П. Н. ДЕВЯНИН

МОДЕЛИ БЕЗОПАСНОСТИ КОМПЬЮТЕРНЫХ СИСТЕМ

Допущено
Учебно-методическим объединением вузов по образованию
в области информационной безопасности в качестве учебного пособия
для студентов высших учебных заведений, обучающихся
по специальностям 075200 «Компьютерная безопасность»
и 075500 «Комплексное обеспечение информационной
безопасности автоматизированных систем»

Москва
ACADEMIA
2005

УДК 65.012.810(075.8)
ББК 32.973.202я73
Д259

Рецензенты:
канд. техн. наук, зав. кафедрой Информационной безопасности МИЭМ *А. Б. Лось*; канд. техн. наук, доцент МИФИ *А. И. Толстой*; канд. техн. наук, доцент МИФИ *В. А. Петров*

Девянин П. Н.

Д259 Модели безопасности компьютерных систем: Учеб. пособие для студ. высш. учеб. заведений / Петр Николаевич Девянин. — М.: Издательский центр «Академия», 2005. — 144 с.
ISBN 5-7695-2053-1

Рассмотрены с полными доказательствами положения основных моделей безопасности компьютерных систем: дискреционного, мандатного, ролевого разграничений доступа, безопасности информационных потоков, изолированной программной среды. Приведен используемый в рассматриваемых моделях математический аппарат.

Учебное пособие разработано при содействии Академии криптографии Российской Федерации.

Для студентов высших учебных заведений. Может быть полезно специалистам в области защиты информации.

УДК 65.012.810(075.8)
ББК 32.973.202я73

ISBN 5-7695-2053-1

ПРЕДИСЛОВИЕ

В настоящее время имеется достаточно много литературы, в том числе учебной, в которой приводятся обзоры и описания различных моделей безопасности компьютерных систем. В то же время их изложение, как правило, носит фрагментарный характер. Основное внимание в имеющейся литературе уделяется лишь общей формулировке основных определений и результатов моделей безопасности либо краткому их перечислению обзорного характера (без подробного рассмотрения, применяемого математического аппарата и приведения доказательств). В то же время в книгах, где доказательства приводятся, они, как правило, даются в общих чертах.

В данном учебном пособии рассмотрены с полными доказательствами положения ряда основных моделей безопасности компьютерных систем. Приведен используемый в рассматриваемых моделях математический аппарат.

Учебное пособие разработано при содействии Академии криптографии Российской Федерации.

Учебное пособие предназначено для дисциплин «Теоретические основы компьютерной безопасности», «Безопасность операционных систем», «Безопасность вычислительных систем» по специальностям 075200 «Компьютерная безопасность» и 075500 «Комплексное обеспечение информационной безопасности автоматизированных систем» и для дисциплины «Основы компьютерной безопасности» для непрофилирующих специальностей.

Учебное пособие написано на основе десятилетнего опыта преподавания перечисленных дисциплин в Институте криптографии, связи и информатики (ИКСИ).

ГЛАВА 1

ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ, ИСПОЛЬЗУЕМЫЕ ПРИ ОПИСАНИИ МОДЕЛЕЙ БЕЗОПАСНОСТИ КОМПЬЮТЕРНЫХ СИСТЕМ

1.1. ЭЛЕМЕНТЫ ТЕОРИИ ЗАЩИТЫ ИНФОРМАЦИИ

1.1.1. Объект, субъект, доступ

Дадим основные определения на основе [2].

Используем следующие обозначения:

A — конечный алфавит;

A^* — множество слов конечной длины в алфавите A;

$Я \subset A^*$ — язык описания информации, являющийся множеством слов, выделенных по определенным правилам из A.

Аксиома 1.1. Любая информация в компьютерной системе представляется словом в некотором языке $Я$.

Определение 1.1. Объект относительно языка $Я$ (или просто объект) — произвольное конечное множество слов языка $Я$.

Определение 1.2. Преобразование информации — отображение, заданное на множестве слов языка $Я$.

Описание преобразования также является словом.

Для выполнения преобразования информации в компьютерных системах требуется время, поэтому каждое преобразование может или выполняться, или храниться. В первом случае описание преобразования взаимодействует с другими ресурсами компьютерной системы. Во втором случае, как правило, речь идет о хранении описания преобразования в некотором файле.

Определение 1.3. Субъект — объект, описывающий преобразование, которое выполняется в компьютерной системе.

Используем следующие обозначения:

O — множество объектов системы;

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

Определение 1.4. Информационным потоком от объекта o (источника) к объекту o' (приемнику) назовем преобразование информации в объекте o', зависящее от информации в объекте o.

Субъект для выполнения преобразования использует информацию, содержащуюся в объектах компьютерной системы, т.е. осуществляет к ним доступ. Основными видами доступа являются:

- доступ субъекта к объекту на чтение (*read*);
- доступ субъекта к объекту на запись (*write*), при этом возможно уничтожение информации, имевшейся в объекте;

• доступ субъекта к объекту на активизацию (*execute*), при этом инициируется выполнение (производится активизация) в компьютерной системе преобразования информации, описанного в объекте.

Определение 1.5. В каждом состоянии компьютерной системы на множестве субъектов введем бинарное отношение *a* активизации: выполняется s_1 *a* s_2, где s_1, $s_2 \in S$, если субъект s_1, выполняя описанное в нем преобразование информации, инициирует выполнение преобразования информации, описанного в субъекте s_2.

Определение 1.6. В каждом состоянии компьютерной системы определим ориентированный граф активизации $G = (S, E)$, где S — вершины графа; E — ребра графа, соединяющие субъектов, связанных бинарным отношением активизации.

Очевидно, что граф активизации представляет собой множество деревьев, или лес (см. подразд. 1.2).

Определение 1.7. Пользователи — субъекты компьютерной системы, соответствующие корням деревьев графа активизации.

В критериях оценки защищенности компьютерных систем *TCSEC* [23] («Оранжевая книга») приводится основная аксиома теории защиты информации, позволяющая выделить элементы компьютерной системы, необходимые для анализа ее безопасности.

Аксиома 1.2 (основная аксиома теории защиты информации). Все вопросы безопасности информации описываются доступами субъектов к объектам.

1.1.2. Классификация угроз безопасности информации

Определение 1.8. Угроза безопасности информации (компьютерной системы) — потенциально возможное воздействие на информацию (компьютерную систему), которое прямо или косвенно может нанести урон пользователям или владельцам информации (компьютерной системы).

При классификации угроз выделяют три основных свойства безопасности информации [6].

Определение 1.9. Конфиденциальность информации — субъективно определяемая характеристика информации, указывающая на необходимость введения ограничений на круг субъектов, имеющих доступ к данной информации.

Определение 1.10. Целостность информации — свойство информации, заключающееся в ее существовании в неискаженном виде (неизменном по отношению к некоторому фиксированному ее состоянию).

Определение 1.11. Доступность информации — свойство компьютерной системы (среды, средств и технологии обработки), в

которой циркулирует информация, характеризующееся способностью обеспечивать своевременный беспрепятственный доступ субъектов к интересующей их информации и готовность соответствующих автоматизированных служб к обслуживанию поступающих от субъектов запросов всегда, когда в обращении к ним возникает необходимость.

В соответствии с тремя основными свойствами безопасности информации различают три классические угрозы безопасности информации.

Определение 1.12. Угроза конфиденциальности информации состоит в нарушении установленных ограничений на доступ к информации.

Определение 1.13. Угроза целостности информации — несанкционированное изменение информации, случайное или преднамеренное.

Определение 1.14. Угроза доступности информации осуществляется, когда несанкционированно блокируется доступ к информации (блокирование может быть постоянным или на некоторое время, достаточное, чтобы информация стала бесполезной).

Кроме перечисленных угроз выделяют еще одну угрозу, реализация которой, как правило, предшествует реализации любой из классических угроз.

Определение 1.15. Угроза раскрытия параметров компьютерной системы — преодоление защиты компьютерной системы, выявление параметров, функций и свойств ее системы безопасности.

Осуществление угрозы конфиденциальности информации нередко происходит с использованием неблагоприятных информационных потоков (каналов утечки информации). Выделяют два основных вида неблагоприятных информационных потоков: по памяти и по времени.

Информационный поток по памяти описывается схемой, представленной на рис. 1.1.

Информационный поток по времени описывается схемой, представленной на рис. 1.2.

Опасность неблагоприятного информационного потока по времени определяется долей ценной информации, модулируемой на процесс s_1. Наиболее распространенными случаями реализации неблагоприятного информационного потока по времени являются:

- перехват информации в канале связи;
- побочные каналы утечки информации по излучению, сети питания или акустике.

При анализе угрозы целостности информации следует иметь в виду, что язык ее описания аналогичен языку описания угрозы конфиденциальности. Используя при описании требований защиты информации от угрозы целостности доступы субъектов к объектам, можно сделать выводы, аналогичные выводам, полученным

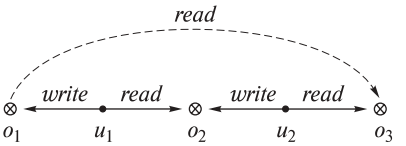


Рис. 1.1. Информационный поток по памяти:

u_1 — нарушитель; u_2 — пользователь, обрабатывающий ценную информацию; o_1 — объект, доступный нарушителю на запись; o_2 — общедоступный объект; o_3 — ценный объект

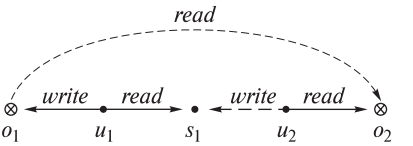


Рис. 1.2. Информационный поток по времени:

u_1 — нарушитель; u_2 — пользователь, обрабатывающий ценную информацию; s_1 — процесс, на который отражается по времени информация о работе пользователя u_2; o_1 — объект, доступный нарушителю на запись; o_2 — ценный объект

при описании требований защиты от угрозы конфиденциальности; при этом следует заменить доступы на чтение информации доступами на запись, и наоборот.

Угроза доступности информации, как правило, описывается с использованием параметра, называемого максимальным временем ожидания ответа на запрос на доступ к ресурсу (*MWT — maximum wait time*). Таким образом, для каждого ресурса компьютерной системы определяется время, приемлемое для ожидания его получения.

Следует считать, что реализовалась угроза доступности информации, если ресурс с соответствующим максимальным временем ожидания, санкционировано запрашиваемый пользователем в момент времени t, не предоставлен ему к моменту времени $t + MWT$.

Можно использовать несколько подходов для определения *MWT*:

- определив $MWT = \infty$ для всех ресурсов, можно исключить рассмотрение угрозы доступности информации при анализе безопасности компьютерной системы;
- можно определить *MWT* только для некоторых доступов субъектов к ресурсам компьютерной системы, действительно критичных с точки зрения ее безопасности.

1.1.3. Основные виды политик безопасности

Дискреционная политика безопасности

Определение 1.16. Дискреционная политика безопасности — политика безопасности, основанная на дискреционном управлении доступом (*Discretionary Access Control*), которое определяется двумя свойствами:

- все субъекты и объекты идентифицированы;

• права доступа субъектов на объекты системы определяются на основании некоторого внешнего по отношению к системе правила.

Основным элементом систем дискреционного разграничения доступа является матрица доступов.

Определение 1.17. Матрица доступов — матрица размером $|S| \times |O|$, строки которой соответствуют субъектам, а столбцы соответствуют объектам. При этом каждый элемент матрицы доступов $M[s, o] \subseteq R$ определяет права доступа субъекта s на объект o, где R — множество прав доступа.

К достоинствам дискреционной политики безопасности можно отнести относительно простую реализацию системы разграничения доступа. Этим обусловлен тот факт, что большинство распространенных в настоящее время компьютерных систем обеспечивают выполнение требований именно данной политики безопасности.

К недостаткам дискреционной политики безопасности относится статичность определенных в ней правил разграничения доступа. Данная политика безопасности не учитывает динамику изменений состояний компьютерной системы. Кроме того, при использовании дискреционной политики безопасности возникает вопрос определения правил распространения прав доступа и анализа их влияния на безопасность компьютерной системы. В общем случае при использовании данной политики безопасности перед системой защиты, которая при санкционировании доступа субъекта к объекту руководствуется некоторым набором правил, стоит алгоритмически неразрешимая задача — проверить, приведут ли его действия к нарушению безопасности или нет. Данный факт доказывается в подразд. 2.1.

В то же время имеются модели компьютерных систем, реализующих дискреционную политику безопасности (например, рассматриваемая в подразд. 2.2 модель *Take-Grant*), которые предоставляют алгоритмы проверки безопасности.

Тем не менее, в общем случае дискреционная политика разграничения доступа не позволяет реализовать ясную и четкую систему защиты информации в компьютерной системе. Этим обусловливается поиск других, более совершенных политик безопасности.

Мандатная политика безопасности

Определение 1.18. Мандатная (полномочная) политика безопасности — политика безопасности, основанная на мандатном разграничении доступа (*Mandatory Access Control*), которое определяется четырьмя условиями:

• все субъекты и объекты системы однозначно идентифицированы;

• задана решетка уровней конфиденциальности информации;
• каждому объекту системы присвоен уровень конфиденциальности, определяющий ценность содержащейся в нем информации;
• каждому субъекту системы присвоен уровень доступа, определяющий уровень доверия к нему в компьютерной системе.

Основная цель мандатной политики безопасности — предотвращение утечки информации от объектов с высоким уровнем доступа к объектам с низким уровнем доступа, т.е. противодействие возникновению в компьютерной системе неблагоприятных информационных потоков сверху вниз.

Чаще всего мандатную политику безопасности описывают в терминах, понятиях и определениях свойств модели Белла—ЛаПадула, которая рассмотрена в подразд. 3.1. Для систем мандатного разграничения доступа задача проверки безопасности является алгоритмически разрешимой. Кроме того, по сравнению с компьютерными системами, построенными на основе дискреционной политики безопасности, для систем, реализующих мандатную политику, характерна более высокая степень надежности. Правила мандатной политики безопасности более ясны и просты для понимания разработчиками и пользователями, что также является фактором, положительно влияющим на уровень безопасности системы.

С другой стороны, реализация систем с политикой безопасности данного типа довольно сложна и требует значительных ресурсов компьютерной системы.

Политика безопасности информационных потоков

Политика безопасности информационных потоков основана на разделении всех возможных информационных потоков между объектами системы на два непересекающихся множества: множество благоприятных информационных потоков и множество неблагоприятных информационных потоков. Цель реализации политики безопасности информационных потоков состоит в том, чтобы обеспечить невозможность возникновения в компьютерной системе неблагоприятных информационных потоков.

Политика безопасности информационных потоков в большинстве случаев используется в сочетании с политикой другого вида, например с политикой дискреционного или мандатного разграничения доступа. Реализация политики безопасности информационных потоков, как правило, на практике является трудной для решения задачей, особенно, если необходимо обеспечить защиту компьютерной системы от возникновения неблагоприятных информационных потоков по времени.

Некоторые из видов моделей политик безопасности информационных потоков рассмотрены в гл. 4.

Политика ролевого разграничения доступа

Ролевое разграничение доступа является развитием политики дискреционного разграничения доступа; при этом права доступа субъектов системы на объекты группируются с учетом специфики их применения, образуя роли.

Задание ролей позволяет определить более четкие и понятные для пользователей компьютерной системы правила разграничения доступа. Ролевое разграничение доступа позволяет реализовать гибкие, изменяющиеся динамически в процессе функционирования компьютерной системы правила разграничения доступа. На основе ролевого разграничения доступа, в том числе, может быть реализовано мандатное разграничение доступа. Ролевое разграничение доступа рассмотрено в гл. 5.

Политика изолированной программной среды

Целью реализации политики изолированной программной среды является определение порядка безопасного взаимодействия субъектов системы, обеспечивающего невозможность воздействия на систему защиты и модификации ее параметров или конфигурации, результатом которых могло бы стать изменение реализуемой системой защиты политики разграничения доступа.

Политика изолированной программной среды реализуется путем изоляции субъектов системы друг от друга и путем контроля порождения новых субъектов таким образом, чтобы в системе могли активизироваться только субъекты из предопределенного списка. При этом должна контролироваться целостность объектов системы, влияющих на функциональность активизируемых субъектов.

Подробно политика изолированной программной среды рассмотрена в гл. 6.

1.2. МАТЕМАТИЧЕСКИЕ ОСНОВЫ МОДЕЛЕЙ БЕЗОПАСНОСТИ

1.2.1. Основные понятия

Бóльшая часть используемых при описании свойств моделей безопасности компьютерных систем сведений из математики рассмотрены в тех подразделах, в которых они непосредственно ис-

пользуются. Особое внимание необходимо уделить следующим наиболее важным математическим понятиям, используемым в нескольких моделях безопасности:

- автомат;
- граф;
- алгоритмически разрешимые и алгоритмически неразрешимые проблемы;
- решетка.

1.2.2. Элементы теории автоматов

В математических моделях безопасности компьютерных систем понятие автомата часто используется для описания свойств системы защиты, ее отдельных элементов или компьютерной системы в целом.

Определение 1.19. Автоматом $A(X, S, Y, h, f)$ называется совокупность пяти элементов:

$X \neq \varnothing$ — конечный входной алфавит;

$S \neq \varnothing$ — конечное множество состояний;

$Y \neq \varnothing$ — конечный выходной алфавит;

h: $X \times S \to S$ — функция переходов автомата;

f: $X \times S \to Y$ — функция выходов автомата.

При этом A_{s_0}: $X^* \to Y^*$ — автоматное отображение, где s_0 — начальное состояние автомата; X^* и Y^* — конечные последовательности элементов соответственно входного и выходного алфавитов.

Определение 1.20. Гомоморфизмом φ автомата $A(X, S, Y, h, f)$ в автомат $A'(X', S', Y', h', f')$ называется совокупность трех функций $\varphi = (\alpha, \beta, \gamma)$, где

α: $X \to X'$;

β: $S \to S'$;

γ: $Y \to Y'$.

При этом для $x \in X$, $s \in S$, $y \in Y$ справедливы равенства:

$\beta(h(x, s)) = h'(\alpha(x), \beta(y))$;

$\gamma(h(x, s)) = f'(\alpha(x), \beta(y))$.

Таким образом, коммутативна диаграмма, представленная на рис. 1.3.

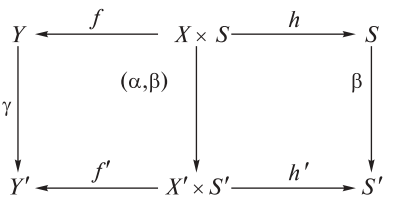


Рис. 1.3. Гомоморфизм автоматов

Определение 1.21. Суммой двух автоматов $A_1(X, S_1, Y_1, h_1, f_1)$ и $A_2(X, S_2, Y_2, h_2, f_2)$, где $S_1 \cap S_2 = \varnothing$, называется автомат $(A_1 + A_2)(X, S_1 \cup S_2, Y_1 \cup Y_2, h, f)$; при этом для $x \in X$ выполняются условия (рис. 1.4):

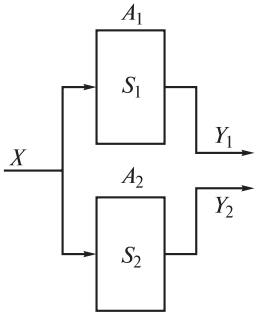


Рис. 1.4. Сумма двух автоматов $(A_1 + A_2)(X,\ S_1 \cup S_2,\ Y_1 \cup Y_2,\ h,\ f)$

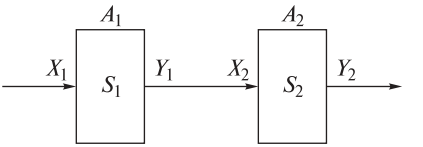


Рис. 1.5. Последовательное соединение двух автоматов $(A_1 \to A_2)(X_1,\ S_1 \times S_2,\ Y_2,\ h,\ f)$

$h(x,\ s) = h_i(x,\ s)$, если $s \in S_i$, для $i \in \{1,\ 2\}$;
$f(x,\ s) = f_i(x,\ s)$, если $s \in S_i$, для $i \in \{1,\ 2\}$.

Определение 1.22. Последовательным соединением двух автоматов $A_1(X_1,\ S_1,\ Y_1,\ h_1,\ f_1)$ и $A_2(X_2,\ S_2,\ Y_2,\ h_2,\ f_2)$, где $Y_1 \subseteq X_2$, называется автомат $(A_1 \to A_2)(X_1,\ S_1 \times S_2,\ Y_2,\ h,\ f)$; при этом для $x \in X_1$, $s_1 \in S_1$, $s_2 \in S_2$ справедливы равенства (рис. 1.5):

$$h(x,\ (s_1,\ s_2)) = (h_1(x,\ s_1),\ h_2(f(x,\ s_1),\ s_2));$$

$$f(x,\ (s_1,\ s_2)) = f_2(f_1(x,\ s_1),\ s_2).$$

1.2.3. Элементы теории графов

Элементы теории графов широко используются в математических моделях безопасности компьютерных систем, например для представления траекторий состояний функционирования системы, матрицы доступов систем дискреционного разграничения доступа.

Определение 1.23. Графом $G = (V,\ E)$ называется совокупность двух элементов:

V — конечное множество вершин графа;

$E \subseteq V \times V$ — множество ребер графа.

При этом, если ребра $(v_1,\ v_2)$ и $(v_2,\ v_1)$ считаются одинаковыми, то граф называется неориентированным, в противном случае граф называется ориентированным.

Определение 1.24. Вершины $v,\ v' \in V$ соединены путем в графе $G = (V,\ E)$, если существует последовательность ребер $(v_1,\ v_2)$, ..., $(v_{n-1},\ v_n) \in E$, где $v_1 = v$, $v_n = v'$ и $n \geq 1$. Если $v = v'$, то такой путь называется циклом.

Определение 1.25. Подграфом $G' = (V',\ E')$ графа $G = (V,\ E)$ называется граф, где $E' \subset E$ и $V' \subset V$ — множество вершин, являющихся концами ребер из E'.

Определение 1.26. Вершины $v, v' \in V$ являются связными в графе $G = (V, E)$, если v соединена путем с v' или v' соединена путем с v.

Определение 1.27. Вершины $v, v' \in V$ являются сильно связными в графе $G = (V, E)$, если v соединена путем с v' и v' соединена путем с v.

Определение 1.28. Подграф, состоящий из связных (сильно связных) между собой вершин, называется компонентой связности (сильной связности) графа.

Сильно связные вершины являются связными. В неориентированном графе связные вершины являются сильно связными.

Определение 1.29. Ориентированный граф $G = (V, E)$ называется деревом, если:

- в графе не содержатся циклы;
- существует единственная вершина (корень дерева), в которую не входит ни одно ребро;
- из корня в любую другую вершину графа существует единственный путь;
- в любую вершину, кроме корня, входит только одно ребро.

Определение 1.30. Граф, состоящий из совокупности деревьев, называется лесом.

Наиболее часто в математических моделях безопасности компьютерных систем используются следующие близкие по идее реализации алгоритмы на графах [4]:

- алгоритм обхода вершин графа;
- алгоритм поиска кратчайшего пути между вершинами графа;
- алгоритм поиска компонент связности;
- алгоритм обхода вершин дерева.

1.2.4. Алгоритмически разрешимые и алгоритмически неразрешимые проблемы

Понятие алгоритмически неразрешимой проблемы рассмотрим на основе [5].

Рассмотрим так называемые массовые проблемы. Массовая проблема представляет собой бесконечную серию индивидуальных задач.

Без ограничения общности рассмотрим массовые задачи, все индивидуальные задачи которых имеют двузначный ответ («да» и «нет»).

Определение 1.31. Характеристической функцией для массовой проблемы P называется функция f: $P \to \{0, 1\}$, где для индивидуальной задачи $\pi \in P$ выполняются условия:

- $f(\pi) = 1$, если π имеет ответ «да»;
- $f(\pi) = 0$, если π имеет ответ «нет».

Определение 1.32. Массовая задача является алгоритмически разрешимой, если существует алгоритм, вычисляющий ее характеристическую функцию. В противном случае массовая проблема является алгоритмически неразрешимой.

Решая конкретную массовую проблему, следует считаться с возможностью, что она может оказаться алгоритмически неразрешимой, поэтому необходимо иметь представление о технике доказательства неразрешимости. Основной метод, применяемый для доказательства алгоритмической неразрешимости, базируется на следующем рассуждении. Пусть имеются две массовые проблемы P_1 и P_2. Пусть имеется алгоритм A, который по всякой индивидуальной задаче $p_1 \in P_1$ строит индивидуальную задачу $p_2 \in P_2$ такую, что p_1 имеет ответ «да» тогда и только тогда, когда p_2 имеет ответ «да».

В этом случае говорят, что проблема P_1 сводится к проблеме P_2. Если проблема P_1 неразрешима, то проблема P_2 также неразрешима.

Примером алгоритмически неразрешимой проблемы является проблема останова машины Тьюринга.

1.2.5. Модель решетки

Пусть X — конечное множество.

Определение 1.33. Бинарное отношение «<» на множестве X называется отношением строгого порядка, если для любых $a, b, c \in X$ выполняются три свойства:

- антирефлексивность: не выполняется $a < a$;
- транзитивность: $(a < b,\ b < c) \Rightarrow (a < c)$;
- антисимметричность: $(a < b,\ b < a) \Rightarrow (a = b)$.

Определение 1.34. Бинарное отношение «≤» на множестве X называется отношением частичного порядка, если для любых $a, b, c \in X$ выполняются три свойства:

- рефлексивность: $a \le a$;
- транзитивность: $(a \le b,\ b \le c) \Rightarrow (a \le c)$;
- антисимметричность: $(a \le b,\ b \le a) \Rightarrow (a = b)$.

Определение 1.35. Для $a, b \in X$ элемент $c = a \oplus b \in X$ называется наименьшей верхней границей, если:

- $a \le c,\ b \le c$;
- для $d \in X$ истинно $(a \le c,\ b \le c) \Rightarrow (c \le d)$.

Определение 1.36. Для $a, b \in X$ элемент $c = a \otimes b \in X$ называется наибольшей нижней границей, если:

- $c \le a,\ c \le b$;
- для $d \in X$ истинно $(d \le a,\ d \le b) \Rightarrow (d \le c)$.

Для пары элементов частично упорядоченного множества X не обязательно существует наименьшая верхняя (наибольшая ниж-

няя) граница, но если она существует, то из антисимметричности следует ее единственность.

Определение 1.37. Пусть X — частично упорядоченное множество. $(X, \leq)$ называется решеткой, если для любых $a, b \in X$ существуют $a \oplus b \in X$ и $a \otimes b \in X$.

Лемма 1.1. Для любого набора $S = \{a_1, a_2, \ldots, a_n\}$ элементов решетки $(X, \leq)$ существуют единственные элементы:

$\oplus S = a_1 \oplus a_2 \oplus \ldots \oplus a_n$ — наименьшая верхняя граница S;

$\otimes S = a_1 \otimes a_2 \otimes \ldots \otimes a_n$ — наибольшая нижняя граница S.

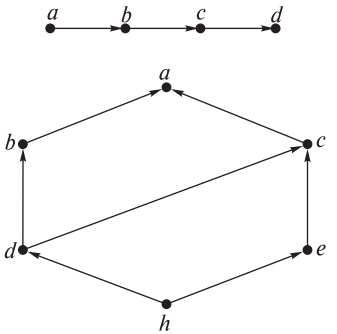


Рис. 1.6. Примеры решеток

Для решетки $(X, \leq)$ существует максимальный элемент $high = \oplus X$ и минимальный элемент $low = \otimes X$.

Определение 1.38. Линейная решетка (линейная шкала) из n элементов — это линейное упорядоченное множество; можно всегда считать $X = \{0, 1, \ldots, n\}$.

Как правило, решетки представляют с помощью ориентированных графов (рис. 1.6). При этом вершинами графа являются элементы множества X и для $a_1, a_2 \in X$ справедливо неравенство $a_1 \leq a_2$, если в графе существует путь из a_1 в a_2.

Частным важным случаем решеток является решетка подмножеств некоторого конечного множества U.

Определение 1.39. Пусть U — конечное множество, $X = 2^U$ — множество всех подмножеств множества U. Определим решетку $(X, \leq)$ с бинарным отношением частичного порядка «≤», где для $a, b \subseteq U$, $a, b \in X$ выполняется условие

$$a \leq b \text{ тогда и только тогда, когда } a \subseteq b.$$

При этом

$$a \oplus b = a \cup b,\ a \otimes b = a \cap b.$$

Другим распространенным случаем решеток является решетка многоуровневой безопасности (*MLS*). Данная решетка строится как прямое произведение линейной решетки L и решетки X подмножеств множества U.

Определение 1.40. Пусть $(L, \leq)$ — линейная решетка, $(X, \leq)$ — решетка подмножеств U. Определим решетку многоуровневой безопасности $(X \times L, \leq)$ с бинарным отношением частичного порядка «≤», где для $(a, \alpha), (b, \beta) \in X \times L$ выполняется условие

$$(a, \alpha) \leq (b, \beta) \text{ тогда и только тогда, когда } a \subseteq b,\ \alpha \leq \beta.$$

При этом

$$(a, \alpha) \oplus (b, \beta) = (a \cup b, \max\{\alpha, \beta\});$$
$$(a, \alpha) \otimes (b, \beta) = (a \cap b, \min\{\alpha, \beta\}).$$

На практике при использовании многоуровневых решеток безопасности решетка $(L, \leq)$ является линейной шкалой уровней конфиденциальности, а $(X, \leq)$ — решеткой подмножеств множества неиерархических категорий информации.

Рис. 1.7. Схема классификации и взаимосвязи положений математических моделей безопасности компьютерных систем

1.3. ОСНОВНЫЕ ВИДЫ МОДЕЛЕЙ БЕЗОПАСНОСТИ

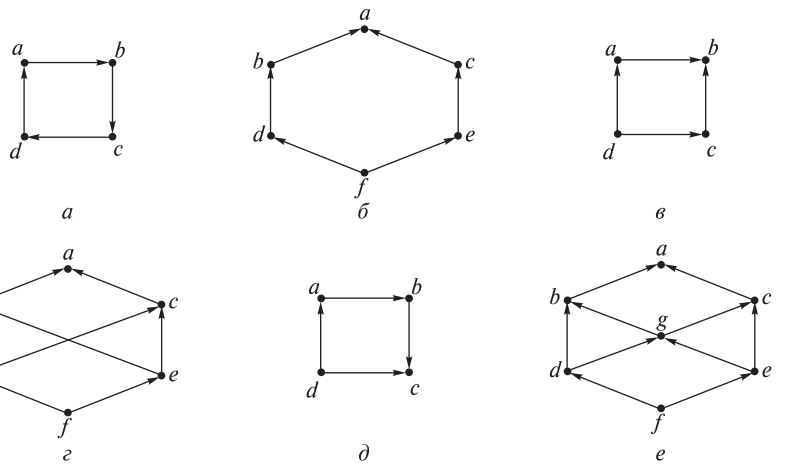
Все математические модели безопасности компьютерных систем, рассматриваемые в данном пособии, классифицируются по пяти основным видам:

- модели систем дискреционного разграничения доступа;
- модели систем мандатного разграничения доступа;
- модели безопасности информационных потоков;
- модели ролевого разграничения доступа;
- субъектно-ориентированная модель изолированной программной среды.

Схема классификации и взаимосвязи положений рассматриваемых математических моделей безопасности компьютерных систем представлены на рис. 1.7.

Контрольные вопросы

1. В чем состоит важность основной аксиомы теории защиты информации?
2. Какие основные угрозы безопасности информации рассматриваются в теории защиты информации?
3. Приведите примеры наиболее распространенных в современных операционных системах и системах управления базами данных неблагоприятных информационных потоков по памяти и по времени.
4. Какие основные виды политик безопасности рассматриваются в теории защиты информации?
5. Задают ли решетку следующие графы?



6. Нарисуйте граф, соответствующий решетке многоуровневой безопасности $(X \times L, \leq)$, для решетки $(L, \leq) = \{Low, High\}$ и $(X, \leq)$ — решетки подмножеств множества $U = \{Political, Economic, Military\}$.

ГЛАВА 2

МОДЕЛИ СИСТЕМ ДИСКРЕЦИОННОГО РАЗГРАНИЧЕНИЯ ДОСТУПА

2.1. МОДЕЛЬ МАТРИЦЫ ДОСТУПОВ ХРУ

2.1.1. Элементы модели ХРУ

Модель Харрисона — Руззо — Ульмана (ХРУ) [6, 11, 13] используется для анализа систем защиты, реализующих дискреционную политику безопасности.

Элементами модели ХРУ являются:

O — множество объектов системы;

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

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

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

Определение 2.1. Автомат, построенный согласно положениям модели ХРУ, называется системой ХРУ.

Функционирование системы рассматривается только с точки зрения изменений в матрице доступа. Возможные изменения определяются шестью видами примитивных операторов, представленных в табл. 2.1.

В результате выполнения примитивного оператора α осуществляется переход из состояния $q = (S, O, M)$ в результирующее состояние $q' = (S', O', M')$. Данный переход обозначим через $q \vdash_\alpha q'$.

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

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

Таким образом, запись команды имеет следующий вид:

command $c(x_1, \ldots, x_k)$
  if $(r_1 \in M[x_{s_1}, x_{o_1}])$ *and* ... *and* $(r_m \in M[x_{s_m}, x_{o_m}])$ *then*
    α_1;
    …
    α_n;
  endif
end,

где $r_1, ..., r_m \in R$ — права доступа; $\alpha_1, ..., \alpha_n$ — последовательность примитивных операторов, параметрами которых являются параметры команды $x_1, ..., x_k$. Следует отметить, что наличие условия в теле команды не является обязательным.

При выполнении команды $c(x_1, ..., x_k)$ система осуществляет переход из состояния q в новое состояние q'. Данный переход обозначим

$$q \vdash_{c(x_1, ..., x_k)} q'.$$

При этом

$q' = q$, если одно из условий команды $c(x_1, ..., x_k)$ не выполнено;

Таблица 2.1. **Примитивные операторы модели ХРУ**

Примитивный оператор	Исходное состояние $q = (S, O, M)$	Результирующее состояние $q' = (S', O', M')$
«Внести» право $r \in R$ в $M[s, o]$	$s \in S$ $o \in O$	$S' = S, O' = O,$ $M'[s, o] = M[s, o] \cup \{r\},$ для $(s', o') \neq (s, o)$ справедливо равенство $M'[s', o'] = M[s', o']$
«Удалить» право $r \in R$ из $M[s, o]$	$s \in S$ $o \in O$	$S' = S, O' = O,$ $M'[s, o] = M[s, o] \setminus \{r\},$ для $(s', o') \neq (s, o)$ справедливо равенство $M'[s', o'] = M[s', o']$
«Создать» субъект s'	$s' \notin O$	$S' = S \cup \{s'\}, O' = O \cup \{s'\},$ для $(s, o) \in S \times O$ справедливо равенство $M'[s, o] = M[s, o],$ для $o \in O'$ справедливо равенство $M'[s', o] = \varnothing,$ для $s \in S'$ справедливо равенство $M'[s, s'] = \varnothing$
«Создать» объект o'	$o' \notin O$	$S' = S, O' = O \cup \{o'\},$ для $(s, o) \in S \times O$ справедливо равенство $M'[s, o] = M[s, o],$ для $s \in S'$ справедливо равенство $M'[s, o'] = \varnothing$
«Уничтожить» субъект s'	$s' \in S$	$S' = S \setminus \{s'\}, O' = O \setminus \{s'\},$ для $(s, o) \in S' \times O'$ справедливо равенство $M'[s, o] = M[s, o]$
«Уничтожить» объект o'	$o' \in O$ $o' \notin S$	$S' = S, O' = O \setminus \{o'\},$ для $(s, o) \in S' \times O'$ справедливо равенство $M'[s, o] = M[s, o]$

$q' = q_n$, если условие команды $c(x_1, ..., x_k)$ выполнено и существуют состояния $q_1, ..., q_n$ такие, что $q = q_0 \vdash_{\alpha_1} q_1 \vdash_{\alpha_2} ... \vdash_{\alpha_n} q_n$.

Пример 2.1. 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 2.2. 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*];
    endif
end.

2.1.2. Анализ безопасности систем ХРУ

Согласно требованиям основных критериев оценки безопасности компьютерных систем [10, 23], их системы защиты должны строиться на основе математических моделей. С использованием математических моделей должно быть теоретически обосновано соответствие системы защиты требованиям заданной политики безопасности. Для решения этой задачи необходим алгоритм, позволяющий осуществлять данную проверку. Однако, как показывают результаты анализа модели ХРУ, задача построения алгоритма проверки безопасности систем, реализующих дискреционную политику разграничения прав доступа, не может быть решена в общем случае.

Определение 2.2. Будем считать, что возможна утечка права $r \in R$ в результате выполнения команды $c(x_1, ..., x_k)$, если при переходе системы $q \vdash_{c(x_1, ..., x_k)} q'$ выполняется примитивный оператор, вносящий *r* в элемент матрицы доступов *M*, до этого *r* не содержавший.

Определение 2.3. Начальное состояние q_0 называется безопасным по отношению к некоторому праву $r \in R$, если не возможен переход системы в такое состояние *q*, в котором возможна утечка права *r*.

Определение 2.4. Система называется монооперационной, если каждая команда системы содержит один примитивный оператор.

Теорема 2.1. Существует алгоритм, проверяющий: является ли исходное состояние монооперационной системы безопасным по отношению к праву $r \in R$.

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

Заметим, что нет необходимости рассматривать в последовательностях команды, содержащие примитивные операторы вида «удалить»... и «уничтожить»..., так как необходимо проверить наличие права доступа, а не его отсутствие. Заметим также, что нет необходимости рассматривать последовательности команд, содержащих более одного примитивного оператора вида «создать»... Это обусловлено тем, что все последовательности команд, которые проверяют или вносят права доступа в новые элементы матрицы доступов *M*, могут быть заменой параметров в командах представлены последовательностями, действующими с существующими субъектами и объектами. Тем не менее, необходимо сохранить одну команду создания субъекта (которая должна быть выполнена первой) на случай, если в начальном состоянии системы $q_0 = (S_0, O_0, M_0)$ множество $S_0 = \varnothing$.

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

Число различных примитивных операторов «внести»...

$$n = |R|(|S_0| + 1)(|O_0| + 1).$$

Все команды, содержащие один и тот же примитивный оператор, но разные условия, объединяются в алгоритме проверки безопасности в одну команду с общим составным условием (с применением логических операторов *and* и *or*). Таким образом, число последовательностей команд, которые необходимо рассмотреть, равняется *n*!; при этом длина каждой последовательности равна *n*. Теорема доказана.

Следствие 2.1. Алгоритм проверки безопасности монооперационных систем имеет экспоненциальную сложность.

Теорема 2.2. Задача проверки безопасности произвольных систем ХРУ алгоритмически неразрешима.

Доказательство. Для доказательства теоремы воспользуемся фактом, доказанным в теории машин Тьюринга [5]: 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.

Для элементов и команд машины Тьюринга используем следующие обозначения:

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

$Q = \{q_0, q_1, ..., q_k\}$ — 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 управляющей головки; e — управляющая головка не перемещается;

C: $Q \times A \to Q \times A \times D$ — функция, задающая команды машины Тьюринга. Если $C(q_{i_0}, a_{i_0}) = (q_{i_1}, a_{i_1}, d)$, то это означает, что когда машина находится в состоянии q_{i_0} и управляющая головка указывает на ячейку ленты, содержащую символ a_{i_0}, тогда выполняется шаг машины, в результате которого в эту ячейку записывается символ a_{i_1}, машина переходит в состояние q_{i_1}, 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 системы модели ХРУ.

Пусть машина Тьюринга выполнила некоторое число шагов. Пронумеруем все ячейки, пройденные считывающей головкой (включая те, в которые была изначально занесена информация), числами от 1 до n. Тогда $(a_{s_1}, ..., a_{s_n})$ — заполнение ленты, где $a_{s_i} \in A$, $s_i \in \{0, 1, ..., m\}$ для $i = 1, ..., n$. Пусть считывающая головка указывает на ячейку с номером $i \in \{1, ..., n\}$, содержащую символ $a_{i_t} \in A$, где $i_t \in \{0, 1, ..., m\}$, состояние машины $q_{i_j} \in Q$, где $i_j \in \{1, ..., k\}$, и должна быть выполнена команда $C(q_{i_j}, a_{i_t}) = (q_{i_{j'}}, a_{i_{t'}}, d)$, где $q_{i_{j'}} \in Q$, $a_{i_{t'}} \in A$, $i_{t'} \in \{0, 1, ..., m\}$, $i_{j'} \in \{1, ..., k\}$, $d \in D$.

Каждой ячейке ленты поставим в соответствие субъект модели ХРУ; при этом будем считать, что $O = S = \{s_1, ..., s_n\}$. Зададим матрицу доступов M для текущего состояния. Пусть множество прав доступа $R = Q \cup A \cup \{own, left, right\}$ и в матрицу доступов внесены права:

- $a_{s_j} \in M[s_j, s_j]$ для $j = 1, ..., n$ — заполнение ленты;
- $own \in M[s_j, s_{j+1}]$ для $j = 1, ..., n - 1$ — упорядочивание субъектов, соответствующих ячейкам ленты;
- $q_{i_j} \in M[s_i, s_i]$ — управляющая головка указывает на ячейку с номером i;
- $left \in M[s_1, s_1]$ — признак крайней левой из пройденных ячеек на ленте;
- $right \in M[s_n, s_n]$ — признак крайней правой из пройденных ячеек на ленте.

	s_1	s_2	s_3	…	s_i	…	s_{n-1}	s_n
s_1	a_{s_1}, *left*	*own*						
s_2		a_{s_2}	*own*					
s_3			a_{s_3}					
…								
s_i					a_{s_i}, q_{ij}			
…								
s_{n-1}							$a_{s_{n-1}}$	*own*
s_n								a_{s_n}, *right*

Рис. 2.1. Заполнения матрицы доступов модели ХРУ

Таким образом, текущему состоянию машины Тьюринга соответствует матрица доступов M системы модели ХРУ, представленная на рис. 2.1.

Построим гомоморфизм машины Тьюринга в систему ХРУ. Для каждой команды машины Тьюринга $C(q_{i_j}, a_{i_t}) = (q_{i_{j'}}, a_{i_{t'}}, d)$ зададим соответствующую ей команду модели ХРУ:

- если $d = e$, то

command $Eq_{i_j}a_{i_t}q_{i_{j'}}a_{i_{t'}}(s)$:
  if ($q_{i_j} \in M[s, s]$) *and* ($a_{i_t} \in M[s, s]$) *then*
    «удалить» право q_{i_j} из $M[s, s]$;
    «удалить» право a_{i_t} из $M[s, s]$;
    «внести» право $q_{i_{j'}}$ в $M[s, s]$;
    «внести» право $a_{i_{t'}}$ в $M[s, s]$;
  endif
end;

- если $d = r$, то необходимо задать две команды для случаев, когда считывающая головка указывает или не указывает на самую правую ячейку ленты:

command $R1q_{i_j}a_{i_t}q_{i_{j'}}a_{i_{t'}}(s, s')$:
  if ($q_{i_j} \in M[s, s]$) *and* ($a_{i_t} \in M[s, s]$) *and* (*own* $\in M[s, s']$) *then*
    «удалить» право q_{i_j} из $M[s, s]$;
    «удалить» право a_{i_t} из $M[s, s]$;
    «внести» право $a_{i_{t'}}$ в $M[s, s]$;
    «внести» право $q_{i_{j'}}$ в $M[s', s']$;
  endif
end;

command $R2q_{i_j}a_{i_t}q_{i_{j'}}a_{i_{t'}}(s, s')$:
if $(q_{i_j} \in M[s, s])$ *and* $(a_{i_t} \in M[s, s])$ *and* $(right \in M[s, s])$ *then*
«удалить» право q_{i_j} из $M[s, s]$;
«удалить» право a_{i_t} из $M[s, s]$;
«удалить» право *right* из $M[s, s]$;
«внести» право $a_{i_{t'}}$ в $M[s, s]$;
«создать» субъект s';
«внести» право *own* в $M[s, s']$;
«внести» право a_0 в $M[s', s']$;
«внести» право $q_{i_{j'}}$ в $M[s', s']$;
«внести» право *right* в $M[s', s']$;
endif
end;

• если $d = l$, то две команды для этого случая задаются аналогично командам для случая $d = r$.

Если машина Тьюринга останавливается в своем конечном состоянии q_0, то в соответствующей системе ХРУ 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 соответствующей ей системе ХРУ. Таким образом, в общем случае для систем дискреционного разграничения доступа, построенных на основе модели ХРУ, задача проверки безопасности алгоритмически неразрешима. Теорема доказана.

Приведенные выше теоремы 2.1 и 2.2 указывают на имеющуюся проблему выбора у разработчиков систем защиты. С одной стороны, общая модель ХРУ может выражать большое разнообразие политик дискреционного разграничения доступа, но при этом не предоставляет 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 право владения на созданные ими объекты, так как не существует одного примитивного оператора, который одновременно и создает объект, и помечает его как принадлежащий создающему субъекту.

Дальнейшие исследования модели ХРУ велись в основном в направлении определения условий, которым должна удовлетворять система, чтобы для нее задача проверки безопасности была алгоритмически разрешимой. Так, в 1976 г. было доказано [13], что задача проверки безопасности алгоритмически разрешима для систем, в которых нет примитивных операторов вида «создать»... В 1978 г. было установлено [14], что таковыми могут быть системы монотонные и моноусловные, т. е. системы, команды которых не содержат операторов вида «уничтожить»... или «удалить»... и содержат условия, состоящие из не более одной проверки вида

$r \in M[s, o]$. В том же году было установлено [17], что задача проверки безопасности для систем с конечным множеством субъектов разрешима, но вычислительно сложна.

2.1.3. Модель типизированной матрицы доступов

Другая дискреционная модель, получившая название «Типизированная матрица доступов» (ТМД), представляет собой развитие модели ХРУ, дополненной концепцией типов, что позволяет несколько смягчить те условия, для которых возможно доказательство безопасности системы. Рассмотрим модель ТМД на основе [3] и [22].

Формальное описание модели ТМД включает в себя следующие элементы:

O — множество объектов системы;

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

R — множество прав доступа субъектов к объектам;

M — матрица доступов;

C — множество команд;

T — множество типов объектов;

t: $O \rightarrow T$ — функция, ставящая в соответствие каждому объекту некоторый тип;

$q = (S, O, t, M)$ — состояние системы;

Q — множество состояний системы.

Состояние системы изменяется с помощью команд из множества C. Команды ТМД имеют тот же формат, что и в модели ХРУ, но всем параметрам приписывается определенный тип:

command $c(x_1: t_1, \ldots, x_k: t_k)$
  if $(r_1 \in M[x_{s_1}, x_{o_1}])$ *and* ... *and* $(r_m \in M[x_{s_m}, x_{o_m}])$ *then*
    α_1;
    …
    α_n;
  endif
end.

Перед выполнением команды происходит проверка типов фактических параметров. Если они не совпадают с указанными в определении, то команда не выполняется. В модели используются шесть видов примитивных операторов, отличающихся от аналогичных операторов модели ХРУ только использованием типизированных параметров (табл. 2.2).

Таким образом, ТМД является обобщением модели ХРУ, которую можно рассматривать как частный случай ТМД с одним единственным типом для всех объектов и субъектов. С другой стороны, любую систему ТМД можно выразить через систему ХРУ, введя для обозначения типов специальные права доступа, а про-

Таблица 2.2. **Примитивные операторы модели ТМД**

Оператор	Исходное состояние $q = (S, O, M)$	Результирующее состояние $q' = (S', O', M')$
«Внести» право $r \in R$ в $M[s, o]$	$s \in S$ $o \in O$	$S' = S$, $O' = O$, $t'(o) = t(o)$ для $o \in O$, $M'[s, o] = M[s, o] \cup \{r\}$, для $(s', o') \neq (s, o)$ справедливо равенство $M'[s', o'] = M[s', o']$
«Удалить» право $r \in R$ из $M[s, o]$	$s \in S$ $o \in O$	$S' = S$, $O' = O$, $t'(o) = t(o)$ для $o \in O$, $M'[s, o] = M[s, o] \setminus \{r\}$, для $(s', o') \neq (s, o)$ справедливо равенство $M'[s', o'] = M[s', o']$
«Создать» субъект s' с типом t_s	$s' \notin S$	$S' = S \cup \{s'\}$, $O' = O \cup \{s'\}$, для $o \in O$ справедливы равенства $t'(o) = t(o)$, $t'(s') = t_s$, для $(s, o) \in S \times O$ справедливо равенство $M'[s, o] = M[s, o]$, для $o \in O'$ справедливо равенство $M'[s', o] = \varnothing$, для $s \in S'$ справедливо равенство $M'[s, s'] = \varnothing$
«Создать» объект o' с типом t_o	$o' \notin O$	$S' = S$, $O' = O \cup \{o'\}$, $t'(o') = t_o$, для $(s, o) \in S \times O$ справедливо равенство $M'[s, o] = M[s, o]$, для $s \in S'$ справедливо равенство $M'[s, o'] = \varnothing$
«Уничтожить» субъект s'	$s' \in S$	$S' = S \setminus \{s'\}$, $O' = O \setminus \{s'\}$, для $o \in O'$ справедливо равенство $t'(o) = t(o)$, для $(s, o) \in S' \times O'$ справедливо равенство $M'[s, o] = M[s, o]$
«Уничтожить» объект o'	$o' \in O$ $o' \notin S$	$S' = S$, $O' = O \setminus \{o'\}$, для $o \in O'$ справедливо равенство $t'(o) = t(o)$, для $(s, o) \in S' \times O'$ справедливо равенство $M'[s, o] = M[s, o]$

верку типов в командах заменив проверкой наличия соответствующих прав доступа.

Появление в каждой команде дополнительных неявных условий, ограничивающих область применения команды только объектами соответствующих типов, позволяет несколько смягчить же-

сткие условия классической модели, при которых критерий безопасности является разрешимым.

Определение 2.5. Модель монотонной типизированной матрицы доступов (МТМД) — модель ТМД, в командах которой отсутствуют немонотонные примитивные операторы вида «удалить»... и «уничтожить»...

Определение 2.6. Каноническая форма модели МТМД (КФМТМД) — модель МТМД, в которой команды, содержащие примитивные операторы вида «создать»..., не содержат условий и примитивных операторов вида «внести»...

Терема 2.3. Любая система МТМД эквивалента системе КФМТМД.

Доказательство. Пусть задана система МТМД, в которой определены множества R, T, Q, C. Построим эквивалентную ей систему КФМТМД, определив множества R^*, T^*, Q^*, C^*.

Пусть

$R^* = R \cup \{active\}$;

$T^* = T \cup \{t_{active}\}$.

В каждом состоянии $q^* = (S^*, O^*, t^*, M^*)$, соответствующем состоянию $q = (S, O, t, M)$, справедливы равенства:

$S^* = S \cup \{s_{active}\}$;

$O^* = O \cup \{s_{active}\}$.

Пусть также для каждого $o \in O$ справедливо равенство $t^*(o) = t(o)$ и s_{active} — единственный субъект такой, что $t^*(s_{active}) = t_{active}$. Кроме того, для $s \in S$, $o \in O$ справедливо равенство $M^*[s, o] = M[s, o]$ и в начальном состоянии системы для $o \in O_0$ справедливо равенство $M^*[s_{active}, o] = \{active\}$.

Таким образом, право доступа *active* обозначает активизированные субъекты и объекты КФМТМД.

Каждую команду $c(x_1: t_1, ..., x_k: t_k)$ системы МТМД, не содержащую примитивные операторы «создать»..., представим командой $c(x_1: t_1, ..., x_k: t_k, s: t_{active})$ системы КФМТМД, полученной из исходной команды добавлением условий проверки $active \in M[s, x_i]$, для $i = 1, ..., k$.

Каждую команду $c(x_1: t_1, ..., x_k: t_k)$ системы МТМД, содержащую примитивные операторы «создать»..., представим двумя монотонными командами КФМТМД:

- $c'(x_1: t_1, ..., x_k: t_k)$ — команда без проверки условий, содержащая все примитивные операторы «создать»... команды $c(x_1: t_1, ..., x_k: t_k)$;
- $c''(x_1: t_1, ..., x_k: t_k, s: t_{active})$ — команда, содержащая условия и примитивные операторы «внести»... команды $c(x_1: t_1, ..., x_k: t_k)$, условия проверки $active \in M[s, x_i]$, для всех родительских (не создаваемых в команде $c(x_1: t_1, ..., x_k: t_k)$) объектов x_i, примитивные операторы «внести» право *active* в $M[s, x_i]$ для всех объектов x_i, создаваемых в команде $c(x_1: t_1, ..., x_k: t_k)$.

Таким образом, только «активизированные» объекты (в том числе и субъекты) системы КФМТМД соответствуют объектам системы МТМД, а все преобразования над ними в системе КФМТМД соответствуют преобразованиям системы МТМД. Теорема доказана.

Для того чтобы сформулировать ограничения, необходимые для алгоритмической разрешимости задачи проверки безопасности в системах МТМД, определим отношения между типами.

Определение 2.7. Пусть $c(x_1: t_1, ..., x_k: t_k)$ — некоторая команда ТМД. Будем говорить, что t_i является дочерним типом в $c(x_1: t_1, ..., x_k: t_k)$, если в ее теле имеется один из следующих примитивных операторов:

- «создать» субъект s с типом t_i;
- «создать» объект o с типом t_i.

В противном случае будем говорить, что t_i является родительским типом в $c(x_1: t_1, ..., x_k: t_k)$.

Заметим, что в одной команде тип может быть одновременно и родительским, и дочерним. Например:

command foo(s_1: u, s_2: u, s_3: v, o_1: w, o_2: b)
    «создать» субъект s_2 с типом u;
    «создать» субъект s_3 с типом v;
end.

Здесь u является родительским типом относительно s_1 и дочерним типом относительно s_2. Кроме того, w и b являются родительскими типами, а v — дочерним типом.

Тогда можно описать взаимосвязи между различными типами с помощью графа, определяющего отношение «наследственности» между типами, устанавливаемые через операции порождения объектов.

Определение 2.8. Граф создания — ориентированный граф с множеством вершин T, в котором ребро от u к v существует тогда и только тогда, когда в системе имеется команда, в которой u является родительским типом, а v — дочерним типом.

Граф создания для каждого типа позволяет определить:

- объекты каких типов должны существовать в системе, чтобы в ней мог появиться объект или субъект заданного типа;
- объекты каких типов могут быть порождены при участии объектов заданного типа.

Определение 2.9. Система МТМД (КФМТМД) называется ациклической (АМТМД или, соответственно, АКФМТМД) тогда и только тогда, когда ее граф создания не содержит циклов; в противном случае говорят, что система является циклической.

Например, граф создания для приведенной выше команды *foo*(), содержит следующие ребра: {(u, u), (u, v), (w, u), (w, v), (b, u), (b, v)}. Система МТМД, содержащая эту команду, будет циклической, поскольку тип u является для нее одновременно и

родительским, и дочерним, что приводит к появлению в графе цикла (u, u).

Алгоритм проверки безопасности систем АМТМД будет строиться для эквивалентных им систем АКФМТМД. Он состоит из двух шагов.

Шаг 1. Используя команды, содержащие только примитивные операторы «создать»... и не содержащие условий, перейти из начального состояния системы в некоторое развернутое состояние, обеспечивающее минимально необходимый и достаточный для распространения прав доступа состав объектов.

Шаг 2. Используя команды, не содержащие примитивные операторы «создать»..., перейти из развернутого состояния в замкнутое состояние, в котором дальнейшее применение таких команд не приведет к изменениям в матрице доступов.

Второй шаг алгоритма, очевидно, всегда будет иметь конечную сложность. Так как множество объектов не изменяется, то необходимо перебрать все последовательности различных команд (по аналогии с доказательством теоремы 2.1), которых конечное число.

Наибольшую трудность в общем случае представляет разработка алгоритма построения развернутого состояния. Однако для АМТМД или АКФМТМД такой алгоритм существует. Перед его рассмотрением дадим определение.

Определение 2.10. Пусть α и β — две различные команды системы МТМД, содержащие примитивные операторы «создать»... Будем считать, что $\alpha < \beta$ тогда и только тогда, когда для каждого дочернего типа в α в графе создания найдется путь в некоторый родительский тип в β.

Лемма 2.1. В АМТМД отношение «<» на множестве команд является отношением строгого порядка (обладает свойствами антирефлексивности, антисимметричности и транзитивности).

Доказательство. Выполняется по определению отношения «<» и АМТМД.

Алгоритм 2.1. Алгоритм построения развернутого состояния для АКФМТМД.

Шаг 1. Упорядочить линейно в списке все команды, содержащие примитивные операторы вида «создать»... (команда α следует в списке перед командой β тогда и только тогда, когда или $\alpha < \beta$, или α и β несравнимы).

Шаг 2. Начиная с начального состояния, применять по созданному на шаге 1 списку все команды; при этом каждая команда применяется со всеми возможными для нее наборами родительских объектов.

Лемма 2.2. Алгоритм 2.1 заканчивает работу за конечное время на любом начальном состоянии произвольной АКФМТМД.

Доказательство. Число команд в списке конечно. Для каждой команды из списка на каждом шаге работы алгоритма существует

конечный набор объектов, которые могут являться ее параметрами. Следовательно, алгоритм 2.1 всегда заканчивает работу за конечное время. Лемма доказана.

Определение 2.11. Пусть $(q_0, q_1, ..., q_i, ...)$ — некоторая история МТМД. Для каждого состояния $q_i = (S_i, O_i, t_i, T)$, $i = 0, 1, ...$, на множестве O_i рекурсивно определим функцию порождения объектов $\pi()$:

- для каждого $o \in O_0 \subset O_i$ зададим $\pi(o) = o$,
- если объект $o \in O_k \subset O_i$ создан на шаге $0 < k \le i$ истории командой α_k, где $o_1, o_2, ..., o_m \in O_{k-1}$ — последовательность входящих в нее объектов родительских типов, то зададим $\pi(o) = \alpha_k(\pi(o_1), \pi(o_2), ..., \pi(o_m))$.

Пример 2.3. Пусть система МТМД с двумя командами:

command $cv(x: u, y: v)$
«создать» субъект y с типом v;
end,
command $cw(x: u, y: v, z: w)$
«создать» объект z с типом w;
end,

и история имеет вид $q_0 \vdash_{cv(x, y)} q_1 \vdash_{cw(x, y, z)} q_2$, где $q_0 = (S_0, O_0, t_0, T) = (\{x\}, \{x\}, \{(x, u)\}, \{u, v, w\})$. Тогда в состоянии $q_2 = (S_2, O_2, t_2, T) = (\{x, y\}, \{x, y, z\}, \{(x, u), (y, v), (z, w)\}, \{u, v, w\})$ функция $\pi()$ будет иметь следующие значения:

$\pi(x) = x$,
$\pi(y) = cv(x)$,
$\pi(z) = cw(x, cv(x))$.

Лемма 2.3. В развернутом состоянии АМТМД для каждого возможного значения функции порождения объектов $\pi()$ существует только один объект.

Доказательство. Развернутое состояние для АМТМД строится по алгоритму 2.1 за конечное число шагов; при этом, по определению шага 2, оно содержит объекты для всех возможных значений функции $\pi()$. В то же время, на каждом шаге построения по алгоритму 2.1 получается новый объект с новым значением функции $\pi()$. Лемма доказана.

Теорема 2.4. Существует алгоритм проверки безопасности систем АМТМД.

Доказательство. Без ограничения общности по теореме 2.3 будем рассматривать АКФМТМД.

Пусть f — начальное состояние системы; q — развернутое состояние системы, полученное из f по алгоритму 2.1; m — замкнутое состояние системы, полученное из q с использованием всех команд, не содержащих примитивные операторы вида «создать»...; при этом дальнейшее применение таких команд к m не приводит к изменениям в матрице доступов.

Так как система монотонная, то для $(s, o) \in S_q \times O_q$ выполняется условие

$$M_q[s, o] \subseteq M_m[s, o]. \quad (2.1)$$

Покажем, что для систем АКФМТМД для любой истории $(f, ..., h, ...)$ может быть найдена история $(q, ..., g, ...)$ без команд, содержащих примитивные операторы вида «создать»..., где для $(s, o) \in S_h \times O_h$ выполняется условие

$$M_h[s, o] \subseteq M_g[\pi(s), \pi(o)]. \quad (2.2)$$

Если условие (2.2) выполняется, то из (2.1) и (2.2) следует, что для $(s, o) \in S_f \times O_f$ выполняется условие $M_h[s, o] \subseteq M_g[\pi(s), \pi(o)] \subseteq M_m[\pi(s), \pi(o)]$. Так как f — начальное состояние системы, то заменяем $\pi(s)$ на s, $\pi(o)$ на o и получаем, что для $(s, o) \in S_f \times O_f$ выполняется условие

$$M_h[s, o] \subseteq M_g[s, o] \subseteq M_m[s, o],$$

что и означает алгоритмическую разрешимость задачи проверки безопасности систем АМТМД.

Обоснуем (2.2), для чего построим алгоритм преобразования последовательности команд истории $(f, ..., h, ...)$ в последовательность команд истории $(q, ..., g, ...)$.

Шаг 1. В командах истории $(f, ..., h, ...)$ удалить примитивные операторы вида «создать»...

Шаг 2. Команды $c(x_1: t_1, ..., x_k: t_k)$ истории $(f, ..., h, ...)$ заменить на команды $c(\pi(x_1): t_1, ..., \pi(x_k): t_k)$ истории $(q, ..., g, ...)$.

По индукции по длине истории $(f, ..., h, ...)$ легко показать, что выполняются два условия.

Условие 1. Условие каждой команды истории $(q, ..., g, ...)$ истинно тогда и только тогда, когда истинно условие соответствующей ей команды истории $(f, ..., h, ...)$.

Условие 2. Для состояния h истории $(f, ..., h, ...)$ и соответствующего ему состояния g истории $(q, ..., g, ...)$ выполняется условие для $(s, o) \in S_h \times O_h$ истинно $(r \in M_h[s, o]) \Rightarrow (r \in M_g[\pi(s), \pi(o)])$.

Таким образом, (2.2) обосновано. Теорема доказана.

Следствие 2.2. Алгоритм проверки безопасности систем АМТМД имеет экспоненциальную сложность.

Доказательство. При построении замкнутого состояния системы из развернутого состояния используется алгоритм, аналогичный алгоритму, использованному в теореме 2.1. По следствию 2.1 получаем, что алгоритм проверки безопасности систем АМТМД имеет экспоненциальную сложность.

Определим подмножество АМТМД, называемое тернарными АМТМД.

Определение 2.12. Тернарными называются АМТМД, в которых каждая команда имеет не более трех параметров.

Для тернарной АМТМД доказано [8], что алгоритм проверки ее безопасности имеет полиномиальную сложность.

Таким образом, введение строгого контроля типов в дискреционную модель ХРУ позволило доказать критерий безопасности систем для более приемлемых ограничений, что существенно расширило область ее применения.

2.2. МОДЕЛЬ РАСПРОСТРАНЕНИЯ ПРАВ ДОСТУПА *TAKE-GRANT*

2.2.1. Основные положения классической модели *Take-Grant*

Классическая модель *Take-Grant* ориентирована на анализ путей распространения прав доступа в системах дискреционного разграничения доступа. Классическую модель *Take-Grant* рассмотрим на основе [6].

Основные элементы модели *Take-Grant*:

O — множество объектов;

$S \subseteq O$ — множество субъектов;

$R = \{r_1, r_2, ..., r_m\} \cup \{t, g\}$ — множество видов 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. Элементы множеств S и O являются вершинами графа, которые соответственно будем обозначать: $\otimes$ — объекты (элементы множества $O \setminus S$), $\bullet$ — субъекты (элементы множества S). Элементы множества $E \subseteq O \times O \times R$ являются ребрами графа. Каждое ребро помечено непустым подмножеством множества видов прав доступа R.

Состояние системы описывается соответствующим ему графом доступов. В отличие от модели ХРУ в модели *Take-Grant* возможно наличие прав доступа не только у субъектов на объекты, но и у объектов на объекты.

Основная цель классической модели *Take-Grant* — определение и обоснование алгоритмически проверяемых условий проверки возможности утечки права доступа по исходному графу доступов, соответствующего некоторому состоянию системы.

Порядок перехода системы модели *Take-Grant* из состояния в состояние определяется правилами преобразования графа доступов, которые в классической модели носят название де-юре правил. Преобразование графа G в граф G' в результате выполнения правила *op* обозначим следующим образом:

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