Программно-аппаратная защита

Лекция на тему: «Модели безопасности защиты от несанкционированного доступа»

ФИО преподавателя: Оцоков Ш.А.

2021

Алгоритмическая разрешимость

Проблема распознавания выводимости

Алгоритмическая неразрешимость некоторого класса задач:

- Решение уравнения в радикалах и др

Уточнить понятие алгоритма, Тьюринг 1937 г

Машина Тьюринга

1. Машина располагает конечным числом знаков (символов)

$$s_1, s_2, \ldots, s_k,$$

образующих так называемый *внешний алфавит*, в котором кодируются сведения, подаваемые в машину, а также те, которые вырабатываются в ней. Для общности последующих рассмотрений удобно принять, что среди знаков внешнего алфавита имеется *пустой знак* (пусть это будет для определенности s_1), посылка которого (вписывание которого) в какую-либо ячейку ленты (памяти) гасит (стирает) тот знак, который в ней раньше хранился, и оставляет ее пустой. О пустой ячейке будем говорить, что она хранит пустой знак.

Машина Тьюринга

- В каждой ячейке хранится не более одного знака
- Каждое сведение, хранящееся на ленте, изображается конечным набором знаков внешнего алфавита,
- К началу работы машины на ленту подается начальное сведение
- В качестве начальной информации на ленту можно подать любое слово в этом алфавите

В зависимости от того, какая была подана начальная информация $\mathfrak{A}$, возможны два случая:

а) после конечного числа тактов машина останавливается, подавая сигнал об остановке; при этом на ленте оказывается изображенной некоторая информация $\mathfrak{B}$. В таком случае говорят, что машина применима к начальной информации $\mathfrak{A}$ и перерабатывает ее в результирующую информацию $\mathfrak{B}$;

Машина Тьюринга

б) остановка и сигнал об остановке никогда не наступают. В таком случае говорят, что машина не применима к начальной информации $\mathfrak{A}$.

В машине Тьюринга система элементарных операций и вместе с нею система одноадресных приказов еще больше упрощены: на каждом отдельном такте приказ предписывает лишь замену единственного знака s_i, хранящегося в обозреваемой ячейке, каким-либо другим знаком s_j. При $j = i$ это означает, что содержание обозреваемой ячейки не изменяется;

П — обозревать соседнюю справа ячейку,
Л — обозревать соседнюю слева ячейку,
Н — продолжать обозревать ту же ячейку, что и прежде.

Машина Тьюринга

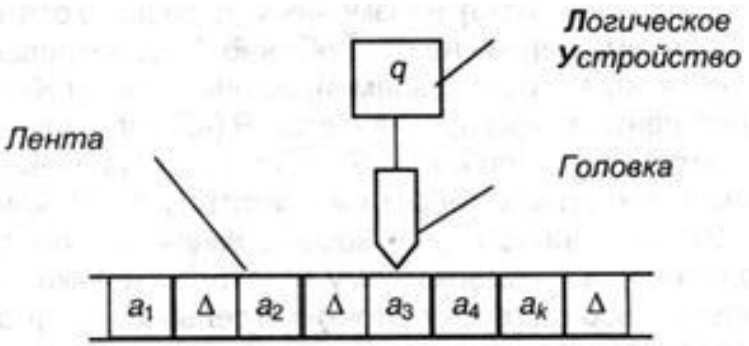


Рис. 7.1. Схема машины Тьюринга

В состав машины Тьюринга входит неограниченная в обе стороны *лента* (возможны машины Тьюринга, которые имеют несколько бесконечных лент), разделённая на ячейки[2][3], и *управляющее устройство* (также называется *головкой записи-чтения* (*ГЗЧ*)), способное находиться в одном из *множества состояний*. Число возможных состояний управляющего устройства конечно и точно задано.

Управляющее устройство может перемещаться влево и вправо по ленте, читать и записывать в ячейки символы некоторого конечного алфавита. Выделяется особый *пустой* символ, заполняющий все клетки ленты, кроме тех из них (конечного числа), на которых записаны входные данные.

Машина Тьюринга

Управляющее устройство работает согласно *правилам перехода*, которые представляют алгоритм, *реализуемый* данной машиной Тьюринга. Каждое правило перехода предписывает машине, в зависимости от текущего состояния и наблюдаемого в текущей клетке символа, записать в эту клетку новый символ, перейти в новое состояние и переместиться на одну клетку влево или вправо. Некоторые состояния машины Тьюринга могут быть помечены как *терминальные*, и переход в любое из них означает конец работы, остановку алгоритма.

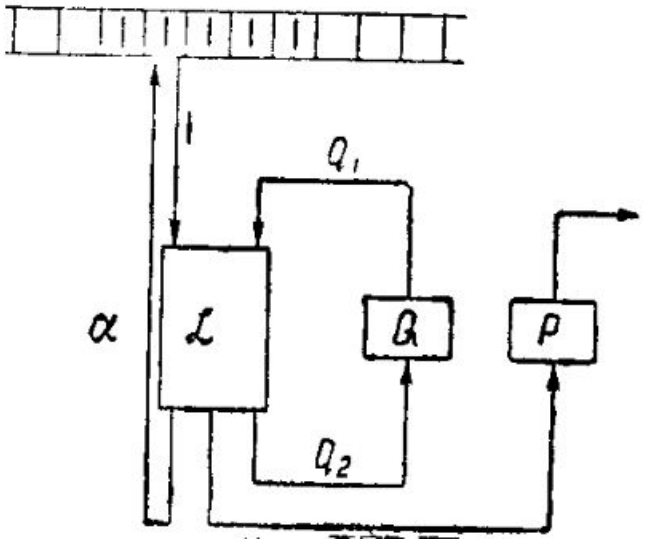
В машине Тьюринга обработка информации происходит *в логическом блоке*, который также может пребывать в одном из конечного числа состояний; пусть

$$q_1, q_2, \ldots, q_m$$

— специальные знаки, введенные для обозначения этих состояний. Блок имеет два входных канала; по одному из них на каждой стадии работы машины (в каждом такте) поступает знак из обозреваемой ячейки, по другому — знак q_l того состояния, которое предписывается блоку на данный такт; по выходному же каналу блок посылает в

Машина Тьюринга

Эту функцию, которую мы будем называть *логической функцией* машины, удобно изобразить в виде прямоугольной таблицы, столбцы которой занумерованы знаками состояний, а строки — знаками внешнего алфавита; в каждой же ячейке таблицы записана соответствующая выходная тройка знаков. Эту таблицу будем называть *функциональной схемой* машины;



Машина Тьюринга

Работа машины Тьюринга протекает следующим образом. Перед ее запуском на ленту заносится начальная информация (на нашем рис последовательность из пяти палочек) и в «поле зрения» машины устанавливается определенная начальная ячейка (на чертеже — ячейка, содержащая четвертую справа палочку), а в ячейки P и Q вводятся знаки начального состояния и начального сдвига (предположим q_1 и H). Дальнейший процесс протекает уже автоматически и однозначным образом определяется функциональной схемой машины. Посмотрим, например, что произойдет в том случае, когда задана функциональная схема

	q_1	q_2	q_3	q_4	q_5
Λ	$\Lambda П q_4$	$\Lambda Л q_3$	$\Lambda П q_1$	$\Lambda H q_5$	$\Lambda H q_5$
$\|$	$\alpha H q_2$	$\beta H q_1$	$\| П q_1$	$\| Л q_1$	$\| H q_5$
α	$\alpha Л q_1$	$\alpha П q_2$	$\| Л q_3$	$\Lambda П q_4$	$\alpha H q_5$
β	$\beta Л q_1$	$\beta П q_2$	$\Lambda Л q_3$	$\| П q_4$	$\beta H q_5$

Первый такт. Обозревается знак | (палочка) из начальной ячейки (сдвиг H) при состоянии q_1. Результат: выходная тройка знаков $\alpha H q_2$, т. е. знак | заменен знаком α, а в ячейки P, Q послана на хранение до следующего такта очередная команда Hq_2.

Машина Тьюринга

МИРЭА
Российский технологический университет

Второй такт. Обозревается знак α из той же ячейки (сдвиг H) при состоянии q_2. Выходная тройка: $\alpha П q_2$, т. е. знак α оставляется без изменения с переходом к команде $П q_2$.

Третий такт. Обозревается палочка из соседней справа ячейки (сдвиг $П$), при состоянии q_2. Результат: знак | заменен знаком β с переходом к команде Hq_1 и т. д.

Как видно из последнего столбца функциональной схемы рис. 6, остановка машины произойдет лишь при условии, что на некоторой стадии процесса возникнет состояние q_5. Действительно, каков бы ни был обозреваемый знак, он не будет заменен никаким другим, и машина будет продолжать обозревать его (сдвиг H) при том же состоянии q_5. Это и есть *стоп-состояние*, сигнализирующее о результативном завершении процесса в том случае, когда машина применима к информации, поданной в нее до запуска.

Машина Тьюринга

k-й конфигурация – изображение ленты машины с информацией, сложившейся на ней к началу k-го такта.

	q_1	q_2	q_3	q_4
λ	Πq_4	Πq_3	Πq_1	!
1	αq_2	Πq_3	Πq_1	!
α	Π	Π	1Π	$\lambda\Pi$
β	Π	Π	$\lambda\Pi$	1Π

Условимся еще в следующей упрощенной записи функциональных схем, которая делает схему более обозримой и удобной при выписывании конфигураций. Именно, мы откажемся от полной записи выходной тройки $s_j P q_l$, опуская знаки s_j и q_l, если они не отличаются от соответствующих входных знаков, а также знак H, указывающий на отсутствие сдвига.

Машина Тьюринга

МИРЭА
Российский технологический университет

Решается задача следующего типа:

Дана десятичная запись числа n (т. е. представление натурального числа n в десятичной системе счисления); *требуется указать десятичную запись числа* $n+1$.

Для этого берется внешний алфавит, состоящий из десяти цифр 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 и пустого знака $\wedge$. Машина может пребывать лишь в двух состояниях: q_0 (рабочее состояние) и! (остановка). Заданное число n, а также результирующее число $n+1$ будут записаны в десятичной системе, причем цифры будут помещаться по одной в ячейке (ячейки следуют подряд одна за другой без пропусков). Соответствующая функциональная схема дана на рис. 9 в виде той части указанной на нем таблицы, которая получится, если

Машина Тьюринга

Пусть к началу работы в поле зрения установлена цифра разряда единиц числа n, а машина настроена в состояние q_0;

	q_0	q_1
0	1 !	
1	2 !	
2	3 !	
3	4 !	
4	5 !	
5	6 !	
6	7 !	
7	8 !	
8	9 !	
9	0 Л	
Λ	1 !	
1	Л	Λ $Лq_0$

если эта цифра отлична от 9, то машина остановится уже после первого такта работы, в котором происходит замена этой цифры другой цифрой в соответствии со схемой. Если же последняя цифра 9, то машина заменяет ее нулем и производит сдвиг влево (к соседнему, более высокому разряду) и продолжает пребывать в рабочем состоянии q_0 (таким образом обеспечивается перенос единицы в более высокие разряды). Если число оканчивается k девятками, то машина закончит работу в точности после $k+1$ такта. На рис. 10 выписаны соответствующие конфигурации при $n = 389$.

Машина Тьюринга

Пусть на ленте задана десятичная запись числа n, а в нескольких ячейках, подряд расположенных правее этой записи, записаны палочки, по одной в ячейке. Посмотрим, как будет работать машина с этой функциональной схемой, если к началу работы в поле зрения установлена самая правая палочка, а сама машина настроена в состоянии q_1. В первом такте (входная пара $q_1|$) стирается эта палочка, а также происходит сдвиг налево и переход в состояние q_0 (выходная тройка $\wedge \Lambda q_0$). В последующих тактах машина продолжает сдвиг налево при состоянии q_0, сквозь все палочки до первой цифры разряда единиц. Начиная с этого момента все протекает уже как и в предыдущем алгоритме, т. е. происходит переработка записи числа n в запись числа $n+1$, и процесс завершается.

Короче, машина уменьшает на единицу число палочек и осуществляет в десятичной записи переход от числа n к числу $n+1$.

Термины

МИРЭА
Российский технологический университет

Большинство моделей безопасности оперируют терминами «сущность», «субъект», «объект».

Сущность — любая именованная составляющая защищаемой АС.

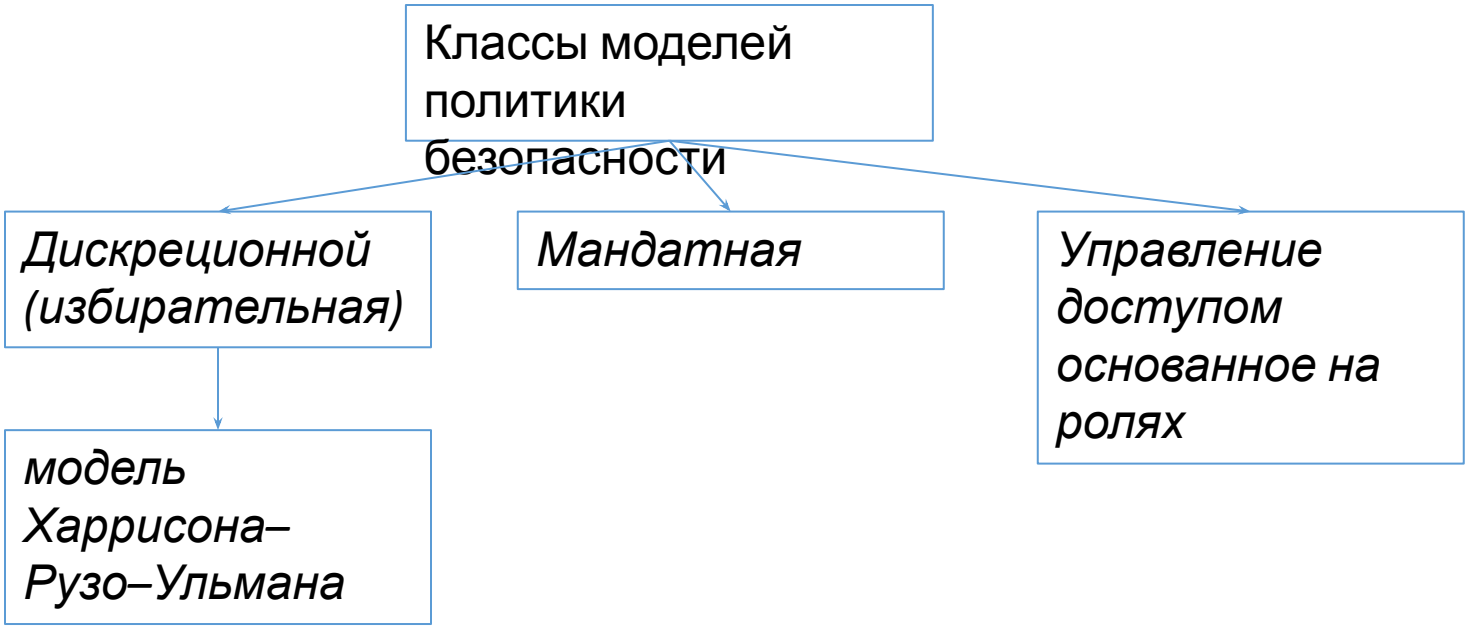
Субъект — активная сущность, которая может инициировать запросы ресурсов и использовать их для выполнения каких-либо вычислительных операций. В качестве субъекта может выступать выполняющаяся в системе программа или «пользователь» (не реальный человек, а сущность АС).

Объект — пассивная сущность, используемая для хранения или получения информации. В качестве объекта может рассматриваться, например, файл с данными.

Термины

При использовании матричной модели доступа должны быть определены множества субъектов **S**, объектов **O** и прав доступа **R**. В качестве субъектов системы рассматриваются в первую очередь выполняющиеся программы, поэтому предполагается, что $\mathbf{S} \subset \mathbf{O}$. Условия доступа субъекта $s \in \mathbf{S}$ к объекту $o \in \mathbf{O}$ определяются матрицей доступа. Пусть, например, множество прав доступа состоит из прав на чтение (r), запись (w), выполнение (e). Запрет будет соответствовать пустому множеству прав доступа ($\varnothing$). Тогда матрица доступа может быть такой, как представлено в табл. 1.1.

Классы моделей политики безопасности

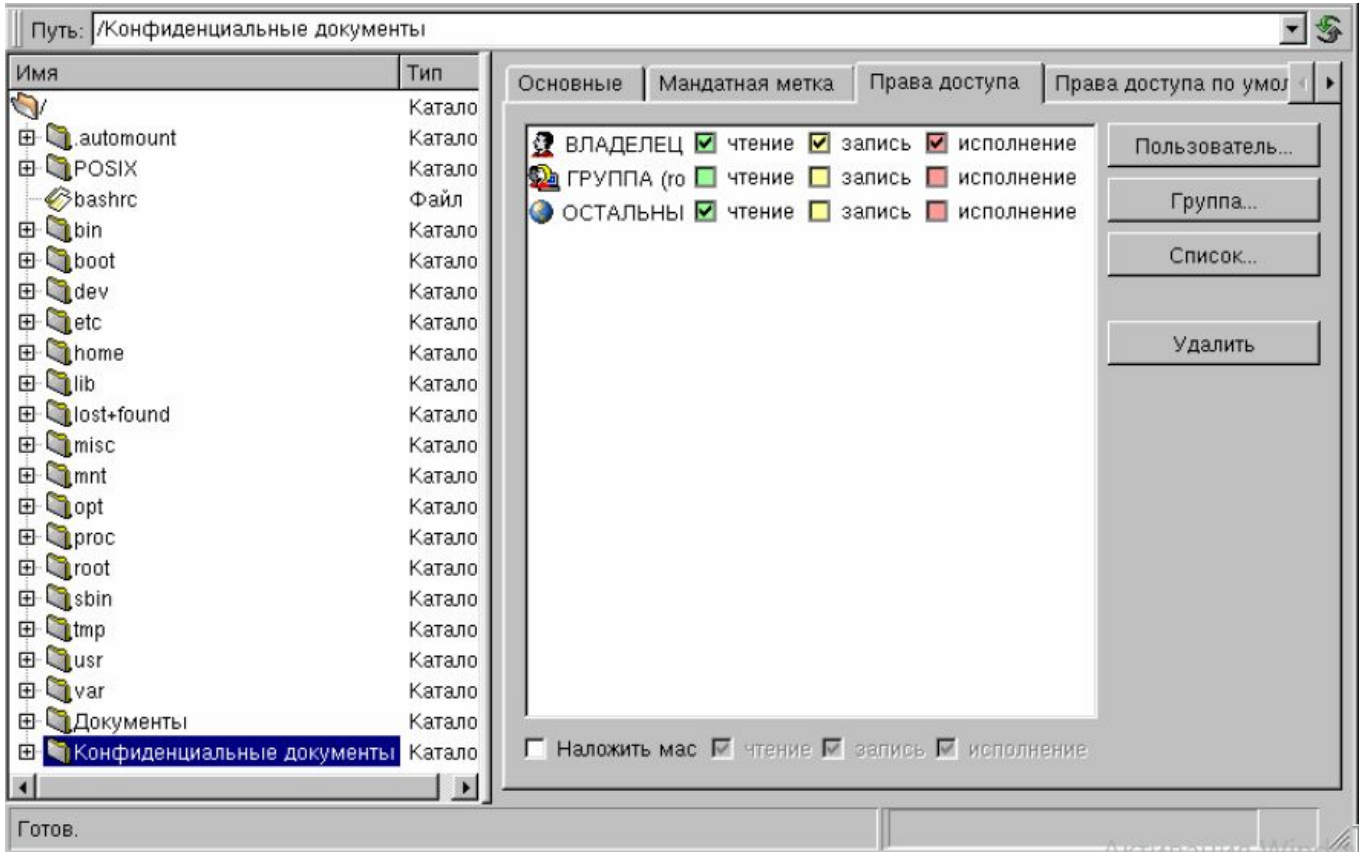


Модель Харрисона–Рузо–Ульмана

МИРЭА
Российский технологический университет

Пример матрицы доступа

	o_1	o_2	o_3	o_4
s_1	rwe	∅	rw	rw
s_2	e	rwe	r	∅



Модель Харрисона–Рузо–Ульмана

МИРЭА Российский технологический университет

Функционирование системы рассматривается с точки зрения изменений в матрице доступа. Модель определяет 6 примитивных операций: «создать»/«уничтожить» объект и субъект, «внести»/«удалить» право доступа субъекта к объекту. Их описание приведено в табл.

Элементами модели ХРУ являются:

O — множество объектов системы;

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

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

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

Модель Харрисона–Рузо–Ульмана

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

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

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

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

Таким образом, запись команды имеет следующий вид:

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

Модель Харрисона–Рузо–Ульмана

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

Пример 2.1. Команда создания субъектом s личного файла f.

command «создать файл» (s, f):
    «создать» объект f;
    «внести» право владения *own* в $M[s, f]$;
    «внести» право на чтение *read* в $M[s, f]$;
    «внести» право на запись *write* в $M[s, f]$;
end.

Пример 2.2. Команда передачи субъекту s' права *read* на файл f его владельцем субъектом s.

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

Модель Харрисона–Рузо–Ульмана

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

Начальное состояние $\mathbf{Q}_0$ называется *безопасным по отношению к некоторому праву* **r**, если невозможен переход системы в такое состояние **Q**, в котором может возникнуть утечка права **r**.

Система называется *монооперационной*, если каждая команда содержит только один примитивный оператор.

Для модели Харрисона–Рузо–Ульмана были доказаны следующие утверждения:

1. Существует алгоритм, который проверяет, является ли исходное состояние монооперационной системы безопасным для данного права **r**.

2. Задача проверки безопасности произвольных систем алгоритмически неразрешима.

Модель Харрисона–Рузо–Ульмана

- Общая модель Харрисона– Рузо–Ульмана может выражать большое разнообразие политик дис крецион ного доступа, но при этом не существует общего алгоритма проверки их безопасности.
- Для монооперационных систем алгоритм проверки безопасности существует, но данный класс является узким

Мандатная политика безопасности

Основу *мандатной* политики безопасности составляет мандатное управление доступом, которое подразумевает, что:

- все субъекты и объекты должны быть идентифицированы;
- задан линейно упорядоченный набор меток секретности;
- каждому объекту системы присвоена метка секретности, определяющая ценность содержащейся в нем информации — его *уровень секретности*;
- каждому субъекту системы присвоена метка секретности, определяющая уровень доверия к нему — его *уровень доступа*;
- решение о разрешении доступа субъекта к объекту принимается исходя из типа доступа и сравнения метки субъекта и объекта.

Модель Белла–ЛаПадула

S — множество субъектов (например, множество пользователей и программ);

O — множество объектов (например, множество файлов);

L — линейно упорядоченное множество уровней безопасности (например, «общий доступ», «для служебного пользования», «секретно», «совершенно секретно»);

F: $\mathbf{S} \cup \mathbf{O} \rightarrow \mathbf{L}$ — функция, определяющая уровень безопасности субъекта или объекта в данном состоянии;

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

Модель Белла–ЛаПадула

Система описывается начальным состоянием $v_0 \in \mathbf{V}$, множеством запросов **R** и функцией переходов T: $(\mathbf{V} \times \mathbf{R}) \to \mathbf{V}$, описывающей переход системы из состояния в состояние под действием запроса.

В модели Белла–ЛаПадула вводится определение двух свойств безопасности системы: безопасность по чтению и безопасность по записи.

Состояние (F, **M**) *безопасно по чтению* тогда и только тогда, когда для $\forall$ s $\in$ **S**, $\forall$ o $\in$ **O** выполняется требование:

$$\text{чтение} \in M[s, o] \Rightarrow F(s) \geq F(o),$$

т. е. субъект s может прочитать информацию из объекта o, только если уровень секретности *o* меньше или равен уровню доступа s. Данное свойство безопасности также называется правилом запрета чтения с верхнего уровня.

Модель Белла–ЛаПадула

Модель Белла–ЛаПадула

Состояние системы $v \in \mathbf{V}$ *безопасно* тогда и только тогда, когда оно безопасно и по чтению, и по записи.

Система $(v_0, \mathbf{R}, T)$ *безопасна* тогда и только тогда, когда ее начальное состояние v_0 безопасно и любое состояние, достижимое из v_0 после выполнения конечной последовательности запросов из **R**, также безопасно.

Большим достоинством модели Белла–ЛаПадула является то, что для нее доказана основная теорема безопасности.

Модель Белла–ЛаПадула

МИРЭА
Российский технологический университет

Основная теорема безопасности для модели Белла–ЛаПадула

Система (v_0, **R**, T) (т. е. система с начальным состоянием v_0, множеством запросов **R**, функцией переходов T) безопасна тогда и только тогда, когда состояние v_0 безопасно, и функция переходов T такова, что для $\forall v \in \mathbf{V}$, достижимого из состояния v_0 после выполнения конечной последовательности запросов из **R** (таких что $T(v, r) = v^*$, где $v = (F, \mathbf{M})$ — исходное состояние, $v^* = (F^*, \mathbf{M}^*)$ — состояние после перехода), для $\forall s \in \mathbf{S}$, $\forall o \in \mathbf{O}$ выполняются следующие условия:

- 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 $\notin M^*[s, o]$.

Модель Белла–ЛаПадула

Необходимость. Если система безопасна, то начальное состояние v_0 безопасно по определению. Пусть существует некоторое состояние v^*, достижимое из v_0 путем выполнения конечного числа запросов из **R** и полученное в результате перехода из безопасного состояния v: $T(v, r) = v^*$. Тогда, если при таком переходе нарушено хотя бы одно из первых двух ограничений, накладываемых теоремой на функцию T, то состояние v^* не будет безопасным по чтению. Если функция T нарушает одно из двух последних условий теоремы, то состояние v^* не будет безопасным по записи. Таким образом, при нарушении условий теоремы система становится небезопасной. Необходимость доказана.

Недостатки модели Белла–ЛаПадула

1. *Завышение уровня секретности*, связанное с одноуровневой природой объектов и правилом безопасности по записи. Если субъект с высоким уровнем доступа хочет записать что-то в объект с низким уровнем секретности, то сначала приходится повысить уровень секретности объекта, а потом осуществлять запись. Таким образом, даже один параграф, добавленный в большой документ субъектом с высоким уровнем доступа, повышает уровень секретности всего этого документа. Если по ходу работы изменения в документ вносят субъекты со все более высоким уровнем доступа, уровень секретности документа также постоянно растет.

Недостатки модели Белла–ЛаПадула

2. *Запись вслепую*. Эта проблема возникает, когда субъект производит операцию записи в объект с более высоким уровнем безопасности, чем его собственный. В этом случае после завершения операции записи субъект не сможет проверить правильность выполнения записи при помощи контрольного чтения, так как ему это запрещено в соответствии с правилом безопасности по чтению.

Доверенные субъекты. Модель Белла–ЛаПадула не учитывает, что в реальной системе, как правило, существуют субъекты, действующие в интересах администратора, а также системные процессы, например, драйверы. Жесткое соблюдение правил запрета чтения с верхнего уровня и запрета записи на нижний уровень в ряде случаев делает невозможной работу подобных процессов. Соответственно, их также приходится выделять.

Ролевая модель

Ролевая модель безопасности появилась как результат развития дискреционной модели. Однако она обладает новыми по отношению к исходной модели свойствами: управление доступом в ней осуществляется как на основе определения прав доступа для ролей, так и путем сопоставления ролей пользователям и установки правил, регламентирующих использование ролей во время сеансов.

В ролевой модели понятие «субъект» замещается понятиями «пользователь» и «роль» . Пользователь — человек, работающий с системой и выполняющий определенные служебные обязанности. Роль — это активно действующая в системе абстрактная сущность, с которой связан набор полномочий, необходимых для выполнения определенной деятельности. Подобное разделение хорошо отражает особенности деятельности различных организаций, что привело к распространению ролевых политик безопасности. При этом как один пользователь может быть авторизован администратором на выполнение одной или нескольких ролей, так и одна роль может быть сопоставлена одному или нескольким пользователям.

Ролевая модель

При использовании ролевой политики управление доступом осуществляется в две стадии:

- для каждой роли указывается набор полномочий (разрешений на доступ к различным объектам системы);
- каждому пользователю сопоставляется список доступных ему ролей.

При определении ролевой политики безопасности используются следующие множества:

U — множество пользователей;

R — множество ролей;

P — множество полномочий (разрешений) на доступ к объектам системы;

S — множество сеансов работы пользователя с системой.

Ролевая модель

Как уже отмечалось выше, ролям сопоставляются полномочия, а пользователям — роли. Это задается путем определения следующих множеств:

$PA \subseteq P \times R$ — определяет множество полномочий, установленных ролям (для наглядности условно может быть представлено в виде матрицы доступа);

$UA \subseteq U \times R$ — устанавливает соответствие между пользователями и доступными им ролями.

Рассмотрим процесс определения прав доступа для пользователя, открывшего сеанс работы с системой (в рамках одного сеанса работает только один пользователь). Правила управления доступом задаются с помощью следующих функций:

$user: S \to U$ — для каждого сеанса $s \in S$ эта функция определяет пользователя, который осуществляет этот сеанс работы с системой: $user(s) = u \mid u \in U$;

Ролевая модель

МИРЭА Российский технологический университет

roles — для каждого сеанса $s \in S$ данная функция определяет подмножество ролей, которые могут быть одновременно доступны пользователю в ходе этого сеанса: $roles(s) = \{r_i \mid (user(s), r_i) \in UA\}$;

$permissions : S \to P$ — для каждого сеанса $s \in S$ эта функция задает набор доступных в нем полномочий, который определяется путем объединения полномочий всех ролей, задействованных в этом сеансе: $permissions(s) = \bigcup_{r \in roles(s)} \{p_i \mid (p_i, r) \in PA\}$.

В качестве критерия безопасности ролевой модели используется следующее правило: система считается безопасной, если любой пользователь системы, работающий в сеансе $s \in S$, может осуществлять действия, требующие полномочия $p \in P$, только в том случае, если $p \in permissions(s)$.

Существует несколько разновидностей ролевых моделей управления доступом, различающихся видом функций *user*, *roles* и *permissions*, а также ограничениями, накладываемыми на множества PA и UA.

Ролевая модель

В частности, может определяться иерархическая организация ролей, при которой роли организуются в иерархии, и каждая роль наследует полномочия всех подчиненных ей ролей.

Могут быть определены взаимоисключающие роли (т. е. такие роли, которые не могут быть одновременно назначены одному пользователю). Также может вводиться ограничение на одновременное использование ролей в рамках одной сессии, количественные ограничения при назначении ролей и полномочий, может производиться группировка ролей и полномочий.

Литература

МИРЭА
Российский технологический университет

1. Нестеров С.А. Основы информационной безопасности, Санкт-Петербург, 2014 г.
2. Девянин П.Н. Модели безопасности компьютерных систем. – М.: Издательский центр «Академия», 2005.
3. Трахтенброт Б.А. Алгоритмы и машинное решение задач. – М.: 1957
4. https://habr.com/ru/company/solarsecurity/blog/509998/. Строим ролевую модель управления доступом. Часть первая, подготовительная

Спасибо за внимание!