

## **12. Модели безопасности**

В настоящей главе модели рассматриваются в соответствии с типом угроз, от которых защищают информацию вычислительные системы, синтезированные на основании данных моделей.

### ***12.1. Модели разграничения доступа***

Модели разграничения доступа служат для синтеза политик безопасности, направленных на предотвращение угрозы раскрытия, заключающейся в том, что информация становится известной тому, кому не следовало бы ее знать. Данные модели могут быть классифицированы следующим образом:

- модели разграничения доступа, построенные по принципу предоставления прав;
- модели разграничения доступа, построенные на основе принципов теории информации;
- модели разграничения доступа, использующие принципы теории вероятностей.

В дальнейших пунктах мы рассмотрим модели разграничения доступа, построенные на основании перечисленных выше принципов.

### ***12.2. Модели дискреционного доступа***

Модели разграничения доступа, построенные по принципу предоставления прав, являются самой естественной основой для построения политик разграничения доступа. Система, политика безопасности которой построена на основании данного принципа, впервые была описана в литературе в середине шестидесятых годов.

Неформально право доступа может быть описано как "билет", в том смысле, что владение "билетом" разрешает доступ к некоторому объекту, описанному в билете.

Основными типами моделей, построенных на предоставлении прав, являются модели дискреционного и мандатного доступов. Модели данного типа используются в большинстве реальных систем, синтезированных в настоящее время. Требования, на которых основаны данные модели, лежат в основе требований, сформулированных в различных государственных нормативных документах.

### ***12.2.1. Модель дискреционного доступа АДЕПТ-50.***

Одна из первых моделей безопасности была модель дискреционного доступа, модель АДЕПТ-50 [Ошибка! Источник ссылки не найден.,Ошибка! Источник ссылки не найден.]. В модели представлено четыре типа объектов, относящихся к безопасности: пользователи( $u$ ), задания( $j$ ), терминалы( $t$ ) и файлы( $f$ ), причем каждый объект описывается четырехмерным кортежем ( $A, C, F, M$ ), включающим параметры безопасности:

**Компетенция  $A$**  – скаляр – элементы из набора иерархически упорядоченных уровней безопасности, таких как НЕСЕКРЕТНО, КОНФИДЕНЦИАЛЬНО, СЕКРЕТНО, СОВЕРШЕННО СЕКРЕТНО.

**Категория  $C$**  – дискретный набор рубрик. Категории не зависят от уровня безопасности. Пример набора рубрик: ОГРАНИЧЕНО, ТАЙНО, ТОЛЬКО ДЛЯ ПРОСМОТРА, ЯДЕРНЫЙ, ПОЛИТИЧЕСКИЙ.

**Полномочия  $F$**  – группа пользователей, имеющих право на доступ к определенному объекту.

**Режим  $M$**  – набор видов доступа, разрешенных к определенному объекту или осуществляемых объектом. Пример: ЧИТАТЬ ДАННЫЕ, ПРИСОЕДИНЯТЬ ДАННЫЕ, ИСПОЛНИТЬ ПРОГРАММУ.

Если  $U=\{u\}$  обозначает набор всех пользователей, известных системе, а  $F(i)$  - набор всех пользователей, имеющих право использовать объект  $i$ , то для модели формулируются следующие правила:

1. Пользователь  $u$  получает доступ к системе  $\Leftrightarrow u \in U$ .
2. Пользователь  $u$  получает доступ к терминалу  $t \Leftrightarrow u \in F(t)$  (то есть в том и только в том случае, когда пользователь  $u$  имеет право использовать

терминал  $t$ ).

3. Пользователь  $u$  получает доступ к файлу  $j \Leftrightarrow A(j) \geq A(f), C(j) \supseteq C(f), M(j) \supseteq M(f)$  и  $u \in F(f)$ , то есть тогда и только тогда, когда выполняются условия:

– привилегии выполняемого задания шире привилегий файла или равны им;

– пользователь является членом  $F(f)$ .

Задавая параметры безопасности  $A, C, F, M$ , можно сформировать матрицу определения параметров безопасности (табл. 12.1.).

Четырехмерный кортеж безопасности, полученный на основе прав задания, а не прав пользователя, используется в модели для управления доступом. Данный подход обеспечивает однородный контроль права на доступ над неоднородным множеством программ и данных, файлов, пользователей и терминалов. Например, наивысшим полномочием доступа к файлу для пользователя "СОВ. СЕКРЕТНО", выполняющего задание с "КОНФИДЕНЦИАЛЬНОГО" терминала будет "КОНФИДЕНЦИАЛЬНО".

Таблица 12.1 Матрица определения параметров безопасности модели АДЕПТ-50

Объект	A	C	F	M
Пользователь $u$	Const	Const	$\{u\}$	Const
Терминал $t$	Const	Const	$\{u(t,i)\}$	Const
Задание $j$	$\min(A(u), A(t))$	$C(u) \cap C(t)$	$\{u(j,i)\}$	$M(u) \cap M(t)$
Существ. файл $f(i)$	Const	Const	$\{u(f,i)\}$	Const
Нов. файл $f=g(f_1, f_2)$	$\max(A(f_1), A(f_2))$	$C(f_1) \cup C(f_2)$	$\{u(f,j)\}$	$M(f_1) \cup M(f_2)$

–  $f_1, f_2$  - старые файлы; новый файл  $f$  является некоторой их функцией.

### 12.2.2. Пятимерное пространство Хартсона

Теперь рассмотрим модель, называемую **пятимерным пространством безопасности Хартсона** [Ошибка! Источник ссылки не найден.]. В данной модели используется пятимерное пространство безопасности для моделирования процессов, установления полномочий и организации доступа на их основании. Модель имеет пять основных наборов:

$A$  – установленных полномочий;  $U$  – пользователей;  $E$  – операций;  $R$  – ресурсов;  $S$  – состояний.

Область безопасности будет выглядеть как декартово произведение:  $A \times U \times E \times R \times S$ . Доступ рассматривается как ряд запросов, осуществляемых пользователями  $u$  для выполнения операций  $e$  над ресурсами  $R$  в то время, когда система находится в состоянии  $s$ . Например, запрос на доступ представляется четырехмерным кортежем  $q = (u, e, R, s)$ ,  $u \in U, e \in E, s \in S, r \subseteq R$ . Величины  $u$  и  $s$  задаются системой в фиксированном виде. Таким образом, запрос на доступ – подпространство четырехмерной проекции пространства безопасности. Запросы получают право на доступ в том случае, когда они полностью заключены в соответствующие подпространства.

Процесс организации доступа можно описать алгоритмически следующим образом. Для запроса  $q$ , где  $q = (u, e, R, s)$ , набора  $U'$  вполне определенных групп пользователей, набора  $R'$  вполне определенных единиц ресурсов и набора  $P$  правильных (установленных) полномочий процесс организации доступа будет состоять из следующих процедур:

1. Вызвать все вспомогательные программы, необходимые для предварительного принятия решений.
2. Определить из  $U$  те группы пользователей, к которым принадлежит  $u$ . Затем выбрать из  $P$  спецификации полномочий, которым соответствуют выделенные группы пользователей. Этот набор полномочий  $F(u)$  определяет привилегию пользователя  $u$ .
3. Определить из  $P$  набор  $F(e)$  полномочий, которые устанавливают  $e$  как основную операцию. Этот набор называется привилегией операции  $e$ .
4. Определить из  $P$  набор  $F(R)$  (привилегию единичного ресурса  $R$ ) – полномочия, которые определяют поднабор ресурсов из  $R'$ , имеющего общие элементы с запрашиваемой единицей ресурса  $R$ .

Полномочия, которые являются общими для трех привилегий в процедурах 2, 3, 4, образуют  $D(q)$  (так называемый домен полномочий для запроса):

$q: D(q)=F(u)\cap F(e)F(R).$

5. Удостовериться, что запрашиваемый ресурс  $R$  полностью включается в  $D(q)$ , то есть каждый элемент из  $R$  должен содержаться в некоторой единице ресурса, которая определена в домене полномочий  $D(q)$ .

6. Осуществить разбиение набора  $D(q)$  на эквивалентные классы так, чтобы два полномочия попадали в эквивалентный класс тогда и только тогда, когда они специфицируют одну единицу ресурса. Для каждого такого класса логическая операция ИЛИ (или И) выполняется с условиями доступа элементов каждого класса.

Новый набор полномочий – один на каждую единицу ресурса, указанную в  $D(q)$ , есть  $F(u, q)$  - фактическая привилегия пользователя  $u$  по отношению к запросу  $q$ .

7. Вычислить ЕАС - условие фактического доступа, соответствующего запросу  $q$ , осуществляя логическое И (или ИЛИ) над условиями доступа членов  $F(u, q)$ . Это И (или ИЛИ) выполняется над всеми единицами ресурсов, которые перекрывают единицу запрошенного ресурса.

8. Оценить ЕАС и принять решение о доступе:

- разрешить доступ к  $R$ , если  $R$  перекрывается;
- отказать в доступе в противном случае.

9. Произвести запись необходимых событий.

10. Вызвать все программы, необходимые для организации доступа после "принятия решения".

11. Выполнить все вспомогательные программы, вытекающие для каждого случая из условия 8.

12. Если решение о доступе было положительным – завершить физическую обработку.

Автор модели Хартстон отмечает, что приведенная последовательность шагов не всегда необходима в полном объеме. Например, в большинстве реализаций шаги 2 и 6 осуществляются во время регистрации пользователя в системе.

### 12.2.3. Модель Харрисона-Руззо-Ульмана

Рассмотрим типичную модель системы защиты, состоящую из следующих конечных наборов [Ошибка! Источник ссылки не найден.,Ошибка! Источник ссылки не найден.,Ошибка! Источник ссылки не найден.]:

- общих прав  $A=\{a_1, \dots, a_n\}$ ;
- исходных субъектов  $S_0$  и объектов  $O_0$ ;
- команд  $C$  формы  $\alpha(X_1, \dots, X_n)$ , где  $\alpha$  - имя;  $X_1, \dots, X_n$  – формальные параметры, указывающие на объекты.

Элементами матрицы доступа являются права доступа, взятые из набора общих прав. Состояния системы изменяются при изменении элементов матрицы доступа  $M$ . Запросы к системе можно выразить в форме:

if  $a_1$  in  $M[s_1, o_1]$  and

$a_2$  in  $M[s_2, o_2]$  and

$a_m$  in  $M[s_m, o_m]$  then

$op_1$ ,

$op_2$ ,

$op_n$ .

Причем,  $\forall a_j \in A$  операция  $op$  является одной из следующих примитивных операций:

- enter  $a$  into  $(s, 0)$ ;
- delete  $a$  from  $(s, 0)$ ;
- create subject  $s$ ;
- create object  $o$ ;
- destroy subject  $o$ ;
- destroy object  $o$ .

Семантика данных операций очевидна. Для системы с начальной конфигурацией  $Q_0$  и права  $a$  можно сказать, что система безопасна для  $a$ , если не существует последовательности запросов к системе в состоянии  $Q_0$  таких, что в результате них право  $a$  будет записано в ячейку, не содержащую ее.

Существуют две теоремы о безопасности данного типа систем. Первая относится к безопасности **моно-операционных систем**. Под **моно-операционной системой** понимается система, в которой каждый запрос имеет только одну операцию.

**Теорема 12.1.** Существует алгоритм для определения, является или нет моно-операционная система безопасной для данного права  $a$ .

Вторая теорема указывает на то, что проблема безопасности для системы с запросами общего вида является неразрешимой.

**Теорема 12.2.** Проблема определения безопасности для данного права  $a$  в системе с запросами общего вида является неразрешимой.

Харрисон, Руззо и Ульман показали, что безопасными являются **монотонные системы** (системы, не содержащие операции `destroy` и `delete`), **системы не содержащие операций `create`** и **моно-условные системы** (системы, запрос к которым содержит только одно условие).

К достоинствам моделей дискреционного доступа можно отнести хорошую гранулированность защиты и относительно простую реализацию. В качестве примера реализаций данного типа моделей можно привести так называемую матрицу доступа, строки которой соответствуют субъектам системы, а столбцы – объектам; элементы матрицы характеризуют права доступа. Проблемы, возникающие в системах, синтезированных на их основании, показаны в следующем параграфе.

### **12.3. Модели мандатного доступа**

Модели дискреционного доступа хотя и обеспечивают хорошо гранулированную защиту, но обладают рядом недостатков. В частности, в системах, построенных на основе DAC, существует проблема *троянских программ (троянских коней)*. Троянскую программу следует определять как любую программу, от которой ожидается выполнение некоторого желаемого действия, а она на самом деле выполняет какое-либо неожиданное и нежелательное действие. Так, троянская программа может выглядеть как

вполне хороший продукт, но реально она может оказаться даже более опасной, чем можно было бы ожидать.

Для того чтобы понять, как может работать троянский конь, вспомним, что когда пользователь вызывает какую-либо программу на компьютере, в системе инициируется некоторая последовательность операций, зачастую скрытых от пользователя. Эти операции обычно управляются операционной системой. Троянские программы рассчитывают на то, что когда пользователь инициирует такую последовательность, он обычно верит в то, что система произведет ее, как полагается. При этом нарушитель может написать версию троянской программы, которая будучи запущенной от имени пользователя-жертвы, передаст его информацию пользователю нарушителю.

В отличие от DAC, мандатный доступ (MAC) накладывает ограничения на передачу информации от одного пользователя другому. Это позволяет разрешить проблему троянских коней.

Классической моделью, лежащей в основе построения многих систем MAC и породившей остальные модели MAC, является модель Белла и Лападула. К сожалению, данная модель не лишена недостатков и с целью устранения данных недостатков были порождены некоторые специфичные модели. В заключении параграфа мы опишем специализированные модели, основанные на рассмотрении конкретных требований, в соответствии с которыми синтезируется данная модель.

#### **12.3.1. Модель Белла и Лападула**

Модель, получившая название модели Белла и Лападула (БЛМ), до сих пор оказывает огромное влияние на исследования и разработки в области компьютерной безопасности. Об этом свидетельствует огромное количество различных документов, ссылающихся в библиографии на первоначальную БЛМ. Данная модель лежит в основе построения MAC. Идеи, заложенные в БЛМ, могут быть использованы при построении различных политик безопасности.



Идеи, лежащие в основе БЛМ, берут происхождение из "бумажного мира". Белл и Лападула перенесли модель безопасности, принятую при работе с документами, в мир компьютерных систем. Основным наблюдением, сделанным Беллом и Лападулой, является то, что в правительстве США все субъекты и объекты ассоциируются с уровнями безопасности, варьирующимися от низких уровней (неклассифицированных) до высоких (совершенно секретных). Кроме того, они обнаружили, что для предотвращения утечки информации к неуполномоченным субъектам этим субъектам с низкими уровнями безопасности **не позволено читать информацию из объектов с высокими уровнями безопасности**. Это ведет к первому правилу БЛМ.

**Простое свойство безопасности**, также известное как правило "нет чтения вверх" (NRU), гласит, что субъект с уровнем безопасности  $x_s$  может читать информацию из объекта с уровнем безопасности  $x_o$ , только если  $x_s$  преобладает над  $x_o$ . Это означает, что если в системе, удовлетворяющей правилам модели БЛМ, субъект с уровнем доступа секретный попытается прочесть информацию из объекта, классифицированного как совершенно секретный, то такой доступ не будет разрешен.

Белл и Лападула сделали дополнительное наблюдение при построении своей модели: в правительстве США субъектам **не позволено размещать** информацию или записывать ее в объекты, имеющие более низкий уровень безопасности. Например, когда совершенно секретный документ помещается в неклассифицированное мусорное ведро, может произойти утечка информации. Это ведет ко второму правилу БЛМ.

**Свойство —\***, известное как правило "нет записи вниз" (NWD), гласит, что субъект безопасности  $x_s$  может писать информацию в объект с уровнем безопасности  $x_o$  только если  $x_o$  преобладает над  $x_s$ . Это означает, что если в системе, удовлетворяющей правилам модели БЛМ, субъект с уровнем доступа секретный попытается записать информацию в неклассифицированный объект, то такой доступ не будет разрешен. Введение

свойства —\* разрешает проблему троянских коней, так как запись информации на более низкий уровень безопасности, типичная для троянских коней, запрещена.

**Правило запрета по записи является большим упрощением некоторых реализаций БЛМ.** Так, некоторые описания включают более детальное понятие типа доступа (например такие, как добавление и выполнение). Помимо этого многие модели БЛМ включают понятие дискретной защиты с целью обеспечения хорошо гранулированной защиты при сохранении всех преимуществ БЛМ.

Правила запрета по записи и чтению БЛМ отвечают интуитивным понятиям того, как предотвратить утечку информации к неуполномоченным источникам.

Рассмотрим формализацию БЛМ. Обозначим через:

$S$  — множество субъектов;

$O$  — множество объектов;

$L$  — решетка уровней безопасности;

$F : S \cup O \rightarrow L$  — функция, применяемая к субъектам и объектам; данная функция определяет уровни безопасности своих аргументов в данном состоянии;

$V$  — множество состояний — множество упорядоченных пар  $(F, M)$ , где  $M$  — матрица доступа субъектов системы к объектам.

Система представляется начальным состоянием  $v_0$ , определенным множеством запросов к системе  $R$  и функцией переходов  $T : (V \times R) \rightarrow V$  такой, что система переходит из состояния в состояние после исполнения запроса. Сформулируем определения, необходимые для доказательства основной теоремы безопасности (ОТБ), доказанной для БЛМ.

**Определение 12.1.** Состояние  $(F, M)$  безопасно по чтению (NRU) тогда и только тогда, когда для  $\forall s \in S$  и для  $\forall o \in O$ , чтение  $\in M[s, o] \rightarrow F(s) \geq F(o)$ .

**Определение 12.2.** Состояние  $(F, M)$  безопасно по записи (NWD, \*-свойство) тогда и только тогда, когда для  $\forall s \in S$  и для  $\forall o \in O$ , запись  $\in M[s, o] \rightarrow F(o) \geq F(s)$ .

**Определение 12.3.** Состояние безопасно тогда и только тогда, когда оно безопасно по чтению и записи.

**Теорема (ОТБ).** Система  $(v_0, R, T)$  безопасна тогда и только тогда, когда состояние  $v_0$  безопасно и  $T$  таково, что для любого состояния  $v$ , достижимого из  $v_0$  после исполнения конечной последовательности запросов из  $R$ ,  $T(v, c) = v^*$ , где  $v = (F, M)$  и  $v^* = (F^*, M^*)$ , переходы системы  $(T)$  из состояния  $v$  в состояние подчиняются следующим ограничениям для  $\forall s \in S$  и для  $\forall o \in O$ :

- если чтение  $\in M^*[s, o]$  и чтение  $\notin M[s, o]$ , то  $F^*(s) \geq F^*(o)$ ;
- если чтение  $\in M[s, o]$  и  $F^*(s) < F^*(o)$ , то чтение  $\notin M^*[s, o]$ ;
- если запись  $\in M^*[s, o]$  и запись  $\notin M[s, o]$ , то  $F^*(o) \geq F^*(s)$ ;
- если запись  $\in M[s, o]$  и  $F(o) < F(s)$ , то запись  $\notin M^*[s, o]$ .

### Доказательство

1. **Необходимость.** Предположим, система безопасна. Состояние  $v_0$  безопасно по определению. Если имеется некоторое состояние  $v$ , достижимое из состояния  $v_0$  после исполнения конечной последовательности запросов из  $R$  таких, что  $T(v, c) = v^*$ , хотя  $v^*$  не удовлетворяет одному из двух первых ограничений для  $T$ , то  $v^*$  будет достижимым состоянием, но противоречащим ограничению безопасности по чтению. Если  $v^*$  не удовлетворяет одному из двух последних ограничений для  $T$ , то  $v^*$  будет достижимым состоянием, но противоречащим ограничению безопасности по записи. В любом случае система небезопасна.

2. **Достаточность.** Предположим, что система небезопасна. В этом случае либо  $v_0$  должно быть небезопасно, либо должно быть небезопасно состояние  $v$ , достижимое из состояния  $v_0$  после исполнения конечной последовательности запросов из  $R$ . Если  $v_0$  небезопасно – все доказано. Если  $v_0$

безопасно, допустим, что  $v^*$  – первое в последовательности запросов небезопасное состояние. Это означает, что имеется безопасное состояние  $v$  такое, что  $T(v, c)=v^*$ , где  $v^*$  – небезопасно. Но это противоречит четырем ограничениям безопасности на  $T$ .

*Теорема доказана*

Несмотря на все достоинства, оказалось, что при использовании БЛМ в контексте практического проектирования и разработки реальных компьютерных систем возникает ряд технических вопросов. Данные вопросы являются логическим следствием достоинства БЛМ – ее простоты. Проблемы возникают при рассмотрении вопросов построения политик безопасности для конкретных типов систем, то есть на менее абстрактном уровне рассмотрения. При данном рассмотрении системный компонент модели усложняется, что может привести к неадекватности БЛМ в ее классической форме. Как следствие, в мире компьютерной безопасности ведется широкая полемика по поводу применимости БЛМ для построения безопасных систем.

Рассмотрим ряд примеров критики БЛМ. Некоторые из них взяты из литературы, посвященной вопросам безопасности, другие часто включаются в техническое описание и представляют собой так называемую "обязательную критику" БЛМ.

Начнем данное рассмотрение с обсуждения проблемы, возникающей в распределенных системах, удовлетворяющих правилам БЛМ. В частности, покажем, что запрос на чтение вызывает протекание потоков информации в обоих направлениях между компонентами, что является нарушением правил модели. Затем рассмотрим проблему использования этой модели для обеспечения безопасности доверенных субъектов, которые выполняют наиболее критичные задачи в компьютерной системе. Завершим обсуждение примером описания модели, известной как система Z.

### **Удаленное чтение**

В свете недавних тенденций использования распределенных конфигураций требуется рассматривать модели безопасности не только для

автономных, но и для распределенных компьютерных систем (распределенная система обычно состоит из нескольких объединенных систем). Очевидным способом распространения БЛМ на распределенные системы будет назначение уровней безопасности различным компонентам и соблюдение гарантий выполнения правил-ограничений по чтению и записи.

Например, некоторым компонентам можно назначить уровни безопасности, меняющиеся от неклассифицированного до совершенно секретного уровня, и на основании принципов БЛМ синтезировать соединения между различными компонентами системы. Может показаться, что если конфиденциальному субъекту А будет разрешено чтение информации из неклассифицированного объекта В, никакая конфиденциальная информация не будет раскрыта. Но при более подробном рассмотрении реализации операции удаленного чтения снизу может быть сделано неприятное наблюдение. Операция чтения между удаленными компонентами приводит к протеканию потока информации от читаемого объекта к запросившему доступ на чтение субъекту. Данный поток является безопасным, поскольку информация не разглашается неавторизованному субъекту. Однако в распределенной конфигурации чтение инициируется запросом от одного компонента к другому. Такой запрос образует **прохождение потока информации в неверном направлении (запись в объект с меньшим уровнем безопасности)**. Таким образом, удаленное чтение в распределенных системах может произойти только если ему предшествует операция записи вниз, что является нарушением правил БЛМ.

Многие исследователи рассматривают эту проблему как наиболее убедительное свидетельство неадекватности БЛМ. Однако на практике эта проблема часто является несущественной; достаточно внедрения в систему дополнительных средств обработки удаленных запросов для обеспечения того, чтобы поток информации от высокоуровневого субъекта к низкоуровневому объекту был **ограничен запросом на доступ**. Фактически, некоторые

архитектуры предлагают отдельные компоненты, выполняющие обработку таких запросов и потока информации в распределенных системах.

### **Доверенные субъекты**

В предыдущем описании правил БЛМ не было указано, какие субъекты должны подчиняться этим правилам. Например, компьютерные системы обычно имеют администратора, который управляет системой, добавляя и удаляя пользователей, восстанавливает функционирование после сбоев, устанавливает специальное программное обеспечение, устраняет ошибки в операционной системе или приложениях и т.п. Очевидно, что процессы, действующие в интересах таких администраторов, не могут управляться правилами БЛМ или каких-либо других моделей, не позволяющих им выполнять функции администрирования.

Это наблюдение высвечивает еще одну техническую проблему, связанную с правилами БЛМ. Можно сказать, что эти правила обеспечивают средства для предотвращения угрозы нарушения секретности для нормальных пользователей, но не говорят ничего по поводу той же проблемы для так называемых **доверенных субъектов**. Доверенные субъекты могут функционировать в интересах администратора. Также они могут быть процессами, **обеспечивающими критические службы такие**, как драйвер устройства или подсистема управления памятью. Такие процессы часто не могут выполнить свою задачу, не нарушая правил БЛМ. Неприменимость БЛМ для доверенных субъектов может быть выражена **путем внесения поправки в данное ранее определение операций чтения и записи БЛМ**. Но хотя это и делает определение более точным, оно несколько не облегчает задачу для разработчика, желающего построить безопасный драйвер или утилиту поддержки работы администратора.

Одним из решений, рассматриваемых в литературе по безопасности, было предложение представлять и использовать для потока информации модель, требующую того, чтобы никакая **высокоуровневая информация никогда не протекала на более низкий уровень**. В данных моделях

низкоуровневые пользователи не могут сделать выводы или затронуть работу высокоуровневых пользователей.

### **Проблема системы Z**

Джон МакЛин разработал концептуальное описание системы, названной Система Z. Данное описание показывает, что система, удовлетворяющая правилам БЛМ, может иметь ряд проблем с секретностью. Система Z выражается в терминах набора субъектов и объектов, с каждым из которых связан уровень безопасности. Совокупность уровней безопасности для каждого субъекта и объекта в некоторый момент времени описывает состояние системы. Система Z удовлетворяет БЛМ, если во всех состояниях системы комбинации уровней субъектов и объектов таковы, что в этом состоянии никакой субъект не может осуществить запись вниз или чтение сверху.

Предположив, что система Z удовлетворяет условиям БЛМ, можно быть уверенным, что любая угроза секретности будет обнаружена. Однако МакЛин указал на техническую деталь, которая не очевидна в таких системах. Если в некотором состоянии секретный субъект захотел прочитать совершенно секретный объект, то до тех пор, пока система удовлетворяет БЛМ, осуществить это будет невозможно. Но МакЛин заявляет, что ничто в БЛМ не предотвращает систему от "деклассификации" объекта от совершенно секретного до секретного (по желанию совершенно секретного пользователя).

В качестве иллюстрации можно привести следующий пример. Допустим, субъект с высокой степенью доверия A читает информацию из объекта, уровень классификации которого также равен A. Далее данный субъект понижает свою степень доверия до уровня B ( $A > B$ ). После этого он может записать информацию в файл с классификацией B. Нарушения БЛМ формально не произошло, но безопасность системы нарушена.

Фактически, МакЛин описал конфигурацию, в которой все субъекты могут читать и записывать любой объект путем назначения соответствующих уровней безопасности объекта перед выполнением запросов на доступ. В такой



системе, которая очевидно не обеспечивает секретность информации, все состояния могут быть рассмотрены как удовлетворяющие требованиям БЛМ.

Все описанное выше является справедливым для модели БЛМ в "ее классической формулировке", кочующей из книги в книгу и из статьи в статью. Но в оригинальной модели, представленной авторами, было введено требование **сильного и слабого спокойствия**. Данные требования снимают проблему Z-системы. Рассмотрим их.

Правило **сильного спокойствия** гласит, что уровни безопасности субъектов и объектов никогда не меняются в ходе системной операции. Реализовав это правило в конкретной системе, можно легко сделать заключение, что описанный выше тип потенциальных проблем никогда не произойдет. Очевидным недостатком такой реализации в системе является потеря гибкости при выполнении операций.

Правило **слабого спокойствия** гласит, что уровни безопасности субъектов и объектов никогда не меняются в ходе системной операции таким образом, чтобы нарушить заданную политику безопасности. Это правило может потребовать, чтобы субъекты и объекты воздерживались от действий в период времени, когда меняются их уровни безопасности. Например, может потребоваться, чтобы уровень безопасности объекта никогда не менялся в то время, как к нему обращается некоторый субъект. Однако, если операция чередуется с изменением уровня безопасности, не вызывающего нарушения безопасности (например, субъект повышает свой уровень с секретного до совершенно секретного в ходе выполнения операции чтения неклассифицированного объекта), то правило слабого спокойствия будет по-прежнему соблюдено.

Фактически система Z описывает алгебру моделей, самой строгой из которых (основание) является БЛМ с сильным спокойствием (ни один субъект модели не может изменить свою классификацию), а самой слабой (вершина) - БЛМ в классической формулировке, без ограничений для субъектов на изменение классификации.



#### ***12.4. Специализированные модели***

Как было отмечено в предыдущем параграфе, одним из недостатков, являющимся логическим следствием достоинства простоты БЛМ, является ее слишком большая абстрактность. С точки зрения требований пользователей, в реальных приложениях ограничения, накладываемые БЛМ, оказываются слишком строгими. Введение в модель доверенных процессов, позволяющих частично решить данную проблему, не является достаточным. С другой стороны, недостатком БЛМ, не рассмотренным нами ранее, является отсутствие в модели поддержки многоуровневых объектов (например, наличие несекретного параграфа в секретном файле данных) и отсутствие зависящих от приложения правил безопасности. С целью устранения данных недостатков при проектировании системы передачи военных сообщений (MMS) Лендвером и МакЛином была разработана модель MMS.

##### ***12.4.1. Модель MMS***

В модели MMS используются следующие понятия.

**Классификация** – обозначение, накладываемое на информацию, отражающее ущерб, который может быть причинен неавторизованным доступом; включающее уровни: TOP SECRET, SECRET и т.д. и множество меток ("CRYPTO", "NUCLEAR" и т.д.). Множество классификаций и отношение между ними образуют решетку.

**Степень доверия пользователю** – уровень благонадежности персоны. Каждый пользователь имеет степень доверия, и операции, производимые системой для данного пользователя, могут проверить степень доверия пользователю и классификацию объектов, с которыми он оперирует.

**Пользовательский идентификатор** – строка символов, используемая для того, чтобы отметить пользователя системы. Для использования системы пользователь должен предъявить ей пользовательский идентификатор, и система должна провести аутентификацию пользователя. Данная процедура

называется login. Каждый пользователь должен иметь уникальный идентификатор.

**Пользователь** - персона, уполномоченная для использования системы.

**Роль** – работа, исполняемая пользователем (например, пользователь, имеющий право удалять, распространять или понижать классификацию объектов). Пользователь всегда ассоциирован как минимум с одной ролью в некоторый момент времени, и он может **менять роль в течение сессии**. Для действий в данной роли пользователь должен быть уполномочен. Некоторые роли могут быть связаны только с одним пользователем в данный момент времени (например, распространитель). С любой ролью связана способность выполнения определенных операций.

**Объект** – одноуровневый блок информации. Это минимальный блок информации в системе, который имеет классификацию. Объект не содержит других объектов, он не многоуровневый.

**Контейнер** – многоуровневая информационная структура. Имеет классификацию и может содержать объекты (каждый со своей классификацией) и (или) другие контейнеры. Файл – это контейнер. Некоторые структуры файла могут быть контейнерами. Различие между объектом и контейнером базируется на типе, а не на текущем содержимом: если один из файлов данного типа является контейнером, то все остальные файлы данного типа являются контейнерами, даже если некоторые из них содержат только объекты или пусты. Устройства такие, как диски, принтеры, ленты, сетевые интерфейсы и пользовательские терминалы - контейнеры.

**Сущность** – Объект или Контейнер.

**Требование Степени Доверия Контейнеров** – атрибут некоторых контейнеров. Для некоторых контейнеров важно требовать минимум степени доверия, то есть пользователь, не имеющий соответствующего уровня благонадежности, не может просматривать содержимое контейнера. Такие контейнеры помечаются соответствующим атрибутом (CCR). Например, пользователь, имеющий степень доверия CONFIDENTAL, не может

просматривать CONFIDENTAL параграф сообщения, помеченного TOP SECRET, если оно содержится в CCR контейнере. Если пользователь должен иметь возможность просматривать данное сообщение, контейнер не должен быть помечен как CCR.

**Идентификатор (ID)** – имя сущности без ссылки на другие сущности, например, имя файла есть идентификатор этого файла. Обычно все сущности имеют идентификатор.

**Ссылка** на сущность **Прямая**, если это идентификатор Сущности.

**Ссылка** на сущность **Косвенная**, если это последовательность двух или более имен Сущностей (из которых только первая – идентификатор). Пример – "текущее сообщение, первый абзац, вторая строка".

**Операция** – функция, которая может быть применена к сущности. Она может позволять просматривать или модифицировать сущность. Некоторые операции могут использовать более одной сущности (пример – операция копирования).

**Множество Доступа** – множество троек (**Пользовательский Идентификатор** или Роль, Операция, Индекс операнда), которые связаны с сущностью. Операция, которая может быть специфицирована для особых сущностей, зависит от типа данной сущности. Если операция требует более одного операнда, индекс операнда специфицирует позицию, на которой ссылка на данный операнд может появиться в операции.

**Сообщение** – особый тип, реализуемый в MMS. Сообщение является контейнером. Сообщение включает поля Куда, Откуда, Время, предмет, текст, автор. Чертежные сообщения включают поле чертежа.

### **Неформальная модель MMS**

Пользователь получает доступ к системе только после прохождения процедуры login. Для этого пользователь предоставляет системе *Пользовательский идентификатор*, и система производит аутентификацию, используя пароли, отпечатки пальцев или другую адекватную технику. После успешного прохождения аутентификации *Пользователь* запрашивает у

системы *Операции* для использования функций системы. *Операции*, которые *Пользователь* может запросить у системы, зависят от его *ID* или *Роли*, для которой он авторизован: с использованием *Операций* *Пользователь* может просматривать или модифицировать *Объекты* или *Контейнеры*. Система реализует ограничения, описанные ниже.

### **Предположения безопасности**

Пользователь всегда может скомпрометировать информацию, к которой он имеет законный доступ. Таким образом, надо сформулировать предположения безопасности, которые могут быть выполнены только пользователями системы.

**A1.** Офицер безопасности системы присваивает уровни доверия, классификацию устройств и множества ролей корректно.

**A2.** Пользователь вводит корректную классификацию, когда изменяет, объединяет или переклассифицирует информацию.

**A3.** Пользователь классифицирует сообщения и определяет множества доступа для сущностей, которые он создает, так, что только пользователь с требуемой благонадежностью может просматривать информацию.

**A4.** Пользователь должным образом контролирует информацию объектов, требующих благонадежности.

### **Ограничения безопасности**

Ограничения безопасности, в отличие от предположений безопасности, должны поддерживаться не пользователями системы, а непосредственно компьютерной системой.

**B1.** Авторизация – пользователь может запрашивать операции над сущностями, только если пользовательский идентификатор или текущая роль присутствуют во множестве доступа сущности вместе с этой операцией и со значением индекса, соответствующим позиции операнда, в которой сущность относят в требуемой операции.

**B2. Классификационная иерархия** – классификация контейнера всегда больше или равна классификации сущностей, которые он содержит.

**B3. Изменения в объектах** – информация, переносимая из объекта, всегда наследует классификацию данного объекта. Информация, вставляемая в объект, должна иметь классификацию ниже классификации этого объекта.

**B4. Просмотр** – пользователь может просматривать (на некотором устройстве вывода) только сущности с классификацией меньше, чем классификация устройства вывода и степень доверия к пользователю (данное ограничение применяется к сущностям, адресуемым прямо или косвенно).

**B5. Доступ к контейнерам, требующим степени доверия** – пользователь может получить доступ к косвенно адресованной сущности внутри контейнера, требующего степени доверия, только если его степень доверия не ниже классификации контейнера.

**B6. Преобразование косвенных ссылок** – пользовательский идентификатор признается законным для сущности, к которой он обратился косвенно, только если он авторизован для просмотра этой сущности через ссылку.

**B7. Требование меток** – сущности, просмотренные пользователем, должны быть помечены его степенью доверия.

**B8. Установка степеней доверия, ролей, классификации устройств** – только пользователь с ролью офицера безопасности системы может устанавливать данные значения. Текущее множество ролей пользователя может быть изменено только офицером безопасности системы или самим пользователем.

**B9. Понижение классификации информации** – никакая классифицированная информация не может быть понижена в уровне своей классификации, за исключением случая, когда эту операцию выполняет

пользователь с ролью "пользователь, уменьшающий классификацию информации".

**B10. Уничтожение информации** – операция уничтожения информации проводится только пользователем с ролью "пользователь, уничтожающий информацию".

### ***12.5. Проблемы моделей предоставления прав***

Наряду с неоспоримыми достоинствами моделей предоставления прав, выражающимися в их интуитивной понятности и возможности реализации с высокой степенью точности, данные модели имеют ряд недостатков.

**В моделях предоставления прав возможно образование скрытых каналов утечки информации.** Таким образом, несмотря на кажущуюся простоту реализации систем предоставления прав, перекрытие каналов утечки информации является нетривиальной задачей. При анализе защищенных вычислительных систем, построенных по принципу предоставления прав, необходим тщательный анализ каналов утечки информации. Для систем высокой степени доверия данный пункт отражен в требованиях к системе.

Анализ скрытых каналов утечки информации базируется обычно на принципах анализа потоков данных в программном обеспечении (данные принципы разработаны Д. Деннинг), контроля совместно используемых ресурсов, которые могут быть применены для организации скрытых каналов утечки информации (каналы утечки информации на основе хранения) и использования программами таймеров (временные каналы утечки информации).

Хотя каналы утечки информации нетрудно обнаружить, их обычно находят уже после того, как система синтезирована. Как следствие, их ликвидация может быть затруднительна.

## 12.6. Информационные модели

Информационные модели определяют **ограничения на отношение ввода/вывода системы**, которые достаточны для реализации системы. Данные модели накладывают *ограничения на интерфейс программных модулей* системы с целью достижения безопасной реализации. При этом подробности реализации определяются разработчиком системы. Данные модели являются результатом применения теории информации к проблеме безопасности систем. К информационным моделям относятся модели невмешательства и невыводимости. Достоинствами данного типа моделей, в отличие от моделей предоставления прав, являются:

- ☐ отсутствие в них скрытых каналов утечки информации;
- ☐ естественность их использования для реализации сетевых защищенных вычислительных систем.

Теория данных математических моделей бурно развивается в настоящее время.

### 12.6.1. Модель невмешательства

**Невмешательство** - ограничение, при котором ввод высокоуровневого пользователя не может смешиваться с выводом низкоуровневого пользователя.

Модель невмешательства рассматривает систему, состоящую из четырех объектов: высокий ввод (high-in), низкий ввод (low-in), высокий вывод (high-out), низкий вывод (low-out).

Рассмотрим систему, вывод которой пользователю  $u$  определен функцией  $\text{out}(u, \text{hist.read}(u))$ , где  $\text{hist.read}(u)$  – история ввода системы (traces), чей последний ввод был  $\text{read}(u)$  (команда чтения, исполненная пользователем  $u$ ). Для определения безопасности системы необходимо определить термин очищения (*purge*) историй ввода, где ***purge* удаляет команды**, исполненные пользователем, чей уровень безопасности не доминирует над уровнем безопасности  $u$ . **Функция  $\text{clearance}(u)$  – определяет степень доверия к пользователю**

*Определение:*  $\text{purge}$  – функция  $\text{users} \times \text{traces} \rightarrow \text{traces}$  такая, что:

- $\text{purge}(u, \langle \rangle) = \langle \rangle$ , где  $\langle \rangle$  - пустая история ввода;
- $\text{purge}(u, \text{hist.command}(w)) = \text{purge}(u, \text{hist.command}(w))$ , если  $\text{command}(w)$  – ввод, выполненный пользователем  $w$ ;  $\text{clearance}(u) \geq \text{clearance}(w)$ ;
- $\text{purge}(u, \text{hist.command}(w)) = \text{purge}(u, \text{hist})$ , если  $\text{command}(w)$  - ввод, выполненный пользователем  $w$ ;  $\text{clearance}(u) < \text{clearance}(w)$ .

Система удовлетворяет требованию невмешательства, если и только если для всех пользователей  $u$ , всех историй  $T$  и всех команд вывода  $c$   $\text{out}(u, T.c(u)) = \text{out}(u, \text{purge}(u, T).c(u))$ .

Для того, чтобы проверить, удовлетворяет ли система требованиям невмешательства, было разработано множество условий («unwinding conditions»), выполнение которых достаточно для поддержки невмешательства в модели машины состояний. Хотя верификация модели невмешательства труднее, чем верификация БЛМ, после нее в системе не остается скрытых каналов утечки информации. Модель невмешательства ближе к интуитивному понятию безопасности, чем БЛМ.

При сравнении модели невмешательства с БЛМ можно отметить:

1. БЛМ слабее, чем модель невмешательства, так как модель невмешательства запрещает многие скрытые каналы, которые остаются при реализации примитивной БЛМ.

2. Модель невмешательства слабее, чем БЛМ в том, что она разрешает низкоуровневым пользователям копировать один высокоуровневый файл в другой высокоуровневый, что БЛМ запрещает, так как при этом нарушается ее безопасность по чтению.

Было показано, что для определенных систем модель невмешательства особенно хороша в том, что если последовательность ввода  $X$  не смешивается с последовательностью вывода  $Y$ , и  $X$  независима от ввода других пользователей, то  $I(X, Y) = 0$ , где  $I(X, Y)$  – взаимная для  $X$  и  $Y$  информация и представляет собой поток информации от  $X$  к  $Y$ .



### **12.6.2. Модель невыводимости**

Рассмотрим модель невыводимости, также базирующуюся на рассмотрении информационных потоков в системе. Модель невыводимости выражается в терминах пользователей и информации, связанных с одним из двух возможных уровней секретности (высокий и низкий).

Система считается *невыводимо безопасной*, если пользователи с низкими уровнями безопасности не могут получить информацию с высоким уровнем безопасности в результате любых действий пользователей с высоким уровнем безопасности. Другими словами, в таких системах утечка информации не может произойти в результате посылки высокоуровневыми пользователями высокоуровневой информации к низкоуровневым пользователям. Интуитивно это определение относится не к информационным потокам, а к разделению информации. Однако такое определение безопасности не предохраняет информацию высокоуровневых пользователей от просмотра низкоуровневыми пользователями. Данное определение требует, чтобы низкоуровневые пользователи не *были способны использовать доступную им информацию для получения высокоуровневой информации* (это объясняет, почему определение названо невыводимостью).

Многие исследователи предложили рассматривать понятие посылки и интерпретации сигнала шире, чем понятия чтения и записи в модели БЛМ. Иными словами, чтение и запись рассматриваются в контексте этой модели как явные операции, вызываемые пользователями компьютерной системы, и выполняются определенной автоматизированной последовательностью вычислительных действий. Поскольку определение невыводимости, данное выше, неформально, то обычно появляется необходимость представить понятие невыводимости в более строгой форме. Это делается во избежание двусмысленности или других ошибок. Поэтому рассмотрим это понятие, используя модель машины состояний с ограниченными атрибутами, как средство для более точного определения невыводимости. В частности, машина

состояний будет обладать детерминированным поведением и состоять из следующих частей:

- набор пользователей с высоким или низким уровнем безопасности;
- набор возможных последовательностей системных вводов информации от пользователей и выходных реакций системы.

Допустим, что машина принимает ввод от высоко- и низкоуровневых пользователей, обрабатывает эти вводы некоторым заданным образом и затем выдает на выходах к высоко- и низкоуровневым пользователям информацию. Возможно также, что вводят информацию и получают данные вывода одни и те же пользователи. Единственным различием пользователей является то, какой у них уровень безопасности – высокий или низкий.

Если множество вводов от пользователей в машину связано со множеством выводов, получаемых пользователями от машины каким-либо разумным образом (возможно, основываясь на времени их поступления), то тогда можно рассматривать выходную последовательность как трассировку (traces, см. модель невмешательства) системы. Безопасность невыводимости может быть определена в соответствии со множеством всех трассировок системы и множеством вводов и выводов, видимых пользователями.

Точнее, система может быть признана невыводимо безопасной, если для каждой метки безопасности  $x$  и определенной трассировки есть вторая трассировка, показывающая то же поведение, видимое пользователями с меткой безопасности меньшей или равной  $x$ , но не имеющая вводов не меньших или равных  $x$ . Другими словами, высокоуровневые вводы могут всегда быть удалены из трассировки и это не повлияет на то, что видят низкоуровневые пользователи. Можно заметить, что понятие невыводимости не охватывает ситуаций, основанных на концепции "интерпретации информации" в той степени, в которой этого можно было ожидать. Данный недостаток устраняется с помощью ограничения понятия составляющих вводов и выводов модели.

Например, предположим, что некоторая система принимает вводы и обеспечивает выводы для множества высоко- и низкоуровневых пользователей.

Каждый пользователь связан с определенным взглядом на систему (например, видимые вводы и выводы) и может получить информацию, интерпретируя видимое ему поведение. Если система является невыводимо безопасной, то низкоуровневые пользователи не должны получить новой информации, если на вводе системы есть дополнительные высокоуровневые пользователи. Кроме этого, если низкоуровневые пользователи могут получить определенную информацию, основываясь на видимом ими поведении, то удаление высокоуровневых пользователей не должно изменить получаемой низкоуровневыми пользователями информации.

### ***12.7. Вероятностные модели***

Модели этого типа исследуют вероятность преодоления системы защиты за определенное время  $T$ . К достоинствам моделей данного типа можно отнести числовую оценку стойкости системы защиты. К недостаткам – изначальное допущение того, что система защиты может быть вскрыта. Задача модели данного типа – минимизация вероятности преодоления системы защиты.

#### ***12.7.1. Игровая модель***

Игровая модель системы защиты строится по следующему принципу. Разработчик создает первоначальный вариант системы защиты. После этого злоумышленник начинает его преодолевать. Если к моменту времени  $T$ , в который злоумышленник преодолел систему защиты, у разработчика нет нового варианта, система защиты преодолена. Если нет – процесс продолжается. Данная модель описывает процесс эволюции системы защиты в течение времени.

### 12.7.2. Модель системы безопасности с полным перекрытием

Система, синтезированная на основании модели безопасности с полным перекрытием, **должна иметь по крайней мере одно средство** для обеспечения безопасности на каждом возможном пути проникновения в систему (рис.12.1.).

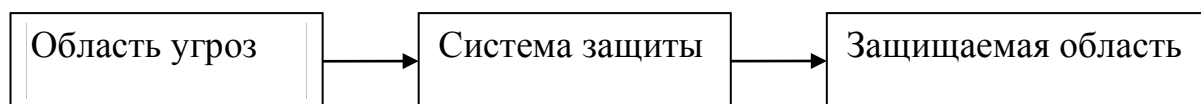


Рис. 12.1. Модель системы защиты с полным перекрытием

В модели точно определяется каждая область, требующая защиты, оцениваются средства обеспечения безопасности с точки зрения их эффективности и их вклад в обеспечение безопасности во всей вычислительной системе. Считается, что несанкционированный доступ к каждому из набора защищаемых объектов  $O$  сопряжен с некоторой величиной ущерба, и этот ущерб может быть определен количественно.

С каждым объектом, требующим защиты, связывается некоторое множество действий, к которым может прибегнуть злоумышленник для получения несанкционированного доступа к объекту. **Можно попытаться перечислить все потенциальные злоумышленные действия по отношению ко всем объектам безопасности для формирования набора угроз  $T$ , направленных на нарушение безопасности.** Основной характеристикой набора угроз является вероятность проявления каждого из злоумышленных действий. В любой реальной системе эти вероятности можно вычислить с ограниченной степенью точности.

Рассмотрим более строгое описание вероятностной модели, построенное на основе теории множеств. Множество отношений объект–угроза образуют двухдольный граф (рис. 12.2.), в котором ребро  $\langle t_i, o_j \rangle$  существует тогда и только тогда, когда  $t_i$  ( $\forall t_i \in T$ ) является средством получения доступа к объекту  $o_j$  ( $\forall o_j \in O$ ). Связь между объектами и угрозами типа "один ко многим", то есть одна угроза может распространяться на любое число объектов и объект может быть уязвим со стороны более чем одной угрозы. Цель защиты состоит в том, чтобы

перекрыть каждое ребро графа и воздвигнуть барьер для доступа по этому пути.

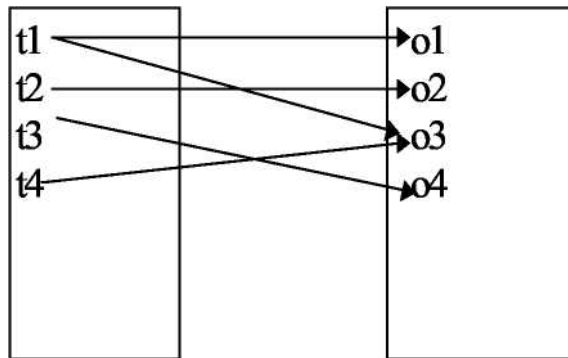


Рис. 12.2. Граф объект – угроза

Завершает модель третий набор, включающий средства безопасности  $M$ , которые используются для защиты информации в вычислительной системе. Идеально, каждое  $m_k$  ( $\forall m \in M$ ) должно устранять некоторое ребро  $\langle t_i, o_j \rangle$  из графа на рисунке 12.2. Набор  $M$  средств обеспечения безопасности преобразует двухдольный граф в трехдольный. В защищенной системе все ребра представляются в виде  $\langle t_i, m_k \rangle$  и  $\langle m_k, o_j \rangle$ . Любое ребро в форме  $\langle t_i, o_j \rangle$  определяет незащищенный объект. Одно и то же средство обеспечения безопасности может перекрывать более одной угрозы и (или) защищать более одного объекта. Отсутствие ребра  $\langle t_i, o_j \rangle$  не гарантирует полного обеспечения безопасности (хотя наличие такого ребра дает потенциальную возможность несанкционированного доступа, за исключением случая, когда вероятность появления  $t_i$  равна нулю).

Рассмотрим базовую систему безопасности Клеменса, представляющую собой пятикортежный набор  $S = \{O, T, M, V, B\}$ , где  $O$  – набор защищаемых объектов;

$T$  – набор угроз;

$M$  – набор средств обеспечения безопасности;

$V$  – набор уязвимых мест – отображение  $T \times O$  на набор упорядоченных пар  $V_i = \langle t_i, o_j \rangle$ , представляющих собой пути проникновения в систему;

$B$  – набор барьеров – отображение  $V \times M$  или  $T \times O \times M$  на набор упорядоченных троек  $\langle t_i, o_j, m_k \rangle$ , представляющих собой точки, в которых требуется осуществлять защиту в системе.

Система с полным перекрытием – это система, в которой имеются средства защиты на каждый возможный путь проникновения. В такой системе  $\langle t_i, o_j \rangle \in V$  предусматривает  $\langle t_i, o_j, m_k \rangle \in B$ . Если это условие не соблюдено, то  $O_j$  не защищено для некоторого  $j$ .

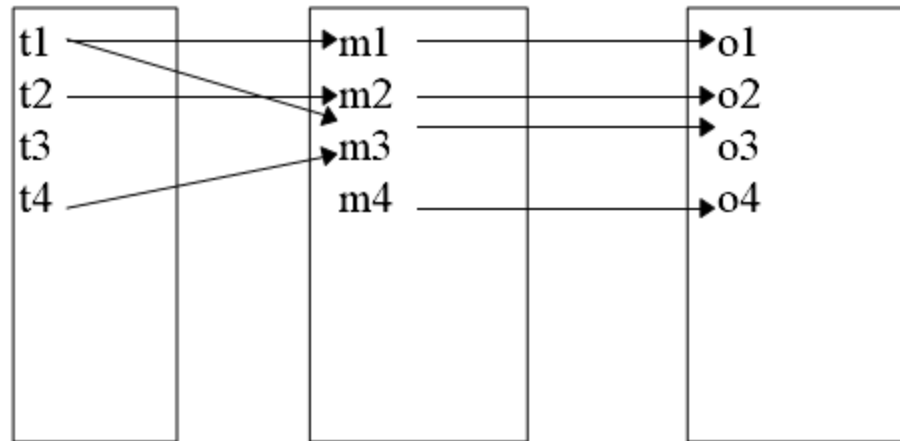


Рис. 12.3. Граф угроза - защита - объект

Таким образом, можно сделать следующие выводы. Основное преимущество вероятностных моделей состоит в **возможности численного получения оценки степени надежности системы защиты информации**. Данные модели не специфицируют непосредственно механизмы защиты информации, а могут использоваться только в сочетании с другими типами моделей систем защиты информации. При анализе систем защиты информации модели данного типа позволяют оценить вероятность преодоления системы защиты и степень ущерба системе в случае преодоления системы защиты. При синтезе систем защиты информации данный подход полезен тем, что позволяет минимизировать накладные расходы (ресурсы вычислительной системы) для реализации заданного уровня безопасности. Модели данного типа могут использоваться при анализе эффективности внешних по отношению к защищаемой системе средств защиты информации. Ярким примером применимости данной модели является анализ на ее основе вероятности вскрытия за конечный временной промежуток средств защиты, предлагаемых для системы MS-DOS. Для систем защиты, построенных на основании других моделей, данная

модель может применяться для анализа эффективности процедуры идентификации/аутентификации.

### **12.8 .Модели контроля целостности**

. Данный параграф начинается с рассмотрения модели, аналогичной БЛМ, используемой для синтеза механизмов контроля целостности информации в системе, модели Биба. Данное рассмотрение заканчивается рассмотрением попытки объединения моделей БЛМ и Биба в одну модель. Далее мы рассмотрим модель Кларка – Вилсона (КВМ). Эта модель является примером неформального выражения политики безопасности. Данная модель сформулирована в виде набора неформальных правил, и хотя в литературе она названа моделью безопасности, ее скорее можно назвать политикой контроля целостности.

#### **12.8.1. Модель Биба**

При рассмотрении БЛМ было показано, что важность или чувствительность субъектов и объектов повышается с ростом в иерархии уровней безопасности. При рассмотрении моделей контроля целостности запись наверх может представлять угрозу в том случае, если субъект с низким уровнем безопасности искажает или уничтожает данные в объекте, лежащем на более высоком уровне. Поэтому, исходя из задач целостности, можно потребовать, чтобы такая запись была запрещена. Следуя подобным аргументам, можно рассматривать чтение снизу как поток информации, идущий из объекта нижнего уровня и нарушающий целостность субъекта высокого уровня. Поэтому весьма вероятно, что и такое чтение необходимо запретить.

Два этих наблюдения сделал в середине семидесятых Кен Биба. Они были последовательно внесены в модель безопасности, которая с тех пор называется моделью целостности Биба (или просто моделью Биба). Биба выразил свою модель таким же способом, каким была выражена БЛМ, за тем исключением, что правила его модели являются полной противоположностью правилам БЛМ. В этом параграфе рассмотрим три



вариации модели Биба: **мандатную модель целостности, модель понижения уровня субъекта и модель понижения уровня объекта.** Фактически, общий термин "модель Биба" используется для обозначения любой или сразу всех трех моделей. Для мандатной модели целостности предлагается формальное описание и дается пример системы, удовлетворяющей модели Биба для иллюстрации определения.

**Мандатную модель целостности Биба** часто называют инверсией БЛМ. Это довольно точное название, поскольку основные правила этой модели просто переворачивают правила БЛМ. Мы будем ссылаться на эти правила как **"нет чтения снизу" (NRD)** и **"нет записи вверх" (NWU)**, и определим их в терминах субъектов, объектов, и нового типа уровней безопасности - *уровней целостности*, над которыми может быть введено отношение преобладания.

Правило NRD мандатной модели целостности Биба определяется как **запрет субъектам на чтение информации из объекта с более низким уровнем целостности.** NRD является полной противоположностью правила NRU БЛМ, за исключением того, что здесь используются уровни целостности, а не безопасности, как в БЛМ. Правило NWU мандатной модели целостности Биба определяется как **запрет субъектам на запись информации в объект с более высоким уровнем целостности.** Это правило является полной противоположностью правилу NWD БЛМ для случая уровней целостности, а не безопасности.

Одним из преимуществ этой модели является то, что она унаследовала многие важные характеристики БЛМ, включая ее простоту и интуитивность. Это значит, что проектировщики реальных систем могут легко понять суть этих правил и использовать их для принятия решений при проектировании. Кроме того, поскольку мандатная модель целостности Биба, подобно БЛМ, основана на простой иерархии, ее легко объяснить и изобразить пользователям системы.

С другой стороны, модель представляет собой очевидное противоречие с правилами NRU и NWD. Это значит, что если необходимо построить систему, которая предотвращает угрозы как секретности, так и целостности, то **одновременное использование правил моделей БЛМ и Биба может привести к ситуации, в которой уровни безопасности и целостности будут использоваться противоположными способами.**

Рассмотрим формальное описание модели Биба. Для этого опишем простые математические конструкции, которые помогут описать различные правила, составляющие мандатную модель целостности Биба.

Начнем с представления множества субъектов и объектов. Уровни целостности субъекта или объекта  $x$  обозначаются как уровень ( $x$ ), и для них введено отношение *преобладания*. Используя эти определения, сформулируем правила NRD и NWU мандатной модели целостности Биба в терминах булевой функции *разрешить*:

NRD:  $\forall s \in \text{субъекты}, o \in \text{объекты}:$

*разрешить* ( $s, o$ , чтение), если и только если уровень ( $o$ ) *преобладает* уровень ( $s$ ).

Данный тип определения предусматривает условия, при которых функция *разрешить* принимает значение истинно. Определение утверждает, что для всех определенных субъектов и объектов операция чтения разрешена только в том случае, если выполняется условие преобладания. Правило NWU просто переворачивает использование отношения преобладания, как показано в следующем определении:

NWU:  $\forall s \in \text{субъекты}, o \in \text{объекты}:$

*разрешить* ( $s, o$ , запись)  $o \text{ clearance}(s) \geq \text{classification}(o)$ .

Это определение утверждает, что для всех субъектов и объектов операция записи разрешается только в том случае, если выполняется условие преобладания. Подобие определения этих двух правил правилам модели БЛМ может предоставить удобный способ для проектировщиков системы

предусмотреть возможность переконфигурирования правил БЛМ таким образом, чтобы поддерживать мандатную модель целостности Биба.

### **Модель понижения уровня субъекта**

Вторая модель Биба заключается в **небольшом ослаблении правила чтения снизу**. Мандатная модель целостности не позволяет субъектам с высокой целостностью читать информацию из объектов с более низкой целостностью. Это правило гарантирует, что информация из объекта с **низкой целостностью не нарушит целостности субъекта**. Однако в модели понижения уровня субъекта *ему разрешается осуществлять чтение снизу, но в результате такого чтения уровень целостности субъекта понижается до уровня целостности объекта*.

Мотивом для введения такого правила может являться то, что субъекты с высокой целостностью рассматриваются как "чистые". Когда к чистому субъекту попадает информация из менее чистого источника, субъект "портится", и его уровень целостности должен быть соответственно изменен.

Одной из характеристик этой модели является то, что она не накладывает никаких ограничений на то, что субъект может прочесть. Если, например, субъект не должен никогда переходить на более низкий уровень целостности, то не следует использовать эту модель, поскольку она может привести к такому нарушению. Если все же эта модель реализована в реальной системе, то необходимо создание некоторых дополнительных мер, предупреждающих субъекта о возможных последствиях выполнения таких операций чтения перед тем, как они будут выполнены.

Следует также заметить, что модель подразумевает **монотонное изменение уровней целостности субъектов**. То есть, уровни целостности субъектов или остаются неизменными, или снижаются. Иными словами, целостность субъекта может остаться прежней или ухудшиться, поскольку модель не предусматривает механизмов повышения уровня целостности субъекта.

### **Модель понижения уровня объекта**

Последний тип модели Биба представляет собой ослабление правила для записи наверх, то есть вместо полного запрета на запись наверх эта модель разрешает такую запись, но **снижает уровень целостности объекта до уровня целостности субъекта, осуществлявшего запись**. Мотивы для такого правила те же, что и в модели понижения уровня субъекта.

Данная модель, подобно предыдущей, не накладывает никаких ограничений на то, что субъект может читать или писать. Поэтому ситуации, когда искажения объекта и понижение его уровня целостности могут вызвать серьезные последствия, не допускают использования этой модели. Например, критическая база данных, включающая данные, целостность которых имеет предельно высокое значение, не может быть реализована на основании этой модели. Если данная модель используется в реальной системе, то необходимо возложить на субъекты ответственность за деградацию объектов с высокой целостностью. Для реализации этого потребуется использование дополнительных средств обработки.

В данной модели происходит монотонное снижение уровня целостности объектов. В этой модели, как и в модели понижения уровней субъектов, не предусмотрено никаких механизмов для повышения уровня целостности объекта. Возможно совместное использование моделей понижения уровня субъекта и объекта в одной системе.

Поскольку модель Биба так похожа на БЛМ, то она обладает большинством достоинств и недостатков этой модели. Например, обе модели просты и интуитивны и могут быть выражены простыми правилами (NRD и NWU). Кроме того, обе модели способствуют введению условий спокойствия для обеспечения того факта, что изменение меток не нарушит безопасности системы (это не относится к двум моделям понижения уровня Биба).

Однако модель Биба также обладает многими проблемами, присущими БЛМ. Например, использование модели Биба в распределенных системах может привести **к двунаправленному потоку информации при удаленном чтении**. Подобным образом, при отсутствии правил спокойствия, для модели

Биба возникает эффект системы  $Z$ , описанный ранее. В практическом применении модель Биба слишком сильно полагается на понятие доверенных процессов, то есть проблема необходимости создания **доверенных процессов** для повышения или понижения целостности субъектов или объектов является весьма существенной. Эта критика последовала за критикой доверенных процессов в БЛМ.

В качестве дополнительной критики модели Биба можно упомянуть то, что она не предусматривает механизмов повышения целостности, что ведет к **монотонному снижению целостности системы**. Помимо этого, многие исследователи критиковали модель Биба за то, что она использует целостность как некую меру и ставили под сомнение то, что понятие "большей целостности" имеет какой-либо смысл. Их аргументом было то, что целостность субъектов и объектов следует рассматривать как двоичный атрибут, который или есть, или нет. В качестве примера они приводили утверждение, что компьютерная программа работает или правильно, или неправильно. С точки зрения логики это имеет смысл, но можно представить себе некоторые уровни правильности (например, минимальные синтаксические ошибки в программе могут снизить ее правильность намного меньше, чем значительный семантический изъян).

### **Объединение моделей безопасности**

Теперь, после рассмотрения БЛМ и модели Биба, можно рассмотреть вопрос о том, как можно объединить на практике две и более моделей. Другими словами, рассмотрим, как проектировщики и разработчики могут использовать различные модели безопасности при проектировании одной системы. При этом появляются вопросы о том, имеют ли эти модели взаимные противоречия, дополняют ли они друг друга, можно ли их реализовать, используя одинаковые конструкции, и т.п. Для рассмотрения вопроса об объединении двух моделей можно определить два подхода.

1. Различные модели могут быть выражены одной универсальной структурой так, что **их правила работают в пределах одной модели**

**безопасности.** Обычно для этого требуется создание весьма общей структуры, которая будет достаточно полна для описания всех концепций различных моделей. Так, например, если модели БЛМ и Биба используются при разработке одной системы, то первым шагом будет объединение этих двух моделей в одну новую модель, которая будет охватывать их необходимые элементы.

2. Модели могут использоваться **отдельно**, и может быть проведен неформальный анализ соответствующих подходов реализации каждой модели. Это позволит сохранять независимость между отдельными моделями и произвольно объединять модели в зависимости от различных требований системы. Так, например, если модели БЛМ и Биба используются при разработке одной системы, то необходимо рассмотреть соответствующие реализации этих моделей, чтобы определить возможность их объединения. Для каждого из этих подходов существуют свои аргументы. Создание общей универсальной структуры предоставляет средства для определения потенциальной несовместимости или избыточности различных моделей. Однако это также приводит к такой реализации модели, которая может быть запутанной и сложной в использовании. Анализ реализаций, с другой стороны, может оказаться менее полезным при сравнении логических свойств различных моделей, но он полезен при практической реализации систем, основанных на этих моделях. Приведем пример использования второго подхода для иллюстрации возможности объединения моделей БЛМ и Биба.

Для этого рассмотрим несколько технических вопросов, касающихся объединения моделей БЛМ и Биба в системе, которая должна решать как проблемы секретности, так и целостности.

Первым подходом к решению этой проблемы может являться создание системы, использующей одну политику безопасности, полученную в результате объединения правил моделей БЛМ и Биба, то есть такая политика

будет включать в себя правила из обеих моделей. Для этого потребуется, чтобы субъектам и объектам были назначены **различные уровни безопасности и целостности**, что позволит проводить оценку каждой модели отдельно. Однако для этого потребуется **создание и управление двумя различными наборами уровней** в соответствии с двумя различными наборами правил.

Другим подходом может быть **логическое объединение** правил этих моделей в одну модель безопасности, основанную на одном уровне как для безопасности, так и для целостности. Это приведет к правилам равного чтения и равной записи, которые удовлетворяют обеим моделям, но значительно снизится гибкость операций чтения и записи в компьютерной системе, использующей такую модель.

Еще один подход основан **на использовании одного уровня как для целостности, так и для безопасности**. В этом случае уровень безопасности обрабатывается в соответствии с БЛМ, но с помощью ловкого подхода будет обеспечиваться также и контроль целостности. Этот подход подразумевает размещение субъектов и объектов с высокой целостностью на нижней ступени иерархии безопасности. Поскольку субъекты и объекты с высокой целостностью находятся внизу иерархии, а компоненты с низкой целостностью - наверху иерархии, то правила NRU и NWD имитируют мандатную модель целостности Биба в структуре БЛМ, то есть чтение сверху в иерархии БЛМ является чтением снизу в иерархии модели Биба. Аналогично, запись наверх в БЛМ является записью вниз в модели Биба.

На практике это позволяет разместить важные системные файлы и администраторов в нижней части иерархии БЛМ. Это защищает целостность важных объектов от обычных пользователей, поскольку правило NWD не позволяет им осуществлять запись в системные файлы. Кроме этого, если рассматривать исполнение как чтение, то администраторы не смогут исполнять программы вне высшего уровня целостности (то есть нижнего



уровня иерархии БЛМ). Это обеспечивает дополнительную защиту целостности для администраторов.

Данная схема обеспечивает защиту системных файлов и администраторов от троянских коней. Если троянский конь находится на одном из верхних уровней, он никогда не сможет исказить системные файлы за счет правила NWD. Таким образом осуществляется защита целостности от троянских коней. Очевидно, такое объединение моделей осуществляет защиту секретности для верхних уровней определенной иерархии и защиту целостности для нижних уровней.

### **12.8.2. Модель Кларка-Вилсона**

В 1987 г. Дэвид Кларк и Дэвид Уилсон представили модель целостности, которая существенно отличалась от уровне-ориентированных моделей безопасности БЛМ и Биба. Созданию этой модели, которая известна как модель Кларка-Вилсона (КВМ), способствовал анализ методов управления коммерческими организациями целостностью своих бумажных ресурсов в неавтоматизированном офисе, то есть был рассмотрен ряд хорошо известных методов учета и сделана попытка распространения их на случай компьютерных приложений. Получившаяся модель целостности представляет собой руководство разработчикам и проектировщикам компьютерных систем для обеспечения целостности определенных вычислительных ресурсов.

Модель КВМ выражается в терминах **набора правил функционирования и обслуживания данного компьютерного окружения или приложения**. Эти правила вырабатываются для обеспечения уровня защиты целостности для некоторого заданного подмножества данных в этом окружении или приложении. Критическим понятием модели КВМ является то, что эти правила выражаются с использованием так называемых правильно сформированных транзакций, в которых субъект инициирует



последовательность действий, которая выполняется управляемым и предсказуемым образом.

В этом параграфе представим несколько исходных концепций модели КВМ, включая правильно сформированные транзакции, и опишем основные правила модели КВМ. Определение каждого правила включает обсуждение проблем практической реализации. Затем следует оценка практического применения модели КВМ. В последнем разделе обсуждается возможность объединения моделей КВМ и Биба.

### **Исходные концепции модели КВМ**

Представим модель КВМ с помощью строгого описания основных компонентов, включенных в модель. Для облегчения обсуждения мы будем использовать некоторые понятия теории множеств, которые несколько упростят концепции, включенные в оригинальную презентацию модели КВМ.

Модель КВМ выражается в терминах конечного множества, которые мы будем обозначать как **D (для данных)**, которые включают все наборы данных в определенной компьютерной системе. Например, если рассматривается операционная система общего назначения, то D будет обозначать все файлы, структуры и другие хранилища информации, управляемые операционной системой. В предыдущих обсуждениях мы ссылались на такие компоненты, как объекты.

Чтобы различать данные, обладающие и не обладающие целостностью, создатели модели разделили D на два непересекающиеся подмножества, которые называются **ограниченные элементы данных (CDI)** и **неограниченные элементы данных (UDI)**. Это можно изобразить следующими определениями:

$$D = CDI \cup UDI$$

$$CDI \cap UDI = \emptyset.$$

Первое определение показывает, что D является объединением CDI и UDI, а второе определение показывает, что нет элементов, принадлежащих и

CDI, и UDI. Набор D разделен таким образом, потому что мы хотим показать, как может меняться целостность данных, то есть данные, не имеющие целостности и находящиеся поэтому в UDI, могут быть некоторым образом модернизированы, так чтобы иметь целостность и находиться соответственно в CDI. Для упрощения нашего обсуждения мы будем ссылаться на элементы CDI и UDI.

**Субъекты** включены в модель как множество компонентов, которые могут инициировать так называемые **процедуры преобразования**. Процедура преобразования определяется как любая ненулевая последовательность элементарных действий. Элементарное действие, в свою очередь, определяется как переход состояния, который может вызвать изменение некоторых элементов данных. **Например, субъекты могут устранять элементы данных, изменять информацию в элементах данных, копировать их и т.д.** Каждая из этих операций называется процедурой преобразования (или просто ПП), поскольку действительный способ, которым каждая из них выполняется, включает в себя последовательность элементарных действий (например, копирование A в B обычно состоит из таких операций, как чтение A, создание B, запись в B).

Если мы будем ссылаться на множество субъектов как на *субъекты*, то ПП могут быть представлены как функции, ставящие в соответствие субъект и элемент данных с новым элементом данных следующим образом ( $A \times B$  является множеством пар  $(a, b)$ , где  $a \in A$ ,  $b \in B$ ):

ПП:  $\text{субъекты} \times D \rightarrow D$ .

ПП являются просто действиями, которые выполняют над данными субъекты, способные изменить определенные данные. Чтобы проиллюстрировать данное определение, мы можем определить некоторую ПП, называемую *копирование*, который обозначает свойство:  $s \in \text{субъекты}$ ,  $d \in D$ : копирование  $(s, d) = d$ .

**Правила модели KBM**

Используя вышеуказанные понятия, мы можем теперь рассмотреть основные правила, составляющие модель KBM. Модель KBM можно рассматривать как набор из девяти правил, которые мы выразим с помощью представленных выше понятий. В ходе обсуждения предполагается, что наши замечания делаются в соответствии с интересующей нас компьютерной системой. Также предполагается, что правила приняты все вместе, так что любое правило может ссылаться на любое другое правило без каких-либо проблем.

Первое правило гласит, что система должна содержать так называемые *процедуры утверждения целостности (IVP)*. IVP утверждают, что данный CDI имеет надлежащий уровень целостности. Таким образом, IVP можно рассматривать как средство для доказательства целостности CDI.

*Правило 1. В системе должны иметься IVP, утверждающие целостность любого CDI.*

Можно представить себе IVP как некий тип процедуры проверки для утверждения целостности каждого CDI и подтверждения отсутствия целостности каждого UDI. Простейшим примером такой процедуры утверждения является проверка контрольной суммы. При использовании этого подхода вычисляется контрольная сумма некоей хранимой информации, и копии этой информации сравниваются с оригиналом путем проверки соответствующих контрольных сумм. Различия в контрольных суммах сигнализируют о внесении изменений.

Одной из проблем, возникших при попытках реализации этой модели в реальных системах, является неясность того, как такие IVP могут быть сконструированы и реализованы. Например, если множество D содержит программное обеспечение (ПО), и мы используем просмотр кода для гарантии целостности CDI, содержащего это ПО, то возникает вопрос об ограниченности просмотра кода. Это остается широким полем для исследователей компьютерной безопасности.

Второе правило модели KBM гласит, что когда любая ПП применяется к любому CDI, то производимые изменения не должны приводить к снижению целостности этих данных.

*Правило 2. Применение любой ПП к любому CDI должно сохранять целостность этого CDI.*

Это правило можно рассматривать как свойство скрытия применения ПП над CDI, то есть любое применение ПП над CDI не приведет к нарушению целостности CDI.

Обратите внимание, что это правило не указывает, как каждая ПП поддерживает скрытие в пределах множества CDI. Это происходит потому, что ничего не говорится о том, как применение ПП влияет на целостность других CDI. Более сильным способом определения правила является гарантия того, что каждая ПП сохраняет целостность каждого CDI, а не только того CDI, который изменяет данная ПП. Это бы гарантировало, что никакая ПП не влияет на целостность, но это гораздо труднее реализовать.

Третье правило определяет защитное ограничение на то, какие процедуры могут влиять на множество CDI.

*Правило 3. Только ПП может вносить изменения в CDI.*

Другими словами, процедуры и действия, не являющиеся ПП, не могут изменить CDI. Это обеспечивает замкнутость в пределах набора CDI. Обратите внимание на отличия от модели Биба понижения уровня объектов. Модель Биба позволяет субъектам с меньшей целостностью изменять объекты с более высокой целостностью, что нарушает их целостность. Данное правило модели KBM не позволяет субъектам с низкой целостностью, то есть не использующим ПП, изменять объекты с высокой целостностью, то есть CDI. В этом плане модель KBM подобна мандатной модели целостности Биба.

Четвертое правило определяет, какие субъекты могут инициировать ПП над соответствующими данными в CDI.

***Правило 4. Субъекты могут инициировать только определенные ПП над определенными CDI.***

Это правило предполагает, что система должна определять и поддерживать некоторые отношения между субъектами ПП и CDI, так называемые KBM–тройки. Каждая такая тройка определяет возможность данного субъекта применить данную ПП к данному CDI. Например, если (s, t, d) является элементом отношения, то субъекту s разрешается применить ПП t к CDI d. Если же эта тройка не является элементом отношения, то такой тип применения ПП будет запрещен. Это правило гарантирует, что всегда можно определить, кто может изменить CDI и как это изменение может произойти.

Пятое правило накладывает дополнительное ограничение на отношения KBM–троек.

***Правило 5. KBM – тройки должны проводить некоторую соответствующую политику разделения обязанностей субъектов.***

Это правило предусматривает, что компьютерная система определяет такую политику, чтобы не позволять субъектам изменять CDI без соответствующего вовлечения других субъектов. Это предотвращает субъектов от возможности наносить ущерб целостности CDI. Некоторые системы управления конфигурацией предоставляют уровень разделения обязанностей. Например, в некоторых системах разработчики ПО должны представить свои модули на просмотр менеджеру по разработке ПО перед тем, как они смогут включить их в конфигурацию. Этот подход защищает целостность конфигурации ПО. Однако ничто в модели KBM не предотвращает от использования неграмотной политики разделения обязанностей.

Шестое правило предоставляет способ модернизации UDI в CDI.

***Правило 6. Некоторые специальные ПП могут превращать UDI в CDI.***

Это правило позволяет определенным ПП получать на вход UDI и после соответствующего повышения целостности выдавать на выходе CDI.

Однако, как и для IVP, остаются неясными способы реализации этого механизма.

Седьмое правило накладывает требование, что все случаи применения ПП регистрируются в специально предназначенном для этого CDI.

*Правило 7. Каждое применение ПП должно регистрироваться в специальном CDI, в который может производиться только добавление информации, достаточной для восстановления картины о процессе работы этого CDI.*

Это правило требует ведения специального регистрационного журнала, который хранится в определенном CDI.

Восьмое правило накладывает требование аутентификации субъектов, желающих инициировать ПП.

*Правило 8. Система должна распознавать субъекты, пытающиеся инициировать ПП.*

Это правило определяет механизмы предотвращения атак, при которых один субъект пытается выдать себя за другого.

Последнее правило накладывает административное требование о том, кому дозволено менять списки авторизации (например, KBM–тройки).

*Правило 9. Система должна разрешать производить изменения в списках авторизации только специальным субъектам (например, офицерам безопасности).*

Это правило гарантирует, что основная защита, определяемая KBM–тройкой, не будет обойдена злоумышленником, пытающимся изменить содержание такого списка.

Можно сказать, что указанные выше девять правил определяют, как может быть проверена целостность, как и кем могут изменяться CDI и как UDI могут быть превращены в CDI.

Следует заметить, что некоторые исследователи ставят под сомнение необходимость включения в модель регистрационных, административных и аутентификационных правил на том основании, что они не дают никакой

выгоды с логической точки зрения. Если в модель включена интерактивная регистрация (например, запись всех действий в защищенное расписание) для обеспечения корректного использования ПП, то это может привести к невыполнению одного из других правил. Это остается областью активных исследований и дебатов.

Основным преимуществом модели KBM является то, что она основана на проверенных временем бизнес-методах обращения с бумажными ресурсами. Поэтому модель KBM не следует рассматривать как академическое исследование, а скорее как комплекс существующих методов. Модель KBM также предоставляет исследователям методы работы с целостностью, отличные от традиционных уровне-ориентированных подходов, таких как модели БЛМ и Биба.

Основным недостатком модели KBM является то, что IVP и методы предотвращения CDI от искажения целостности нелегко реализовать в реальных компьютерных системах. Конечно, эти принципы легко реализовать в ограниченном наборе приложений. Например, в случае стекового приложения IVP можно реализовать путем анализа длины стека на основании всех операций push и pop для определения правильной длины стека. Кроме того, ограничения на ПП можно реализовать за счет использования абстрактного типа данных, для которых единственно возможными операциями являются push и pop.

Однако в менее тривиальных приложениях использование IVP и ПП затруднительно. Мы уже упоминали ограничения на просмотр кода как проблему при создании IVP для ПО. Подобные ограничения существуют для других потенциальных методов проверки ПО, таких как контрольная сумма, синтаксическая проверка, анализатор качества ПО и т.д. Заметьте, однако, что даже если эти методы и не гарантируют целостность полностью, они предоставляют дополнительный уровень обеспечения целостности.

Все недостатки данной модели вытекают из ее неформализованности. Ее можно применять при проектировании систем для спецификации

пользовательских приложений и использовать на соответствующем уровне иерархии рассмотрения защищенной вычислительной системы.

### **12.9. Механизмы защиты от угрозы отказа в обслуживании**

До настоящего времени большинство исследований компьютерной безопасности было связано с угрозами раскрытия и целостности. Одной из причин такого внимания этим угрозам является то, что различные международные организации по защите определяли раскрытие и целостность как основные угрозы. В результате, большинство средств, выделяемых на исследования, тратилось на работы именно в этих направлениях. Другая причина состоит в том, что поскольку отказ в обслуживании очень тесно связан с существующими понятиями доступности системы и разработки систем реального времени (многие годы изучавшимися исследователями и разработчиками), исследователи безопасности все же должны найти подходящую нишу в этой области анализа вычислительных систем.

Несмотря на эти преграды, в области отказа в обслуживании были сделаны некоторые предварительные шаги. В этом параграфе рассмотрим понятия, связанные с описанием и предотвращением угрозы отказа в обслуживании (ОВО). В частности, определим ряд терминов, среди которых важное место занимает понятие **максимального времени ожидания, впервые введенное Virgil и Gligor**. Требования ОВО будут рассмотрены с использованием положений временной логики. Далее представим мандатную модель ОВО **"никаких отказов вверх" (NDU)** и сравним ее с аналогичными уровневými моделями безопасности для угроз раскрытия и целостности. Параграф завершает краткое описание модели распределения ресурсов Миллена, которую можно использовать для описания моделей и стратегий ОВО.

#### **12.9.1. Основные понятия ОВО**

Для того чтобы понять угрозу отказа в обслуживании, необходимо вспомнить, что безопасные вычислительные системы служат



промежуточным звеном при запросе услуг пользователями посредством монитора пересылок. Такой промежуточный монитор пересылок позволяет рассмотреть запросы услуг в терминах простой модели, в которой пользователи являются зарегистрированными либо незарегистрированными, и обеспечиваются запрашиваемой услугой либо получают отказ. В случаях, когда зарегистрированным пользователям не предоставляется запрашиваемая услуга, говорят, что имеет место отказ в обслуживании.

Gligor первым отметил, что в понятия предоставления или отказа в обслуживании должно быть включено время. Добавление времени к простой модели запроса услуг основано на том, что каждая услуга должна быть связана с некоторым периодом времени, называем *максимальным временем ожидания* (MWT). Для некоторой услуги MWT определяется как длина промежутка времени после запроса услуги, в течение которого считается приемлемым предоставление этой услуги.

Можно трактовать приведенное выше определение по-другому. А именно, рассматривать MWT как период времени, в течение которого запрашиваемая услуга не устаревает. Иными словами, если *после запроса услуга обеспечивается в течение слишком долгого времени, то может случиться так, что ее нельзя будет больше использовать*. Заметим, что хотя приведенное выше определение и не выражено в терминах пользователей, запрашивающих услуги, может оказаться, что для данной услуги MWT будет различным для различных пользователей.

Рассмотрим пример, иллюстрирующий MWT. Предположим, что самолет имеет на борту контроллер полета, запрашивающий информацию о положении у узла измерения инерции (например у гироскопа). Очевидно, что во время полета информация о положении должна предоставляться на контроллер немедленно. Например, если контроллер запрашивает информацию о положении, в то время как самолет поворачивается в некотором направлении, то из-за любого отказа информация о положении устаревает по мере того, как меняется положение самолета. Таким образом,

услуге должно соответствовать маленькое MWT, а предоставление информации должно происходить прежде, чем пройдет MWT.

Дав определение MWT, мы можем теперь ввести точное определение угрозы ОВО, а именно, будем говорить, что имеет место **угроза ОВО всякий раз, когда услуга с соответствующим максимальным временем ожидания (MWT) запрашивается зарегистрированным пользователем в момент времени  $t$  и не предоставляется этому пользователю к моменту времени  $(t + \text{MWT})$** . Это означает, что ответы на запросы зарегистрированных пользователей не должны опаздывать.

При более тщательном рассмотрении угрозы ОВО и предложенного понятия MWT возникают некоторые вопросы. Например, угрозы ОВО можно полностью избежать, если определить MWT для всех услуг равным бесконечности. Однако этот подход не годится для тех случаев, когда величина MWT определяется некоторой значимой операционной характеристикой среды (например, в ситуации с измерением инерции самолета). Кроме того, значение MWT можно определить только для конкретного набора услуг, для которых оно необходимо, то есть можно определить некоторое подмножество активнов системы как особенно критическое; тогда значения MWT будут соответствовать услугам, связанным с этими критическими активами.

### **Выражение требований ОВО во временной логике**

Для того чтобы проиллюстрировать приведенные выше понятия ОВО, рассмотрим простое требование ОВО для типичной вычислительной системы и выразим его, используя расширение логики предикатов первого порядка, называемое **временной логикой**.

Требование ОВО будет выражено по отношению к системе, которая состоит из набора субъектов, активно инициирующих запросы услуг в различных состояниях системы. Значение MWT для каждого запроса услуги  $r$  может быть получено вычислением функции  $\text{mwt}(r)$ , областью определения которой являются все запросы услуг, а областью значений - множество

положительных целых чисел. Если максимальное время ожидания услуги равняется некоторому целому числу  $n$ , то удовлетворение запросов этой услуги должно происходить раньше, чем через  $n + 1$  единицу времени.

Требование заключается в том, что если  $\text{mwt}(r) = n$ , то удовлетворение запроса услуги  $r$  в момент времени  $t$  должно произойти к моменту времени  $t + n$ . Это требование легко формализуется в терминах временной логики использованием эвентуального оператора (обозначаемого  $a$ ). Этот оператор определяется следующим образом: если  $P$  - предикат в логике первого порядка, то  $aP$  - утверждение во временной логике. Если утверждение  $aP$  истинно для некоторого состояния  $s$ , это означает, что  $P$  должно быть истинно для состояния  $s$  или для некоторого состояния, имеющего место после  $s$ . Можно установить простые временные границы, определяя оператор  $a$  следующим образом: если  $a(t)P$  истинно для некоторого состояния, это означает, что  $P$  должно быть истинно в некотором последующем состоянии, которое достигается раньше, чем через  $t$  единиц времени.

Для заданных операторов можно определить требование в приведенном выше примере. Предположим, что всякий раз, когда субъект производит запрос услуги  $r$  в некотором состоянии, предикат  $\text{REQ}(r)$  является истинным. Можно предположить, что этот предикат не будет истинным в последующих состояниях, в которых ответ на запрос может все еще ожидаться. Более того, предположим, что всякий раз, когда запрос услуги  $r$  удовлетворяется в некотором состоянии, истинным является предикат  $\text{GRANT}(r)$ . Для упрощения примера предположим далее, что каждый запрос услуги единственен (это позволит облегчить измерение времени от запроса до завершения). Наш пример можно описать следующим образом:

$$\text{REQ}(r) \rightarrow \diamond (\text{mwt}(r)) \text{GRANT}(r).$$

### **12.9.2. Мандатная модель ОВО**

Теперь опишем в общих чертах мандатную модель ОВО, включающую в себя многие характеристики моделей БЛМ и Биба. Система услуг, которая будет использоваться для представления этой модели ОВО, выражается в знакомых терминах субъектов и объектов, которые использовались для описания моделей БЛМ и Биба.

Субъектам системы соответствуют приоритеты, которые могут быть одинаковы, ниже или выше по сравнению с приоритетом любого другого субъекта. Объектам соответствуют степени критичности, имеющие аналогичную иерархическую структуру. Субъект может требовать услугу у вычислительной системы, запрашивая доступ к объектам системы. Говорят, что субъект **получает отказ в обслуживании, если его запрос зарегистрирован, но не удовлетворен в течение соответствующего MWT.**

При описании модели необходимо рассмотреть условия, при которых один субъект может отказать в обслуживании другому субъекту. Такой отказ может быть вполне приемлем для одних случаев и совершенно не допустим для других. Например, администратор может иметь вполне подходящее оправдание, отказывая зарегистрированному пользователю в обслуживании, а нарушитель, как правило, не должен иметь такого оправдания. Правила, составляющие модель, нацелены на то, чтобы определить условия, при которых отказ в обслуживании был бы недопустим.

Рассмотрим правила, описывающие мандатную модель ОВО. Эти правила описывают взаимоотношения субъектов, аналогичные отношениям между субъектами и объектами в моделях БЛМ и Биба. Первое правило - **"никаких отказов вверх" (NDU)** - основано на том наблюдении, что никаким объектам с более низким приоритетом не позволено отказывать в обслуживании субъектам с более высокими приоритетами. Однако некоторым субъектам с более высоким приоритетом (например администраторам системы) должна предоставляться возможность отказывать

в обслуживании объектам с более низким приоритетом, если первые того желают.

Второе правило, являющееся просто обобщением первого, представляет собой альтернативу, которую можно использовать в тех приложениях, для которых **защищенным от угроз отказа в обслуживании должно быть лишь некоторое подмножество объектов**. Таким образом, это правило учитывает то, что проблема отказа в обслуживании может стоять только для объектов из некоторого конкретно определенного множества, то есть это правило позволяет обеспечить защиту от отказа в обслуживании только для особого множества объектов.

Это более общее правило требует, чтобы субъекты с более низкими приоритетами не препятствовали запросам услуг субъектов с более высокими приоритетами, производимыми через объекты из некоторого конкретно определенного множества. Это множество, которое мы обозначим через *C*, обычно содержит те объекты, которые являются наиболее критическими и для которых предоставляемые услуги никогда не должны устаревать.

**Правило NDU(C)** утверждает, что **ни один субъект не может отказать запросам, сделанным субъектом с более высоким приоритетом через объекты из множества C**. Второе правило особенно полезно для вычислительных систем, в которых необходимо обеспечивать защиту ОВО только для **выбранного множества критических услуг**. Например, система, выполняющая некоторую определенную роль, может содержать лишь небольшое множество услуг, напрямую связанных с этой ролью. В результате защиту ОВО с использованием правила NDU(C) можно обеспечить только для этих критических услуг. Это значительно сократит стоимость и трудность реализации стратегии ОВО.

Главным преимуществом двух представленных правил ОВО является то, что они дают средство для предотвращения отказа в обслуживании на основе **понятия (приоритета)**, предположительно уже существующего для

данной системы. Например, для большинства операционных систем существует понятие **приоритета процессов**. Данные правила также являются гибкими в том смысле, что их легко можно приспособить к данной системе. Ограничение на объекты в правиле NDU(C) можно выразить в терминах какого-либо определения критичности объекта.

Недостаток этих правил заключается в том, что они имеют смысл только для систем, в которых можно определить несколько приоритетов. Если это не так, тогда должны быть определены подходящие аналогичные правила внутри уровня с одним приоритетом. Например, формулировка этих правил для ПЭВМ с одним единственным пользователем не будет иметь большого смысла. Кроме того, как это было в случае модели Clark-Wilson, реализация методов, необходимых для гарантии того, что провозглашенные правила имеют силу, нетривиальна.

И, наконец, оказывается, что данные правила содержат многие характеристики (положительные и отрицательные) моделей BLP и Биба. Например, можно легко построить системы типа Системы Z, отвечающие требованиям правил NDU и NDU(C), но не согласующиеся с функциями защиты от отказа в обслуживании.

### ***12.9.3. Модель Миллена распределения ресурсов (MPP)***

Приведенные выше правила ОВО касаются условий, которых необходимо избегать, если мы хотим гарантировать, что действия нарушителя не создадут условий для отказа в обслуживании. Однако среди тех ситуаций, которые могут вызвать угрозу отказа в обслуживании, встречается много таких, которые затрагивают вопросы времени и пространства, не выявляемые простыми правилами типа приведенных выше.

Джонатан Миллен представил модель распределения ресурсов, позволяющую определить правила и стратегии отказа в обслуживании в терминах детального распределения ресурсов, включающего предоставление услуг пользователям вычислительных систем. Модель Миллена отличается

от традиционных уровневых правил в моделях БЛМ и Viba, поскольку в ее основе лежит идея о том, что для выполнения нужного задания субъектам необходимы определенные пространственные и временные требования к ресурсам. Отказ в обслуживании происходит в том случае, если распределение пространства и времени для некоторого процесса не отвечает соответствующим требованиям. Миллен показал, что применительно к его модели можно легко объяснить такие понятия, как конечное время ожидания (FWT) и максимальное время ожидания (MWT).

Модель Миллена включает в себя вполне соответствующий стратегии отказа в обслуживании набор правил, характеризующих семейство вычислительных систем, то есть эти правила построены таким образом, чтобы включить в это семейство понятия, которые во многом помогли бы при описании и анализе стратегий отказа в обслуживании. MPP и соответствующие правила представлены ниже.

Пусть  $P$  - множество активных процессов,  $R$  - множество пассивных ресурсов,  $c$  - некоторая фиксированная граница, используемая для обозначения общего максимального числа единиц всех типов ресурсов, доступных в исследуемой системе. Через вектор распределения  $A_p$  обозначим для каждого ресурса число единиц ресурса, выделенных для процесса  $p$  в некотором состоянии. Таким образом, вектор распределения можно рассматривать как своего рода отображение выделения ресурсов для процесса. Для формирования информации о том, является ли процесс текущим или застывшим, используется особый тип ресурсов - ресурс ЦПУ. В частности, всякий раз, когда  $A_p(\text{CPU}) = 1$ , будем говорить, что истинным является  $\text{running}(p)$ , а всякий раз, когда  $A_p(\text{CPU}) = 0$ , будем говорить, что истинным является  $\text{asleep}(p)$ .

Вектор пространственных требований  ${}^sQ_p$  означает число единиц каждого ресурса, требуемое процессом  $p$  для выполнения необходимого задания в некотором состоянии. Предполагается, что процессы могут определять множество ресурсов, необходимых им для завершения работы, до

того, как они ее начнут. Функция  $T(p)$  показывает, когда в последний раз изменились часы для процесса с целью отражения реального времени. Вектор временных требований  ${}^TQ_p$  обозначает объем времени, необходимого каждому ресурсу процесса  $p$  для выполнения работы. Как и в случае пространственных требований, предполагается, что процесс может определять временные требования для конкретного задания.

Ниже перечислены восемь правил, составляющих модель Миллена. Каждое правило ограничивает семейство систем, соответствующих этой модели. Прежде чем вводить эти правила, важно заметить, что "помеченные" переменные (например  $running(p)$ ) показывают значение переменной после одного перехода.

$$(R1) \sum A_p \leq c.$$

$$\{p \in P$$

Правило R1 утверждает, что сумма единиц выделенных ресурсов для всех процессов из  $P$  должна быть меньше системной границы  $c$ . Стратегии, нарушающие это правило, Миллен называет неправдоподобными, поскольку невозможно выделить ресурсов больше, чем имеется в наличии.

$$(R2) \text{ if } running(p) \text{ then } {}^sQ_p = 0.$$

Правило R2 утверждает, что текущие процессы должны иметь нулевые пространственные требования. Миллен аргументирует это тем, что если процесс не обладает всеми ресурсами, которые необходимы ему в некотором состоянии, то не имеет смысла продолжать этот процесс, пока требования не будут выполнены. "Зависание" происходит, когда процесс имеет ненулевые пространственные требования, которые никогда не будут выполнены (или будут выполнены поздно).

$$(R3) \text{ if } running(p) \text{ and } running(p)' \text{ then } A_p' = A_p.$$

Конструкция " $running(p)$  and  $running(p)'$ " в правиле R3 означает, что в некотором состоянии процесс  $p$  является текущим и остается текущим и в следующем состоянии. Это правило утверждает, что для текущих процессов распределение ресурсов не меняется. Это действительно мощное



предположение, поскольку из него следует, что во время выполнения текущие процессы не выгружаются в случае, если произойдет какое-либо перераспределение ресурсов (в отличие от перераспределения ресурсов ЦПУ).

(R4) if  $A_p(\text{CPU})' = A_p(\text{CPU})$  then  $T(p)' = T(p)$ .

Правило R4 утверждает, что часы процесса изменяются только с изменением распределения ЦПУ. Предполагается, что единицы времени - это положительные целые числа, всегда возрастающие с изменением времени (как будет определено ниже).

(R5) if  $A_p(\text{CPU})' \neq A_p(\text{CPU})$  then  $T(p)' > T(p)$ .

Правило R5 утверждает, что часы изменяются только для того, чтобы отразить увеличение во времени. Заметим, что у каждого процесса имеются свои собственные часы, а синхронизации различных часов друг с другом или с какими-либо часами, показывающими реальное время, не предпринимается.

(R6) if asleep(p) then  ${}^S Q_p' = {}^S Q_p + A_p - A_p'$ .

Правило R6 утверждает, что пространственные требования устанавливаются для застывших процессов. Другими словами, если процесс не является текущим, он должен определить те ресурсы, которые ему потребуются для выполнения задания. Как только все эти ресурсы получены, процесс возобновляется и становится текущим.

(R7) if asleep(p) then  ${}^T Q_p' = {}^T Q_p$ .

Правило R7 утверждает, что для застывших процессов временные требования не устанавливаются, то есть тогда как пространственные требования для застывших процессов устанавливаются, временные требования для таких процессов не устанавливаются.

(R8) if running(p) and asleep(p)' then  $A_p' = A_p - \text{CPU}$ .

Правило R8 утверждает, что переходы, в результате которых процесс останавливается, перераспределяют только ресурсы ЦПУ. Другие изменения в распределении должны происходить после возобновления процесса.

Миллен использует данную модель распределения ресурсов для описания некоторых стратегий. Например, применяя MPP, можно выразить стратегию **конечного времени ожидания (FWT)**. Для указания интервалов, которые могут возникнуть после многократных переходов, мы используем оператор временной логики *leads\_to* (то есть  $A \text{ leads\_to } B$  означает, что во всех последующих состояниях из  $A$  в конце концов следует  $B$ ). Для указания интервалов, которые могут возникнуть после многократных переходов, можно выразить максимальное время ожидания (MWT) аналогичным образом.

$$FWT: \neg p, s : \exists s' : s'(\text{running}(p)) \text{ and } s \text{ leads\_to } s'.$$

В выражении, приведенном выше,  $s(x)$  означает, что  $x$  истинно для состояния  $s$ , а  $s \text{ leads\_to } s'$  означает, что " $s, s' : s((T(p)=n))$  и  $s'((T(p) =m))$  и  $m>n$ ". FWT утверждает, что **пользователи в конце концов получают запрашиваемые ресурсы, то есть они в конечном итоге получают ЦПУ для выполнения работы**. Аналогичным образом можно выразить максимальное время ожидания (MWT).

$$MWT: \exists b : \forall p, s : \exists s' : s'(\text{running}(p)) \text{ and } s \text{ leads\_to}(b) s'.$$

В этом выражении  $s \text{ leads\_to}(b) s'$  означает, что  $\forall s, s' : s((T(p)=n))$  и  $s'((T(p) =m))$  и  $m-n \leq b$ . MWT отличается от FWT тем, что здесь **накладывается ограничение на время ожидания пользователями возможности продвижения в выполнении задания**.