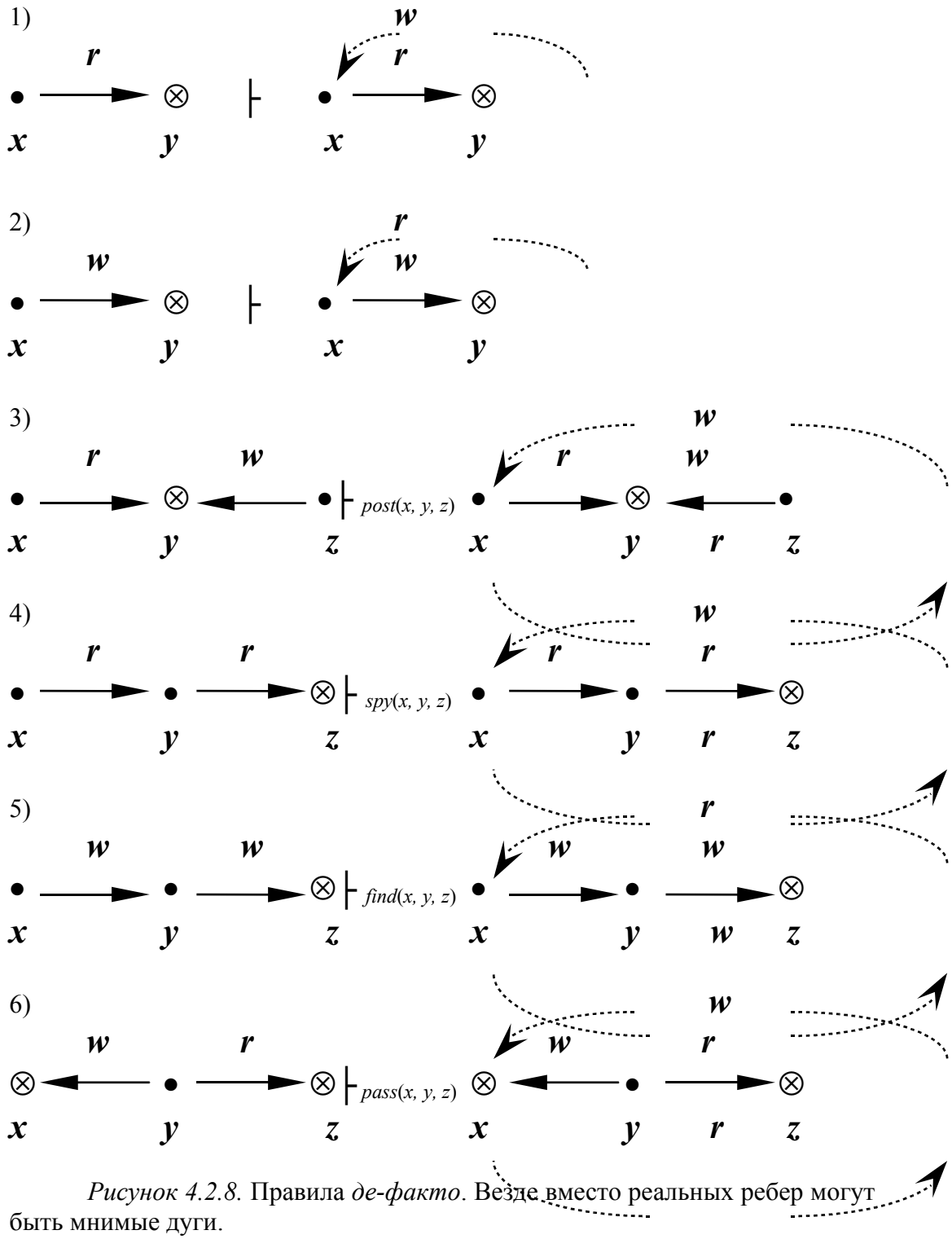


Рисунок 4.2.8. Правила *де-факто*. Везде вместо реальных ребер могут быть мнимые дуги.

Важно отметить, что к мнимым дугам нельзя применять *де-юре* правила преобразования графа доступов. Информационные каналы нельзя брать или передавать другим объектам системы.

Чтобы пояснить смысл *де-факто* правил рассмотрим ряд примеров.

Пример 1. Пусть субъект x не имеет право r на объект z, но имеет это право на субъект y. Пусть кроме этого x имеет право r на y. Тогда x может, просматривая информацию в y, пытаться искать в нем информацию из z. Таким образом, в системе может возникнуть информационный канал от объекта z к субъекту x, что демонстрирует *де-факто* правило

spy. Очевидно также, если бы *y* был объектом, т.е. пассивным элементом системы, то информационный канал от *z* к *x* возникнуть не мог.

Пример 2. Пусть субъект *y* имеет право *r* на объект *z* и право *w* на объект *x*. Прочитанная субъектом *y* информация в *z* может быть записана в *x*. Следовательно, в системе может возникнуть информационный канал от объекта *z* к объекту *x*, что демонстрирует *де-факто* правило *pass*.

Проблемы взаимодействия — центральный вопрос при похищении прав доступа. Мы коснемся их только в постановочном плане.

Каждое *де-юре* правило требует для достижения своей цели участие одного субъекта, а для реализации *де-факто* правила необходимы один или два субъекта. Например, в *де-факто* правилах *post, spy, find* обязательно взаимодействие двух субъектов. Желательно во множестве всех субъектов выделить подмножество так называемых субъектов-заговорщиков — участников процессов передачи прав или информации. В небольших системах эта задача легко решаема. Многократно просматривая граф доступов и применяя к нему все возможные *де-юре* и *де-факто* правила, можно найти замыкание графа доступов, которое будет содержать дуги, соответствующие всем информационным каналам системы. Однако, если граф доступов большой, то найти его замыкание весьма сложно.

Можно рассмотреть проблему поиска и анализа информационных каналов в ином свете. Допустим, факт нежелательной передачи прав или информации уже состоялся. Каков наиболее вероятный путь его осуществления? В классической модели *Take-Grant* не дается прямого ответа на этот вопрос. Мы можем говорить, что есть возможность передачи прав или информации, но не можем определить, какой из путей при этом использовался.

Предположим, что чем больше узлов на пути между вершинами, по которому произошла передача прав доступа или возник информационный поток, тем меньше вероятность использования этого пути. Например, на *рисунке 4.2.9.* видно, что интуитивно наиболее вероятный путь передачи информации от субъекта ***z*** к субъекту ***x*** лежит через объект ***y***. В тоже время злоумышленник для большей скрытности может специально использовать более длинный путь.

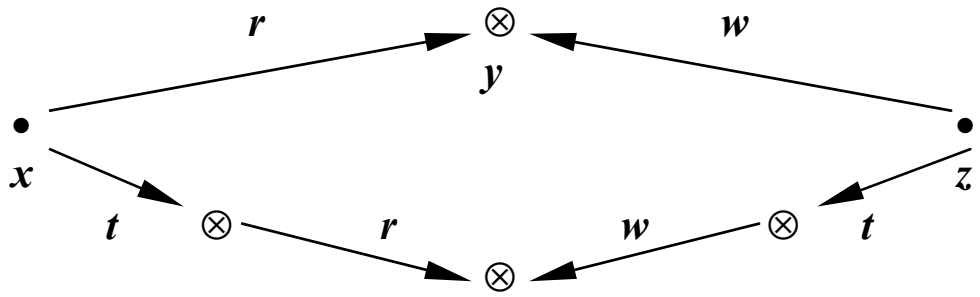


Рисунок 4.2.9. Пути возникновения информационного канала от ***z*** к ***x***.

Таким образом, в расширенную модель *Take-Grant* можно включить понятие вероятности или стоимости пути передачи прав или информации. Путям меньшей стоимости соответствует наивысшая вероятность и их надо исследовать в первую очередь. Есть два основных подхода определения стоимости путей:

1. Подход, основанный на присваивание стоимости каждой дуге на пути в графе доступов. В этом случае стоимость дуги определяется в зависимости от прав доступа, которыми она помечена, а стоимость пути есть сумма стоимостей пройденных дуг;
2. Подход, основанный на присваивание стоимости каждому используемому *де-юре* или *де-факто* правилу. Стоимость правила при этом может быть выбрана, исходя из сферы применения *Take-Grant* системы, и являться:

- константой,
- зависеть от специфики правила,
- зависеть от числа участников при применении правила,
- зависеть от степени требуемого взаимодействия объектов.

Стоимость пути в этом случае определяется как сумма стоимостей примененных правил.

Выводы по модели распространения прав доступа *Take-Grant*

Модель *Take-Grant* служит для анализа систем защиты с дискреционной политикой безопасности. В модели определены условия, при которых происходит передача или похищение прав доступа. Однако на практике редко возникает необходимость в использовании указанных условий, т.к. при анализе большинства реальных систем защиты не возникают столь сложные по взаимосвязи объектов графы доступов. А сами правила *take* и *grant* сравнительно редко используются на практике.

В тоже время наиболее часто в реальных системах субъекты используют права доступа на чтение и запись. Поэтому предложенные в расширенной модели *Take-Grant* подходы к поиску и анализу путей возникновения в системе информационных каналов, определению их стоимости представляются наиболее интересными и актуальными.

Модель системы безопасности *Bell-LaPadula*

Основные положения классической модели *BL*

Модель *Bell-LaPadula* (*BL*) построена для анализа систем защиты, реализующих мандатное (полномочное) разграничение доступа. Возможность ее использования в качестве формальной модели таких систем непосредственно отмечена в критерии *TCSEC* («Оранжевая книга»).

Модель *BL* была предложена *D. Bell, L. LaPadula* в 1975 году. Однако ее полное описание до сих пор является недоступным, поэтому мы приведем его для классической модели *BL* в урезанном виде по работам [2] и [16].

Пусть определены конечные множества:

S — множество субъектов системы (например, пользователи системы и программы);

O — множество объектов системы (например, все системные файлы);

$R = \{read, write, append, execute\}$ — множество видов доступа субъектов из S к объектам из O, где *read* — доступ на чтение, *write* — на запись, *append* — на запись в конец объекта, *execute* — на выполнение.

Обозначим через:

$B = \{b \subseteq S \times O \times R\}$ — множество возможных множеств текущих доступов в системе;

M — матрица разрешенных доступов, где $M_{so} \in R$ - разрешенный доступ субъекта s к объекту o;

L — множество уровней секретности, например, $L = \{U, C, S, TS\}$, где $U < C < S < TS$;

$(f_s, f_o, f_c) \in F = L^s \times L^o \times L^s$ — тройка функций (f_s, f_o, f_c), определяющих

$f_s : S \to L$ — уровень допуска субъекта,

$f_o : O \to L$ — уровень секретности объекта,

$f_c : S \to L$ — текущий уровень допуска субъекта, при этом $\forall s \in S\, f_c(s) \le f_s(s)$;

H — текущий уровень иерархии объектов (далее не рассматривается).

$V = B \times M \times F \times H$ — множество состояний системы.

Q — множество запросов системе;

D — множество решений по запросам, например $\{yes, no, error\}$;

$W \subseteq Q \times D \times V \times V$ — множество действий системы, где четверка $(q, d, v_2, v_1) \in W$ означает, что система по запросу q с ответом d перешла из состояния v_1 в состояние v_2;

N_0 — множество значений времени ($N_0 = 0, 1, 2, \ldots$);

X — множество функций x: $N_0 \to Q$, задающих все возможные последовательности запросов к системе;

Y — множество функций y: $N_0 \to D$, задающих все возможные последовательности ответов системы по запросам;

Z — множество функций z: $N_0 \to V$, задающих все возможные последовательности состояний системы.

Определение 1. $\Sigma(Q, D, W, z_0) \subseteq X \times Y \times Z$ называется системой, если выполняется $(x, y, z) \in \Sigma(Q, D, W, z_0)$ тогда и только тогда, когда для каждого $t \in N_0$ $(x_t, y_t, z_{t+1}, z_t) \in W$, где z_0 начальное состояние системы. При этом каждый набор $(x, y, z) \in \Sigma(Q, D, W, z_0)$ называется реализацией системы, а $(x_t, y_t, z_{t+1}, z_t) \in W$ называется действием системы $\forall t \in N_0$.

Безопасность системы определяется с помощью трех свойств:

- *ss* — свойства простой безопасности (*simple security*),
- * — свойства звезда,
- *ds* — свойства дискретной безопасности (*discretionary security*).

Определение 2. Доступ $(s, o, r) \in S \times O \times R$ обладает *ss* - свойством относительно $f = (f_s, f_o, f_c) \in F$, если выполняется одно из условий:

- $r \in \{execute, append\}$,
- $r \in \{read, write\}$ и $f_s(s) \geq f_o(o)$.

Определение 3. Состояние системы $(b, M, f, h) \in V$ обладает *ss* - свойством, если каждый элемент $(s, o, r) \in b$ обладает *ss* - свойством относительно *f*.

Определение 4. Доступ $(s, o, r) \in S \times O \times R$ удовлетворяет * - свойству относительно $f = (f_s, f_o, f_c) \in F$, если выполняется одно из условий:

- $r = execute$,
- $r = append$ и $f_o(o) \geq f_c(s)$,
- $r = read$ и $f_c(s) \geq f_o(o)$,
- $r = write$ и $f_c(s) = f_o(o)$.

Определение 5. Состояние системы $(b, M, f, h) \in V$ обладает * - свойством, если каждый элемент $(s, o, r) \in b$ обладает * - свойством относительно *f*.

Определение 6. Состояние системы $(b, M, f, h) \in V$ обладает * - свойством, относительно подмножества $S' \subseteq S$, если каждый элемент $(s, o, r) \in b$, где $s \in S'$ обладает * - свойством относительно *f*. При этом $S \setminus S'$ называются множеством доверенных субъектов, т.е. субъектов, имеющих право нарушать политику безопасности.

Определение 7. Состояние системы $(b, M, f, h) \in V$ обладает *ds* - свойством, если для каждого элемента $(s, o, r) \in b$ выполняется $r \in M_{so}$.

Определение 8. Состояние системы (b, M, f, h) называется безопасным, если обладает * - свойством относительно S', *ss* - свойством и *ds* - свойством.

Определение 9. Реализация системы $(x, y, z) \in \Sigma(Q, D, W, z_0)$ обладает *ss* - свойством (* - свойством, *ds* - свойством), если в последовательности $(z_0, z_1, \ldots)$ каждое состояние обладает *ss* - свойством (* - свойством, *ds* - свойством).

Определение 10. Система $\Sigma(Q, D, W, z_0)$ обладает *ss* - свойством (* - свойством, *ds* - свойством), если каждая ее реализация обладает *ss* - свойством (* - свойством, *ds* - свойством).

Определение 11. Система $\Sigma(Q, D, W, z_0)$ называется безопасной, если она обладает *ss* - свойством, * - свойством, *ds* - свойством одновременно.

Прокомментируем введенные выше свойства безопасности системы.

Во-первых, из обладания доступом * - свойством относительно *f* следует обладание этим доступом *ss* - свойством относительно *f*.

Во-вторых, из обладания системой *ss* - свойством следует, что в модели *BL* выполняется запрет на чтение вверх, принятый в мандатной (полномочной) политике безопасности. Кроме того, *ss* - свойство не допускает модификацию с использованием доступа *write*, если $f_s(s) < f_o(o)$. Таким образом, функция $f_s(s)$ определяет для субъекта *s* верхний уровень секретности объектов, к которым он в принципе может получить доступ *read* или *write*.

В-третьих, поясним * - свойство. Если субъект *s* может понизить свой текущий доступ до $f_c(s) = f_o(o)$, то он может получить доступ *write* на объект *o*, но не доступ *read* на объекты *o'*, чей уровень $f_o(o') > f_c(s)$. Хотя при этом, возможно, выполняется $f_s(s) \geq f_o(o')$, и в каких-то других состояниях системы субъект *s* может получить доступ *read* на объект *o'*. Таким образом, * - свойство исключает появление в системе канала утечки информации сверху вниз (см. пример на *рисунке 4.2.13*) и соответствует требованиям мандатной (полномочной) политики безопасности.

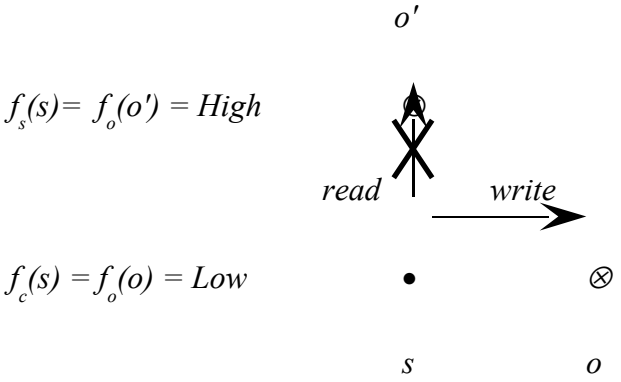


Рисунок 4.2.13. Иллюстрация к * - свойству.

Проверка безопасности системы по определению в большинстве случаев не может быть реализована на практике в связи тем, что при этом требуется проверка безопасности всех реализаций системы, а их бесконечно много. Следовательно, необходимо определить и обосновать условия иные условия безопасности системы, которые можно проверять на практике. В классической модели *BL* эти условия определяются для множества действий системы *W*.

Теорема *A*1. Система $\Sigma(Q, D, W, z_0)$ обладает *ss* - свойством для любого начального состояния z_0, обладающего *ss* - свойством, тогда и только тогда, когда множество действий системы *W* удовлетворяет условиям:

$\forall (q, d, (b^*, M^*, f^*, h^*), (b, M, f, h)) \in W$

Условие 1. $\forall (s, o, r) \in b^* \setminus b$ обладает *ss* - свойством относительно f^*;

Условие 2. Если $(s, o, r) \in b$ и не обладает *ss* - свойством относительно f^*, то $(s, o, r) \notin b^*$.

Доказательство.

Достаточность. Пусть выполнены *условия 1* и *2* и пусть $(x, y, z) \in \Sigma(Q, D, W, z_0)$ произвольная реализация системы. Тогда $(x_t, y_t, (b_{t+1}, M_{t+1}, f_{t+1}, h_{t+1}), (b_t, M_t, f_t, h_t)) \in W$, где $z_{t+1} = (b_{t+1}, M_{t+1}, f_{t+1}, h_{t+1})$, $z_t = (b_t, M_t, f_t, h_t)$ для $t \in N_0$.

$\forall (s, o, r) \in b_{t+1}$ выполняется или $(s, o, r) \in b_{t+1} \setminus b_t$, или $(s, o, r) \in b_t$. Из *условия 1* следует, что состояние системы z_{t+1} пополнилось доступами, которые обладают *ss* - свойством относительно f^*. Из *условия 2* следует, что доступы из b_t, которые не обладают *ss* - свойством относительно f^*, не входят в b_{t+1}. Следовательно, $\forall (s, o, r) \in b_{t+1}$ обладают *ss* - свойством относительно f^* и по определению состояние z_{t+1} обладает *ss* - свойством, для $t \in N_0$. Т.к. по условию и состояние z_0 обладает *ss* - свойством, то выбранная нами произвольная реализация (x, y, z) также обладает *ss* - свойством. Достаточность доказана.

Необходимость. Пусть система $\Sigma(Q, D, W, z_0)$ обладает *ss* - свойством. Будем считать, что в множество W входят только те действия системы, которые используются в ее реализациях. Тогда $\forall(q, d, (b^*, M^*, f^*, h^*), (b, M, f, h)) \in W \ \exists(x, y, z) \in \Sigma(Q, D, W, z_0)$ и $\exists t \in N_0 : (q, d, (b^*, M^*, f^*, h^*), (b, M, f, h)) = (x_t, y_t, z_{t+1}, z_t)$. Т.к. реализация системы (x, y, z) обладает *ss* - свойством, то и состояние $z_{t+1} = (b^*, M^*, f^*, h^*)$ обладает *ss* - свойством по определению. Следовательно, *условия 1* и *2* очевидно выполняются. Необходимость доказана. ■

Теорема *A*2. Система $\Sigma(Q, D, W, z_0)$ обладает * - свойством относительно $S' \subseteq S$ для любого начального состояния z_0, обладающего * - свойством относительно *S'*, тогда и только тогда, когда множество действий системы *W* удовлетворяет условиям:

$\forall (q, d, (b^*, M^*, f^*, h^*), (b, M, f, h)) \in W$

Условие 1. $\forall s \in S'$, $\forall (s, o, r) \in b^* \setminus b$ обладает * - свойством относительно f^*;

Условие 2. $\forall s \in S'$, если $(s, o, r) \in b$ и не обладает * - свойством относительно f^*, то $(s, o, r) \notin b^*$.

Доказательство. Аналогично доказательству *теоремы A1*. ■

Теорема *A*3. Система $\Sigma(Q, D, W, z_0)$ обладает *ds* - свойством для любого начального состояния z_0, обладающего *ds* - свойством, тогда и только тогда, когда множество действий системы *W* удовлетворяет условиям:

$\forall (q, d, (b^*, M^*, f^*, h^*), (b, M, f, h)) \in W$

Условие 1. $\forall (s, o, r) \in b^* \setminus b$, выполняется $r \in m_{so}^*$;

Условие 2. Если $(s, o, r) \in b$ и $r \notin m_{so}^*$, то $(s, o, r) \notin b^*$.

Доказательство. Аналогично доказательству *теоремы A1*. ■

Теорема *BST* (*Basic Security Theorem*). Система $\Sigma(Q, D, W, z_0)$ безопасна тогда и только тогда, когда z_0 безопасное состояние и множество действий системы W удовлетворяет условиям *теорем A1, A2, A3*.

Доказательство. Теорема BST следует из *теорем A1, A2, A3*. ■

Описанная выше классическая модель *BL* предоставляет общий подход к построению систем, реализующих мандатную (полномочную) политику безопасности. В модели *BL* определяется, какими свойствам должны обладать состояния и действия системы, что она была безопасной согласно данному *определению 11*. В тоже время в модели не указывается конкретно, что должна делать система по запросам на доступ субъектов к объектам при переходе из состояния в состояние, как в точности должны при этом изменяться значения элементов модели.

Пример 1. Воспользуемся *рисунком 4.2.13*. Пусть субъект *s* запрашивает доступ *read* на объект *o'*. В данной ситуации система может выбрать один из двух возможных путей:

1. Запретить субъекту *s* запрашиваемый им доступ *read* на объект *o'*.
2. Закрыть доступ *write* субъекта *s* на объект *o*. Повысить текущий уровень секретности $f_c(s)$ до *High*. Разрешить субъекту *s* запрашиваемый им доступ *read* на объект *o'*.

Каждый из описанных путей соответствует требованиям безопасности модели *BL*.

В реальных системах возможны более сложные ситуации, чем ситуация, описанная в *примере 1*. Кроме того, возможно использование в системе каких-то других видов доступа субъектов к объектам, которые потребуют доопределения свойств безопасности, что не всегда легко сделать. В связи с этим большое значение имеет корректное определение свойств безопасности.

Ниже по работе [16] приводится пример некорректного определения безопасности в модели *BL*, где вместо * - свойства используется абсурдное с точки зрения здравого смысла ♣ - свойство. Однако при этом не возникает никаких противоречий в логике доказательства теорем, определяющих условия безопасности системы.

Пример некорректного определения безопасности в модели *BL*

Определение 12. Доступ $(s, o, r) \in S \times O \times R$ удовлетворяет ♣ - свойству относительно $f = (f_s, f_o, f_c) \in F$, если выполняется одно из условий:

- $r \in \{read, append, execute\}$,
- $r = write$ и $f_c(s) \geq f_o(o)$.

Определение 13. Состояние системы $(b, M, f, h) \in V$ обладает ♣ - свойством, если каждый элемент $(s, o, r) \in b$ обладает ♣ - свойством относительно f.

Определение 14. Состояние системы $(b, M, f, h) \in V$ называется ♣ - безопасным, если обладает *ss* - свойством, ♣ - свойством, *ds* - свойством одновременно.

Определение 15. Реализация (x, y, z) системы $\Sigma(Q, D, W, z_0)$ обладает ♣ - свойством, если в последовательности $(z_0, z_1, \ldots)$ каждое состояние обладает ♣ - свойством.

Определение 16. Система $\Sigma(Q, D, W, z_0)$ обладает ♣ - свойством, если каждая ее реализация обладает ♣ - свойством.

Определение 17. Система $\Sigma(Q, D, W, z_0)$ называется ♣ - безопасной, если она обладает *ss* - свойством, ♣ - свойством, *ds* - свойством одновременно.

Теорема *A2*♣. Система $\Sigma(Q, D, W, z_0)$ обладает ♣ - свойством для любого начального состояния z_0, обладающего ♣ - свойством, тогда и только тогда, когда множество действий системы W удовлетворяет условиям:

$\forall (q, d, (b^*, M^*, f^*, h^*), (b, M, f, h)) \in W$

Условие 1. $\forall (s, o, r) \in b^* \setminus b$ обладает ♣ - свойством относительно f^*;

Условие 2. Если $(s, o, r) \in b$ и не обладает ♣ - свойством относительно f^*, то $(s, o, r) \notin b^*$.

Доказательство. Аналогично доказательству *теоремы A1*. ■

Теорема *BST*♣. Система $\Sigma(Q, D, W, z_0)$ ♣ - безопасна тогда и только тогда, когда z_0 ♣ - безопасное состояние и множество действий системы W удовлетворяет условиям *теорем A1, A2*♣*, A3*.

Доказательство. Теорема BST♣ следует из *теорем A1, A2*♣*, A3*. ■

Эквивалентные подходы к определению безопасности в модели *BL*

В зависимости от условий практического использования модели или в целях ее дальнейшего исследования возможно использование эквивалентных подходов к определению свойств безопасности. По работе [18] мы приведем два подхода, изменяющих значения отдельных элементов модели, но не меняющих ее по сути.

При рассмотрении этих подходов положим $R = \{read, write\}$ и $\forall s \in S \ f_s(s) = f_c(s)$. Исключим из рассмотрения матрицу доступов M и множество ответов системы D. Вместо множества действий системы W используем функцию переходов $T(q, v) = v^*$ — переход из состояния v по команде q в состояние v^* и будем обозначать систему через $\Sigma(V, T, z_0)$.

Подход Read-Write (*RW*)

Переопределим *ss* - свойство и * - свойство. Т.к. основные ограничения на доступ *write* следуют из * - свойства, то в *ss* - свойстве зададим ограничения только на *read*.

Определение 18. Доступ $(s, o, r) \in b$ обладает *ss* - свойством, если выполняется одно из условий:

- $r = write$,
- $r = read$ и $f_s(s) \geq f_o(o)$.

Определение 19. Доступ $(s, x, r) \in b$ обладает * - свойством, если выполняется одно из условий:

- $r = read$ и не существует $y \in O$: $(s, y, write) \in b$ и $f_o(x) > f_o(y)$,
- $r = write$ и не существует $y \in$ O: $(s, y, read) \in b$ и $f_o(y) > f_o(x)$.

Заметим особо, что в * - свойстве не рассматривается уровень доступа субъекта посредника *s*. В этом нет необходимости, т.к., если требовать выполнения * - свойства и *ss*

- свойства одновременно и считать, что субъект не может накапливать в себе информацию, то не возникает противоречий по существу с положениями мандатной (полномочной) политики безопасности. Субъект может читать информацию из объектов с уровнем секретности не выше его уровня доступа, и при этом субъект не может стать каналом утечки информации сверху вниз.

По аналогии со свойствами классической модели *BL* определяются *ss* - свойства и * - свойства для состояния, реализации и системы в целом.

Теорема *A1-RW*. Система $\Sigma(V, T, z_0)$ обладает *ss* - свойством для любого начального состояния z_0, обладающего *ss* - свойством, тогда и только тогда, когда функция переходов $T(q, v) = v^*$ удовлетворяет условиям:

Условие 1. Если $(s, o, read) \in b^* \setminus b$, то $f_s^*(s) \geq f_o^*(o)$.

Условие 2. Если $(s, o, read) \in b$ и $f_s^*(s) < f_o^*(o)$, то $(s, o, read) \notin b^*$.

Доказательство. Аналогично доказательству *теоремы A1*. ■

Теорема *A2-RW*. Система $\Sigma(V, T, z_0)$ обладает * - свойством для любого начального состояния z_0, обладающего * - свойством, тогда и только тогда, когда функция переходов $T(q, v) = v^*$ удовлетворяет условиям:

Условие 1. Если $\{(s, x, read), (s, y, write)\} \subseteq b^* \setminus b$, то $f_o^*(y) \geq f_o^*(x)$.

Условие 2. Если $\{(s, x, read), (s, y, write)\} \subseteq b$ и $f_o^*(y) < f_o^*(x)$, то $\{(s, x, read), (s, y, write)\} \not\subset b^*$.

Доказательство. Аналогично доказательству *теоремы A1*. ■

Теорема *BST-RW*. Система $\Sigma(V, T, z_0)$ безопасна для безопасного начального состояния z_0, тогда и только тогда, когда выполнены условия *теоремы A1-RW* и *теоремы A2-RW*.

Доказательство. *Теорема BST-RW* следует из *теорем A1-RW, A2-RW* ■

В данном подходе * - свойство определено таким образом, что его смысл — предотвращение возникновения информационных каналов сверху вниз становится более ясным, чем при использовании функции f_c в классической модели *BL*. В этом заключена основная ценность *подхода Read-Write*.

Подход Transaction (*T*)

В данном подходе используются определения безопасного состояния, реализации и системы *подхода Read-Write*.

Однако в *подходе Transaction* основной акцент делается на определении свойств безопасности функции переходов *T*. При этом на *T* накладывается дополнительное ограничение: за один шаг работы системы вносится только одно изменение в один из параметров. Т.е. изменяется на один элемент множество доступов или одно из значений одной из функций. При этом остальные параметры остаются неизменными.

Определение 20. Функция переходов $T(q, (b, f)) = (b^*, f^*)$ обладает *ss*- свойством, если выполнены условия:

- Если $(s, o, read) \in b^* \setminus b$, то $f_s(s) \geq f_o(o)$ и $f^* = f$,
- Если $f_s(s) \neq f_s^*(s)$, то $f_o^* = f_o$, $b^* = b$ и $(s, o, read) \notin b$, где $f_s^*(s) < f_o(o)$,
- Если $f_o(o) \neq f_o^*(o)$, то $f_s^* = f_s$, $b^* = b$ и $(s, o, read) \notin b$, где $f_s(s) < f_o^*(o)$.

Определение 21. Функция переходов $T(q, (b, f)) = (b^*, f^*)$ обладает * - свойством, если выполнены условия:

- Если $\{(s, x, read), (s, y, write)\} \subseteq b^*$ и $\{(s, x, read), (s, y, write)\} \not\subset b$, то $f_o(y) \geq f_o(x)$ и $f = f^*$,
- Если $f_o^*(y) \neq f_o(y)$, то $b = b^*$, или $\{(s, x, read), (s, y, write)\} \not\subset b$, где $f_o^*(y) < f_o(x)$, или $\{(s, y, read), (s, x, write)\} \not\subset b$, где $f_o(x) < f_o^*(y)$.

Определение 22. Функция переходов *T* безопасна, если она обладает *ss* - свойством и * - свойством.

Теорема *BST-T*. Система $\Sigma(V, T, z_0)$ безопасна для безопасного начального состояния z_0, ее функция переходов безопасна.

Доказательство. По аналогии с доказательством *теоремы A1* необходимо показать, что, если функция переходов $T(q, (b, f)) = (b^*, f^*)$ безопасна, то состояние (b^*, f^*) безопасно в смысле, определенном в *подходе Read-Write*. Этот факт, очевидно, следует из условий *определений 20* и *21*. Таким образом, при безопасном начальном состоянии любая реализация системы, а, следовательно, и система в целом будут безопасными.

■

В рамках *подхода Transaction* можно рассмотреть вопрос о возможностях смены уровня безопасности субъектов и объектов.

Пусть $\forall x \in S$, $c_s(x) \subseteq S$ - множество субъектов, имеющих право менять уровень допуска субъекта x.

Пусть $\forall y \in O$, $c_o(y) \subseteq S$ - множество субъектов, имеющих право менять гриф секретности объекта y.

В этом случае в функции переходов необходимо внести еще один параметр, определяющего субъекта, дающего запрос системе.

Определение 23. Функция переходов $T(s, q, (b, f)) = (b^*, f^*)$ безопасна в смысле изменения уровней, если выполнены условия:

- если $f_s^*(x) \neq f_s(x)$, то $s \in c_s(x)$,
- если $f_o^*(y) \neq f_o(y)$, то $s \in c_o(y)$.

Таким образом, задавая множества $c_s(x)$ и $c_o(y)$, можно рассматривать случаи, когда все субъекты могут менять уровни безопасности, никто не может, или только системный администратор может менять уровни безопасности.

Проблемы использования модели *BL*

Кроме отмеченной выше проблемы некорректного определения безопасности в модели *BL* возможно возникновения трудностей иного рода, возникающих при использовании модели *BL* в реальных компьютерных системах.

Ниже по работе [18] мы рассмотрим ряд примеров программ, написанных на некотором абстрактном языке программирования высокого уровня, иллюстрирующие трудности практического использования положений классической модели *BL*.

Пример 1. Временной канал утечки информации.

Пусть

F_1 — секретный файл, который может содержать или запись «*A*» или запись «*B*»;

F_2 — несекретный файл;

S_1 — субъект, работающий по программе:

```
Process S1(F1: file):
        Open F1  for read
        While F1 = «A» Do End
        Close F1
End;
```

S_2 — субъект злоумышленник, работающий по программе:

```
Process S2(F1: file, F2: file):
        Start S1(F1)
        Wait 10 seconds
        Open F2  for write
        If stop S1 Then
                Write «B» to F2
        Else
                Write «A» to F2
        End If
        Close F2
End;
```

Субъект злоумышленник S_2, не открывая на чтение секретных файлов, запускает процесс S_1 и в зависимости от результатов его выполнения (процесс S_1 либо сразу завершится, либо «зависнет») записывает информацию в несекретный файл. При этом предполагается, что злоумышленник S_2 имеет уровень доступа секретно, т.к. S_1 наследует права запустившего его процесса.

Пример 2. Каналы утечки информации через локальную и логическую переменные.

Пусть F_1 и F_2 секретный и несекретный файлы соответственно.

Приведенные ниже процедуры реализуют каналы утечки информации через локальную переменную (процедура P_1) и логическую переменную (процедура P_2).

```
Procedure P1(F1: file, F2: file):
        Open F1 for read
        Read A from F1
        Close
        Open F2 for write
        Write A to F2
        Close F2
End;
Procedure P2(F1: file, F2: file):
// Считаем, что файл F1 может содержать либо запись «A», либо «B»
        Open F1 for read
        If F1 = «A» Then
                Close F1
                Open F2 for write
                Write «A» to F2
        Else
                Close F1
                Open F2 for write
                Write «B» to F2
        End If
        Close F2
End;
```

Для предотвращения каналов утечки информации через локальные или логические переменные при написании программ можно предложить руководствоваться следующими рекомендациями:

- открывать все файлы, необходимые для работы программы вначале ее выполнения;
- закрывать все файлы в конце выполнения программы;
- непосредственную обработку информации из открытых файлов осуществлять во внутренних процедурах, использующих только локальные переменные.

Таким образом, без дополнительных уточнений свойств безопасности, порядка написания кода программ классическая модель *BL* не способна предотвратить возникновение некоторых каналов утечки информации. В тоже время слишком строгая политика безопасности может привести к трудностям или даже невозможности практического использования системы защиты. Кроме того, как уже отмечалось выше, доопределение и усложнение свойств безопасности также несет в себе скрытую угрозу.

Модель *Low-Water-Mark*

Модель *Low-Water-Mark* (*LWM*) представляет близкий к модели *BL* подход к определению свойств системы безопасности, реализующей мандатную (полномочную) политику безопасности. В модели *LWM* предлагается порядок безопасного функционирования системы в случае, когда по запросу субъекта ему всегда необходимо предоставлять доступ на запись в объект. Описание модели *LWM* дадим по работе [2].

Пусть определены конечные множества:

S — множество субъектов системы;

O — множество объектов системы;

$R = \{read, write\}$ — множество видов доступа субъектов из S к объектам из O.

Обозначим через:

$B = \{b \subseteq S \times O \times R\}$ — множество возможных множеств текущих доступов в системе;

L — множество уровней секретности;

$(f_s, f_o) \in F = L^s \times L^o$ — двойка функций (f_s, f_o), определяющих

$f_s : S \to L$ — уровень допуска субъекта,

$f_o : O \to L$ — уровень секретности объекта,

$V = B \times F$ — множество состояний системы;

$W \subseteq OP \times V \times V$ — множество действий системы, где тройка $(op, (b, f), (b^*, f^*)) \in W$ означает, что система в результате выполнения операции $op \in OP$ перешла из состояния (b, f) в состояние (b^*, f^*).

Множество OP содержит операции *Read*, *Write, Reset*, описанные в таблице:

Операция	Условия выполнения	Результат выполнения операции
Read(s, o)	$f_s(s) \geq f_o(o)$	$f^* = f$; $b^* = b \cup \{(s, o, read)\}$.
Write(s, o)	$f_s(s) \leq f_o(o)$	$f_s^* = f_s$, $\forall o' \neq o \ f_o^*(o') = f_o(o')$, $f_o^*(o) = f_s(s)$, *If* $(f_o^*(o) < f_o(o))$ *Then* $o = \varnothing$, $b^* = b \cup \{(s, o, read)\}$
Reset(s, o)	$f_s(s) > f_o(o)$	$f_s^* = f_s$, $\forall o' \neq o \ f_o^*(o') = f_o(o')$, $f_o^*(o) = max$®.

В результате выполнения операции *Write* уровень секретности объекта снижается до уровня доступа субъекта. Если это снижение реально происходит, то вся информация в объекте стирается. В результате выполнения операции *Reset* уровень секретности объекта становится максимально возможным в системе.

Дадим определения *ss* - свойства и * - свойства для модели *LWM*.

Определение 24. Доступ $(s, o, r) \in S \times O \times R$ обладает *ss* - свойством относительно $f \in F$, если $r \in \{read, write\}$ и $f_s(s) \geq f_o(o)$.

Определение 25. Доступ $(s, o, r) \in S \times O \times R$ обладает *ss* - свойством относительно $f \in F$, если он удовлетворяет одному из условий :

- $r = read$ и $f_s(s) \geq f_o(o)$,
- $r = write$ и $f_s(s) = f_o(o)$.

Определение 26. Состояние системы $(b, f) \in V$ обладает *ss* - свойством (* - свойством), если каждый элемент $(s, o, r) \in b$ обладает *ss* - свойством (* - свойством) относительно f.

Определение 27. Состояние системы называется безопасным, если оно обладает *ss* - свойством и * - свойством одновременно.

Утверждение 1. Операции *Read*, *Write, Reset* переводят систему из безопасного состояния в безопасное состояние.

Доказательство. Из описания операций *Read*, *Write, Reset* следует, что в результате их выполнения новое состояние системы обладает *ss* - свойством и * - свойством.

■

Заметим, что условию стирания информации при выполнении операции *Write* является существенным. Хотя при его отсутствии *утверждение 1* формально останется истинным. Однако в этом случае с позиций здравого смысла система не будет безопасной, т.к. возможно возникновение канала утечки информации. Субъект, запрашивая доступ на запись в объект, понижает его уровень секретности, после чего он может запросить доступ на чтение из этого объекта (см. *рисунок 4.2.14*):

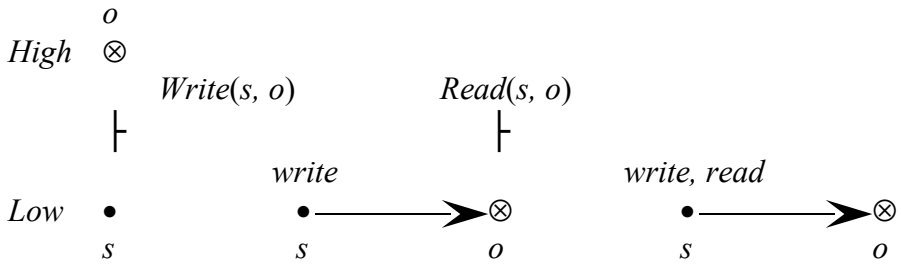


Рисунок 4.2.14. Канал утечки информации

Данный пример еще раз демонстрирует уязвимость определения безопасности в модели *BL*.

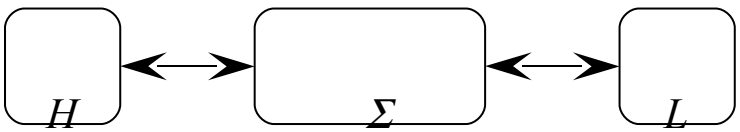
Модель безопасности информационных потоков

Рассмотренные нами модели *HRU, Take-Grant, BL* могут быть использованы при построении и анализе детерминированных систем защиты. Т.е. систем, которые не включают элементов, имеющих вероятностную природу. В тоже время целесообразно при исследовании систем, закономерности, функционирования которых сложны или практически не поддаются описанию, использовать элементы теории вероятностей. К числу таких систем можно отнести глобальные вычислительные сети, например, *Internet*, или современные многозадачные, многопользовательские сетевые операционные системы.

Рассмотрим систему защиты Σ, реализующую мандатное (полномочное) разграничение доступа. Без ограничения общности можно считать:

- в системе используются только два уровня секретности: высокий и низкий,
- все объекты системы делятся на две непересекающиеся группы: высокоуровневые объекты (*H*), имеющие право обрабатывать информацию высокого уровня секретности, и низкоуровневые (*L*),
- все взаимодействие между *H* и *L* осуществляется через систему защиты Σ.

Таким образом, систему можно представить схемой:



Задача системы защиты — не допустить возникновение информационного потока от высокоуровневых объектов к низкоуровневым.

Пусть на множестве значений объектов системы задано вероятностное распределение, т.е. *H* и *L* являются случайными величинами. Для описания и анализа информационных потоков между ними воспользуемся понятиями теории вероятностей: независимости и условного распределения. С их помощью по работе [17] рассмотрим два подхода к определению безопасности информационных потоков, основанных на понятиях:

- компьютерной невыводимости,

- информационного невмешательства.

Понятие компьютерной невыводимости

Определение 1. В системе присутствует информационный поток от высокоуровневых объектов H к низкоуровневым L, если некое возможное значение переменной в некотором состоянии низкоуровневого объекта невозможно одновременно с возможными значениями переменных состояний высокоуровневых объектов.

Определение 2. Система безопасна в смысле компьютерной невыводимости, если в ней отсутствуют информационные потоки вида, описанного в *определении 1*.

Более формально это определение можно дать следующим образом: нет информационного потока от H к L, тогда и только тогда, когда выполняется условие: если $p(H) > 0$, $p(L) > 0$, то $p(H \mid L) > 0$.

Т.к. при $p(H) > 0$, $p(L) > 0$ выполняется

$$p(L \mid H) = p(H, L) / p(H) = p(H \mid L)\, p(L) / p(H),$$

то в условиях *определения 1* из истинности неравенства $p(H \mid L) > 0$ следует истинность $p(L \mid H) > 0$, что предполагает отсутствие информационных потоков от низкоуровневых объектов к высокоуровневым. Таким образом, требования компьютерной невыводимости являются более строгими, чем требования безопасности классической модели *BL*, и фактически предполагают изоляцию друг от друга высокоуровневых и низкоуровневых объектов системы.

Понятие информационного невмешательства

В традиционной модели информационного невмешательства требуется, чтобы низкоуровневая информация была независима от высокоуровневой, т.е. выполнялось равенство

$$p(L \mid H) = p(L),$$

что при $p(H) > 0$, $p(L) > 0$ равносильно

$$p(H \mid L) = p(H).$$

Однако если ввести параметр времени, то ограничение данное выше, слишком строгое. Если L_t описывает состояния всех низкоуровневых объектов, а H_t всех высокоуровневых объектов в момент времени $t = 0, 1, 2, \ldots$, то нет необходимости требовать выполнения

$$p(H_t \mid L_{t-1}) = p(H_t) ,$$

т.е. текущее значение низкоуровневых объектов может содержать информацию о последующих значениях высокоуровневых объектов.

Например, если низкоуровневыми являются некий несекретный файл, обрабатываемый пользователем с низким допуском, а высокоуровневым объектом является журнал аудита. Тогда значение файла и операции, совершаемые над ним пользователем на шаге t, могут отображаться в журнале аудита на шаге $t + 1$.

Казалось бы, необходимо потребовать, чтобы текущее значение низкоуровневых объектов не зависело от значения высокоуровневых объектов на предыдущих тактах работы системы, т.е. выполнялось для $t = 1, 2, \ldots$

$$p(L_t \mid H_{t-1}) = p(L_t) \text{ или равносильно } p(H_{t-1} \mid L_t) = p(H_{t-1})$$

Здесь учитывается тот факт, что состояние системы влияет на последующие только через информацию, хранимую в объектах системы.

Следует отметить, что данный вариант определения соотношения L_t и H_{t-1}. является слишком строгим, т.к. предполагает независимость L_t и H_{t-1}. Однако значение высокоуровневых объектов на текущем шаге часто оказывает влияние на значение низкоуровневых объектов на последующих шагах работы системы. Рассмотрим два примера.

Пример 1. Система резервного копирования

Для защиты низкоуровневых пользователей от сбоев системы все данные, записываемые ими в низкоуровневые файлы, предварительно копируются системой в высокоуровневый аудиторский файл. В результате, если в момент времени t значения низкоуровневого файла было X, то это означает, что в момент времени t - 1 высокоуровневый файл аудита содержал значение X. Налицо зависимость значений L_t и H_{t-1} . Однако, никакой угрозой безопасности это не является.

Пример 2. Монитор ссылок

Монитор ссылок в реальных системах защиты является высокоуровневым субъектом, принимающим решения по запросам на доступ к объектам, полученным от других субъектов системы. Очевидно, что такое решение, полученное низкоуровневым субъектом на шаге t работы системы, содержит информацию о значении высокоуровневого монитора ссылок на предыдущем шаге.

Более целесообразным представляется подход, обеспечивающий невозможность накопления низкоуровневыми объектами новой информации о значение высокоуровневых объектов. Более формально, необходимо потребовать, чтобы знание значений L_{t-1} и L_t, не давало бы новой информации о H_{t-1} , т.е. должно выполняться

$p(L_t \mid H_{t-1} , L_{t-1}) = p(L_t \mid L_{t-1})$, для $t = 1, 2, \ldots$.

Данное равенство равносильно

$p(H_{t-1} \mid L_t , L_{t-1}) = p(H_{t-1} \mid L_{t-1})$,

т.е. тем самым запрещается обратный информационный поток из L_t в H_{t-1}, но не запрещается поток из L_t в H_{t+1} . Кроме этого следует отметить, что, решая проблемы, обозначенные в рассмотренных выше примерах, последнее правило запрещает временные каналы утечки информации.

С учетом того, что состояние системы влияет на последующие только через информацию, хранимую в объектах системы, дадим определение *модели безопасности информационных потоков*.

Определение 3. Система безопасна в смысле информационного невмешательства, если выполняется равенство

$p(L_t \mid H_s , L_s) = p(L_t \mid L_s)$, где $s, t = 0, 1, 2, \ldots$ и $s < t$.

Модель безопасности информационных потоков служит практическим примером подхода к построению системы защиты, которая разрешает корреляцию значений высокоуровневых и низкоуровневых объектов, но при этом остается безопасной.

Пример автоматной модели системы защиты GM

Модель *J. Goguen, J. Meseguer* (*GM*) подробно рассмотрена в работе [2].

Согласно описанию модели *GM* система защиты представляется детерминированным автоматом, на вход которого поступает последовательность команд пользователей. Для каждого пользователя системы определена функция выходов, определяющая, что каждый из них «видит» на устройстве вывода в соответствующем состоянии системы.

В модели *GM* определяется понятие информационного невлияния или невмешательства. Если всех пользователей системы поделить на две группы низкоуровневых и высокоуровневых, то информационное невмешательство есть требование независимости низкоуровневого вывода системы от ее высокоуровневого входа.

Определение 4. Система с выходной функцией *out*(*u, I*), значения которой генерируются для пользователя u в зависимости от входной истории (последовательности команд пользователей) I, обладает свойством информационного невмешательства, если выполняется

out(*u, I*) = *out*(*u, I**), где I* есть I с выброшенными командами пользователей, чей уровень безопасности выше, чем у u.

Для того чтобы сравнить *модель безопасности информационных потоков* и модель *GM*, последнюю можно рассматривать как совокупность 4-ех объектов — высокоуровневых и низкоуровневых портов ввода/вывода: *high-in, high-out, low-in, low-out*. Далее, системный вывод полностью определяется системным вводом. Вероятностные отношения исключаются.

Однако если интерпретировать детерминированность автомата, используя события с вероятностью {0, 1}, то информационное невмешательство можно определить следующим образом.

Определение 5. Система, представленная совокупностью 4-ех событий с вероятностью {0, 1}: *high-in, high-out, low-in, low-out*, обладает свойством информационного невмешательства, если выполняется

$p(low\text{-}out_t \mid high\text{-}in_s,\ low\text{-}out_s) = p(low\text{-}out_t \mid low\text{-}out_s)$, где $s, t = 0, 1, 2, \ldots$ и $s < t$.

Таким образом, в модели *GM* информационное невмешательство можно рассматривать как частный случай соответствующего понятия *модели безопасности информационных потоков*.

Литература к разделу 4

1. *Гайкович В.Ю., Першин А. Ю.* Безопасность электронных банковских систем — М.: Компания «Единая Европа», 1994 — 364 с.
2. *Грушо А.А., Тимонина Е.Е.* Теоретические основы защиты информации. - М.: Яхтсмен, - 1996. - 192 с.
3. *Мафтик С.* Механизмы защиты в сетях ЭВМ. - М: Мир, 1993. - 216 с.
4. *Хоффман Л. Дж.* Современные методы защиты информации. - М: Сов. Радио. - 1980.
5. *Щербаков А.Ю.* Разрушающие программные воздействия. М: Эдель - Киев: Век, 1993. - 64 с.
6. *Теория и практика обеспечения компьютерной безопасности* /Под ред. Зегжды П.Д. М: Яхтсмен, 1996. - 302 с.
7. *Bell D.E., LaPadula L.J.* Secure Computer Systems: Unified Exposition and Multics Interpretation, MTR-2997 Rev. 1, MITRE Corp., Bedford, Mass., March 1976.
8. *Bishop M.,* Theft of Information in the Take-Grant Protection System, Journal of Computer Security 3(4) (1994/1995) pp. 283-308.
9. *Castano S., Fugini M.G., Martella G., Samarati P.* Database Security. - Addison Wesley Publishing Company, ACM Press, 1995, 456 pp.
10. *Harrison M., Ruzzo W.* Monotonic protection systems. In *DeMillo R., Dobkin D., Jones A., Lipton R.*, editors, Foundation of Secure Computation, pp. 337-365. Academic Press, N. Y., 1978.
11. *Harrison M., Ruzzo W., Ullman J.* Protection in operating systems. Communication of ACM, 19(8): 461-471, August 1976.
12. *Jones A., Lipton R., Snyder L.* A Linear Time Algorithm for Deciding Security, Proc. 17th Annual Symp. on the Foundations of Computer Science (Oct. 1976), 33-41.
13. *Lampson B. Protection.* In 5th Princeton Symposium on Information Sciences and Systems, March 1971. Reprinted in ACM Operating Systems Review, 8(1) (1974).
14. *Lipton R., Snyder L.* On synchronization and security. In *DeMillo R., Dobkin D., Jones A., Lipton R.*, editors, Foundation of Secure Computation, pp. 367-385. Academic Press, N. Y., 1978.
15. *McLean J.* Security Models. In Encyclopedia of Software Engineering (ed. John Marciniak), Wiley Press, 1994.
16. *McLean J., John D. A* Comment on the 'Basic Security Theorem' of Bell and LaPadula, Information Processing Letters, vol. 20, no. 2, Feb. 1985.
17. *McLean J., John D.* Security Models and Information Flow, Proceedings of 1990, IEEE Symposium on Research in Security and Privacy, IEEE Press, 1990.
18. *McLean J.*, *John D.* The Specification and Modeling of Computer Security, Computer, Vol. 23, No. 1, Jan. 1990.