МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждения высшего образования

«Юго-Западный государственный университет»

(ЮЗГУ)

Кафедра информационной безопасности

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

Методические рекомендации для практических занятий и самостоятельной работы студентов укрупненной группы специальностей и направлений подготовки 10.00.00 «Информационная безопасность»

Курск 2018

Лабораторная работа 1

Реализация политики безопасности

Цель работы

Целью лабораторной работы является изучение структуры, характеристик, сильных и слабых разлмичных политик информационной безопасности.

Формальные модели управления доступом

МОДЕЛЬ ХАРРИСОНА-РУЗЗО-УЛЬМАНА

Разработанная в 1971 г. **модель Харрисона-Руззо-Ульмана** [16] формализует упоминавшееся ранее понятие **матрицы доступа** – таблицы, описывающей права доступа субъектов к объектам.

Строки матрицы доступа соответствуют субъектам, существующим в системе, а столбцы – объектам. На пересечении строки и столбца указаны права доступа соответствующего субъекта к данному объекту: например субъект subj 3 обладает правами чтения и записи по отношению к объекту obj 3.

Введём следующие **обозначения**:

- S – множество возможных субъектов,
- O – множество возможных объектов (напомним, что $S \subset O$);
- $R=\{r_1, \ldots, r_n\}$ – конечное множество прав доступа
- $O \times S \times R$ - **пространство состояний системы**;
- M – матрица прав доступа, описывающая текущие права доступа субъектов к объектам;
- Q=(S, O, M) – текущее состояние системы;
- M[s,o] - ячейка матрицы, содержащая набор прав доступа субъекта $s \subset S$ к объекту $o \subset O$

Поведение системы во времени моделируется переходами между различными её состояниями. Переходы осуществляются путём внесения изменений в матрицу M с использованием команд следующего вида:

command $\alpha(x_1, \ldots, x_k)$
  if r_1 in $M[x_{s1}, x_{o1}]$ **and**
    r_2 in $M[x_{s2}, x_{o2}]$ **and**
    ...
    r_m in $M[x_{sm}, x_{om}]$
  then
    op_1,
    op_2,
    ...
    op_n,
end

Здесь α - имя команды; x_t - параметры команды, представляющие собой идентификаторы субъектов и объектов, op_i,— элементарные операции.

Элементарные операции $op_1...op_n$ будут выполнены в том случае, если выполняются все без исключения условия из блока **if … then**.

При описании элементарных операций мы будем полагать, что в результате выполнения операции система переходит из состояния Q=(S, O, M) в состояние Q'=(S',0',M').

<Модель предусматривает наличие шести **элементарных операций**:

1. **enter *r* into M[s,o]** (s⊂S,o⊂O) - добавление субъекту s права r по отношению к объекту o. В результате выполнения команды происходят следующие изменения в состоянии системы:
   - $S'=S$,
   - $O'=O$,
   - $M'[x_s, x_o]=M[x_s, x_o]$, *если* $(x_s, x_o) \neq (s,o)$,
   - $M'[s, o]=M[s, o] \cup \{r\}$.

   Заметим, что содержимое ячейки таблицы рассматривается как множество. Это, в частности, означает, что если добавляемый элемент уже присутствовал в ячейке, то её содержимое не изменяется.
2. **delete *r* from M[s,o]** (s ⊂S,o⊂O) - удаление у субъекта s права r по отношению к объекту o. Изменения в состоянии системы:
   - $S'=S$,
   - $O'=O$,
   - $M'[x_s, x_o]=M[x_s, x_o]$, *если* $(x_s, x_o) \neq (s,o)$,
   - $M'[s,o]=M[s,o] \setminus \{r\}$.

   Если удаляемое право отсутствовало в ячейке, то состояние системы в результате выполнения данной команды никак не изменяется.
3. **create subject S** (s ⊂ S) - создание нового субъекта s. Изменения в состоянии системы:
   - $O'=O \cup \{s\}$,
   - $S'=S \cup \{s\}$,
   - $M'[x_s, x_o]=M[x_s, x_o]$ *для* $\forall (x_s, x_o) \in S \times O$,
   - $M'[s, x_o]=\varnothing$ *для* $\forall x_o \in O'$
   - $M'[s, x_s]=\varnothing$ *для* $\forall x_s \in S'$

   Как видим, при создании субъекта в матрицу M добавляются строка и столбец.
4. **destroy subject s** (s∈S) – удаление существующего субъекта s. Изменения в состоянии системы:
   - $S'=S \setminus \{s\}$,
   - $O'=O \setminus \{s\}$,
   - $M'[x_s, x_o]=M[x_s, x_o]$ *для* $\forall (x_s, x_o) \in S' \times O'$.

5. **create object o** ($o \notin O$) – создание нового объекта o. Изменения в состоянии системы:
   - $O' = O \cup \{o\}$,
   - $S' = S$,
   - $M'[x_s, x_o] = M[x_s, x_o]$ для $\forall (x_s, x_o) \in S \times O$,
   - $M'[x_s, o] = \emptyset$ для $\forall x_s \in S'$

   При добавлении объекта в матрице доступа создаётся новый столбец.
6. destroy object o ($o \in O \setminus S$) – удаление существующего объекта o. Изменения в состоянии системы:
   - $O' = O \setminus \{o\}$,
   - $S' = S$,
   - $M'[x_s, x_o] = M[x_s, x_o]$ для $\forall (x_s, x_o) \in S' \times O'$.

Приведём несколько **примеров команд:**

1. **Создание файла**
   Пользователь p создаёт файл f и получает на него права владения, чтения и записи.
   command create_file (p, f)
   create object f,
   enter own into M[p, f],
   enter r into M[p, f],
   enter w into M[p, f],
   end
2. **Создание процесса.**
   Процесс p создаёт процесс q и получает на него право чтения, записи и владения, передавая процессу q права записи и чтения по отношению к самому себе.
   command exec_process(p, q)
   create subject q,
   enter own into M[p, q],
   enter r into M[p, q],
   enter w into M[p, q],
   enter r into M[q, p], enter w into M[q, p],
   end
3. **Передача права чтения по отношению к файлу**
   Право чтения на файл f передаётся владельцем p субъекту q.
   command grant_read(p, q, f)
   if own in M[p, f]
   then enter r into M[q, f],
   end

Формальное описание системы в модели Харрисона-Руззо-Ульмана выглядит следующим образом. **Система** $\sum = (Q,R,C)$ состоит из следующих **элементов**:

1. Конечный набор прав доступа $R=\{r_1 ...,r_n\}$.

2. Конечный набор исходных субъектов $S_0=\{s_1 ..., s_l\}$.
3. Конечный набор исходных объектов $O_0=\{o_1, .., o_m\}$.
4. Исходная матрица доступа M_0.
5. Конечный набор команд $C = \{\alpha_i(x_1,...,x_k)\}$.

Поведение системы во времени рассматривается как последовательность состояний {Q,}, каждое последующее состояние является результатом применения некоторой команды к предыдущему: $Q_{n+i}=C_n(Q_n)$.

Для заданной системы начальное состояние $Q_0=\{S_0, O_0, M_0\}$ называется безопасным **относительно права r**, если не существует применимой к Q_0последовательности команд, в результате выполнения которой право r будет занесено в ячейку матрицы M, в которой оно отсутствовало в состоянии Q_o.

Другими словами это означает, что субъект никогда не получит право доступа г к объекту, если он не имел его изначально.

Если же право г оказалось в ячейке матрицы M, в которой оно изначально отсутствовало, то говорят, что произошла **утечка права r**.

Рассмотрим **пример** [17]. Пусть система допускает использование двух прав доступа: R={r, w}, где г - чтение, а w - запись, и пусть система описывается следующими командами:

1. **Создание субъекта**

command create (s, o)
create subject o,
enter r into M[s, o],
enter w into M[s, o],
end

Команда разрешает создание нового субъекта с одновременным получением по отношению к нему прав доступа на чтение и на запись.

2. **Получение прав доступа**

command takex(s, o, p)
if r in M[s, o] and
x in M[o, p]

Как видим, в результате выполнения шагов 1-5 субъект s обходным путём получает право доступа а по отношению к субъекту о - т.е. происходит утечка права а - а это значит, что исходное состояние не является безопасным.

С практической точки зрения значительный интерес представлял бы универсальный метод определения того, является ли заданная система с некоторым начальным состоянием безопасной относительно того или иного права доступа. Покажем, как эта задача может быть решена для одного из частных случаев

истема $\sum = (Q,R,C)$ называется **монооперационной**, если каждая команда at e C выполняет один примитивный оператор.

Теорема. Существует алгоритм, который проверяет, является ли исходное состояние монооперационной системы безопасным для данного права а.

Покажем, что число последовательностей команд системы, которое необходимо проверить, является ограниченным. В этом случае проверка безопасности исходного состояния системы сведётся к полному перебору всех последовательностей и проверке конечного состояния каждой из них на отсутствие утечки права a.

Заметим, что команды **delete** и **destroy** можно не рассматривать, поскольку нас интересует наличие права a, а не его отсутствие. Аналогично, нет необходимости рассматривать более одного оператора **create**: система является монооперационной, и одна команда не может одновременно создать объект или субъект и модифицировать его права доступа, поскольку путём замены параметров можно ограничиться работой с последовательностями команд, которые оперируют над существующими субъектами и объектами. Единственная команда create будет необходима на случай, если в начальном состоянии в системе не было ни одного субъекта.

Итак, пусть c_1, c_2, ..., c_n — последовательность команд, в результате выполнения которой происходит утечка права a. Упростим эту последовательность команд следующим образом:

1. Удалим все команды **delete** и **destroy**.
2. Добавим в начало последовательности C_1, C_2 ... C_n, —, c,, команду S_{init} вида **create subject**.
3. Проходя последовательность команд справа налево, последовательно удалим все команды вида **create subject** и заменим все ссылки на создаваемые с помощью этих команд субъекты ссылкой на S_{init}.
4. Аналогично удалим все команды вида **create object**, заменяя ссылки на создаваемые с помощью этих команд объекты ссылками на S_{init}.
5. Удалим все команды вида **enter**, вносящие право a в ячейку, которая уже содержит это право.

Согласно приведённым выше замечаниям, получившаяся в результате данных преобразований последовательность команд также приводит к утечке права a. Проанализируем состав возможных команд в получившейся последовательности.

Команды вида create object, destroy subject, destroy object и delete в последовательности отсутствуют. Команда create subject присутствует в единственном числе. Максимальное число команд вида enter равно $|R|(|S_0|+1)(|O_0|+1)$. Тем самым, общее число возможных команд равно $R(S_0 +1)(O_0 +1) + 1$ – а значит, количество последовательностей команд ограничено

К сожалению, расширить полученный результат на произвольные системы невозможно.

Теорема. Для систем общего вида задача определения того, является ли исходное состояние системы безопасным для данного права a, является вычислительно неразрешимой.

Для доказательства этого утверждения достаточно свести задачу проверки безопасности системы к заведомо неразрешимой задаче остановки машины Тьюринга.

Классическая модель Харриона-Руззо-Ульмана до сих пор широко используется при проведении формальной верификации корректности построения систем разграничения доступа в высоко защищённых автоматизированных системах. Развитие моделей дискреционного управления доступом [18, 19] заключается преимущественно в построении всевозможных модификаций модели Харрисона-Руззо-Ульмана, а также в поиске минимально возможных ограничений, которые можно наложить на описание системы, чтобы вопрос её безопасности был вычислительно разрешимым.

МОДЕЛЬ БЕЛЛА-ЛАПАДУЛЫ

Данная модель была предложена в 1975 году [20] для формализации механизмов мандатного управления доступом. Мандатный принцип разграничения доступа, в свою очередь, ставил своей целью перенести на автоматизированные системы практику секретного документооборота, принятую в правительственных и военных структурах, когда все документы и допущенные к ним лица ассоциируются с иерархическими уровнями секретности.

В **модели Белла-ЛаПадулы** по грифам секретности распределяются субъекты и объекты, действующие в системе, и при этом выполняются следующие **правила**:

1. **Простое правило безопасности (Simple Security, SS)**. Субъект с уровнем секретности x_5 может читать информацию из объекта с уровнем секретности x_o тогда и только тогда, когда x_5преобладает над x_o.
2. ***-свойство (*-property)**.
3. Субъект с уровнем секретности x_5 может писать информацию в объект с уровнем секретности x_o в том и только в том случае, когда x_oпреобладает над x_5.

Для первого правила существует мнемоническое обозначение No Read Up, а для второго – No Write Down.

Перейдём к формальному описанию системы. Введём следующие **обозначения**:

- S - множество субъектов;
- O - множество объектов, S с O;
- R={r, w} - множество прав доступа, г - доступ на чтение, w - доступ на запись;
- L={U, SU, S, TS} - множество уровней секретности, U- Unclassified, SU - Sensitive but unclassified, S - Secret, TS - Top secret;
- $\Lambda = (L, \leq, \bullet, \Theta)$ - решётка уровней секретности;
- V - множество состояний системы, представляемое в виде набора упорядоченных пар (F, M), где:
  - F :S U O -> L - функция уровней секретности, ставящая в соответствие каждому объекту и субъекту в системе определённый уровень секретности;
  - M- матрица текущих прав доступа.

становимся более подробно на решётке уровней секретности. Напомним, что решёткой Λ называется алгебраическая система вида $(L, \leq, \bullet, \Theta)$, где:

- $\leq$ - оператор, определяющий частичное нестрогое отношение порядка для уровней секретности;
- • - оператор наименьшей верхней границы;
- Θ - оператор набольшей нижней границы.

Отношение $\leq$ обладает следующими свойствами:

1. **Рефлексивность**:
   $\forall a \in L : a \leq a$. С точки зрения уровней безопасности это означает, что разрешена передача информации между субъектами и объектами одного уровня безопасности.
2. **Антисимметричность**:
   $\forall a_1,a_2 \in L$: $((a_1 \leq a_2)$ & $(a_2 \leq a_1))$—>$a_2 = a_1$. Антисимметричность в нашем случае означает, что если информация может передаваться как от субъектов и объектов уровня А к субъектам и объектам уровня В, так и от субъектов и объектов уровня В к субъектам и объектам уровня А, то эти уровни эквивалентны.
3. **Транзитивность**:
   $\forall a_1,a_2,a_3 \in L$: $((a_1 \leq a_2)$ & $(a_2 \leq a_3))$ -> $a_1 \leq a_3$. Транзитивность означает, что если информации может передаваться от субъектов и объектов уровня А к субъектам и объектам уровня В, и от субъектов и объектов уровня В к субъектам и объектам уровня С, то она может передаваться от субъектов и объектов уровня А к субъектам и объектам уровня С.

Операторы **наименьшей верхней границы** • и **наибольшей нижней границы** Θ определяются следующим образом:

- $a = a_1 \bullet a_2 \Leftrightarrow (a_1, a_2 \leq a) \& (\forall a' \in L : (a' \leq a) \rightarrow (a' \leq a_1 \vee a' \leq a_2))$;
- $a = a_1 \otimes a_2 \Leftrightarrow (a \leq a_1, a_2) \& (\forall a' \in L : (a' \leq a_1 \& a' \leq a_2) \rightarrow (a' \leq a))$.

Нетрудно показать, что для каждой пары $\forall a_1,a_2 \in L$ существует единственныйэлемент наименьшей верхней границы и единственный элемент наибольшей нижней границы.

Заметим, что в качестве уровней безопасности совершенно не обязательно выбирать целые числа, в ряде случаев удобнее использовать более сложные структуры. За счёт этого, например, в пределах каждого уровня секретности можно реализовать категории секретности (см. рис. 2.3.2.2). В этом случае наличие допуска к той или иной категории информации может служить дополнительным механизмом безопасности, ограничивающим доступ к защищаемым субъектам или объектам.

Состояние (F, M) называется безопасным, если оно безопасно по чтению и по записи.

Наконец, система $\sum = (\nu_0,R,T)$ называется безопасной, если её начальное состояние ν_0 безопасно, и все состояния, достижимые из ν0 путём применения конечной последовательности запросов из R, безопасны.

Теорема (Основная теорема безопасности Белла-ЛаПадулы). Система $\sum = (\nu_0,R,T)$ безопасна тогда и только тогда, когда выполнены следующие условия:

1. Начальное состояние ν_0 безопасно.

2. Для любого состояния ν, достижимого из ν_0 путём применения конечной последовательности запросов из R, таких, что T(ν,r) = ν*, ν=(F, M) и ν*=(F*, M*), для ∀ s ∈ S, ∀ o ∈ O выполнены условия:

1. Если r ∈ M*[s,o] и r ∉ M[s,o], то F*(o) ≤ F*(s).
2. Если r ∈ M*[s,o] и F*(s)<F*(o), то r ∈ M*[s,o]
3. Если w ∈ M*[s,o] и w ∉ M[s,o], то F*(o) ≤ F*(s).
4. Если w ∈ M*[s,o] и F*(s)<F*(o), то w ∈ M*[s,o]

В настоящее время модель Белла-ЛаПадулы и другие модели мандатного управления доступом широко используются при построении и верификации автоматизированных систем, преимущественно предназначенных для работы с информацией, составляющей государственную тайну.

Ролевая модель контроля доступа (RBAC)

Ролевой метод управления доступом контролирует доступ пользователей к информации на основе типов их активностей в системе(ролей). Под ролью понимается совокупность действий и обязанностей, связанных с определенным видом деятельности. Примеры ролей: администратор базы данных, менеджер, начальник отдела.

В ролевой модели с каждым объектом сопоставлен набор разрешенных операций доступа для каждой роли, а не для каждого пользователя. Каждому пользователю сопоставлены роли, которые он может выполнять. В некоторых системах пользователю разрешается выполнять несколько ролей одновременно, в других есть ограничение на одну или несколько не противоречащих друг другу ролей в каждый момент времени.

Для формального определения модели RBAC используются следующие соглашения:

S = субъект – человек или автоматизированный агент.

R = роль – рабочая функция или название, определяется на уровне авторизации.

P = разрешения – утверждения режима доступа к ресурсу.

SE = сессия – Соответствие между S, R и/или P.

SA = назначение субъекта (Subject Assignment). SA ? S ? R. При этом субъекты назначаются связям ролей и субъектов в отношении «многие ко многим» (один субъект может иметь несколько ролей, а одну роль могут иметь несколько субъектов).

PA = назначение разрешения (Permission Assignment). PA ? P ? R. При этом разрешения назначаются связям ролей в отношении «многие ко многим».

RH = частично упорядоченная иерархия ролей (Role Hierarchy). PH ? R ? R.

На возможность наследования разрешений от противоположных ролей накладывается ограничительная норма, которая позволяет достичь надлежащего разделения режимов. Например, одному и тому же лицу может быть не позволено создать учетную запись для кого-то, а затем авторизоваться под этой учетной записью.

Основные достоинства ролевой модели:

1. Простота администрирования. В отличие от модели DAC нет необходимости прописывать разрешения для каждой пары «объект-пользователь». Вместо этого прописываются разрешения для пар «объект-роль» и определяются роли каждого пользователя. При изменении области ответственности пользователя, у него просто изменяются роли. Иерархия ролей также упрощает процесс администрирования. Иерархия ролей – это когда роль наряду со своими собственными привилегиями может наследовать привилегии других ролей.

2. Принцип наименьшей привилегии. Ролевая модель позволяет пользователю регистрироваться в системе ролью, минимально необходимой для выполнения требуемых задач. Запрещение полномочий, не требуемых для выполнения текущей задачи, не позволяет обойти политику безопасности системы.

3. Разделение обязанностей.

Задание к работе

1. Написать программу, реализующую модель информационной безопасности (по вариантам).

Вариант

Вариант	Модель информационной безопасности
1	Мандатная политика безопасности
2	Дискреционная политика безопасности
3	Модель матрицы доступов Харрисон-Руззо-Ульмана
4	Модель распространения прав доступа *take-grant*
5	Модель системы безопасности белла-лападула
6	Модель *low-water-mark*
7	Модели ролевого разграничения

3. Определить минимальную длину пароля, алфавит которого состоит из A символов, время перебора которого было не меньше T лет. Скорость перебора V паролей в секунду.

Вариант	A	T	V
1	33	100	100
2	26	120	13
3	52	60	30
4	66	70	20
5	59	50	200
6	118	90	50
7	128	100	500
8	150	30	200
9	250	80	600

4. Освоить программу «ViPNet Генератор паролей».
Оценить стойкость паролей, сгенерированных данной программой.

5. Задание на самостоятельную работу

1. Подготовить доклады на тему:
1) парольная система защиты ОС Windows;
2) парольная система защиты ОС семейства Unix;
3) парольные системы защиты различных служб Интернета (Web-сервера, электронная почта, FTP и т.д.)
4) парольные системы защиты архиваторов.
В докладах предлагается отразить следующие вопросы:
1) Организация (структура) парольной системы.
2) Место, способ хранения паролей и штатные средства защиты базы учетных записей.
3) Средства, предоставляемые администраторам для управления парольной системой.
4) Известные уязвимости парольной системы и методы преодоления парольной системы злоумышленником.
2. Исследовать парольные системы защиты архиваторов и офисных программ (Word, Excel, PowerPoint и т.д.). Определить стойкость парольных систем защиты архиваторов и офисных программ с помощью приложений для взлома пароля (Advanced Office Password Recovery, Archive Password Recovery и подобных).
3. (Дополнительное) Написать программу, которая должна эмулировать работу парольной системы защиты.
Программа должна реализовывать 5 из 10 требований, предъявляемых к парольным системам защиты.

Контрольные вопросы

1. Перечислите основные достоинства модели Белла-ЛаПадулы.
2. Какая модель безопасности используется в современных СУБД
3. Докажите теорему Белла-ЛаПадулы;
4. В чём отличие механизмов использования матрицы разграничения доступа в дискреционной и мандатной моделях
5. Опишите основные понятия модели Харрисона-Руззо-Ульмана

Лабораторная работа 2

Целостность данных. Модель Кларка-Вилсона

Цель работы

Целью работы является закрепление полученного теоретического материала по моделям целостности и применение на практике положений модели целостности Кларка-Вилсона к вычислительным системам.

Основные положения модели целостности Кларка-Вилсона

Модель Кларка-Вилсона появилась в результате проведенного авторами анализа реально применяемых методов обеспечения целостности документооборота в коммерческих компаниях. В отличие от моделей Биба и Белла-ЛаПадулы, она изначально ориентирована на нужды коммерческих заказчиков, и, по мнению авторов, более адекватна их требованиям, чем предложенная ранее коммерческая интерпретация модели целостности на основе решеток. Основные понятия рассматриваемой модели — это корректность транзакций и разграничение функциональных обязанностей.

Модель задает правила функционирования компьютерной системы и определяет две категории объектов данных и два класса операций над ними. Все содержащиеся в системе данные подразделяются на контролируемые и неконтролируемые элементы данных (constrained data items - CDI и unconstrained data items – UDI соответственно). Целостность первых обеспечивается моделью Кларка-Вилсона. Последние содержат информацию, целостность которой в рамках данной модели не контролируется (этим и объясняется выбор терминологии). Далее, модель вводит два класса операций над элементами данных: процедуры контроля целостности (integrity verification procedures - IVP) и процедуры преобразования (transformation procedures - TP).

Первые из них обеспечивают проверку целостности контролируемых элементов данных (CDI), вторые изменяют состав множества всех CDI (например, преобразуя элементы UDI в CDI). Так же модель содержит девять правил, определяющих взаимоотношения элементов данных и процедур в процессе функционирования системы.

Правило C1. Множество всех процедур контроля целостности (IVP) должно содержать процедуры контроля целостности любого элемента данных из множества всех CDI.

Правило C2. Все процедуры преобразования (TP) должны быть реализованы корректно в том смысле, что не должны нарушать целостность обрабатываемых ими CDI. Кроме того, с каждой процедурой преобразования должен быть связан список элементов CDI, которые допустимо обрабатывать данной процедурой. Такая связь устанавливается администратором безопасности.

Правило E1. Система должна контролировать допустимость применения TP к элементам CDI в соответствии со списками, указанными в правиле C2.

Правило E2. Система должна поддерживать список разрешенных конкретным пользователям процедур преобразования с указанием допустимого для каждой TP и данного пользователя набора обрабатываемых элементов CDI.

Правило C3. Список, определенный правилом C2, должен отвечать требованию разграничения функциональных обязанностей.

Правило E3. Система должна аутентифицировать всех пользователей, пытающихся выполнить какую-либо процедуру преобразования.

Правило C4. Каждая TP должна записывать в журнал регистрации информацию, достаточную для восстановления полной картины каждого применения этой TP. Журнал регистрации — это специальный элемент CDI, предназначенный только для добавления в него информации.

Правило C5. Любая TP, которая обрабатывает элемент UDI, должна выполнять только корректные преобразования этого элемента, в результате которых UDI превращается в CDI.

Правило E4. Только специально уполномоченное лицо может изменять списки, определенные в правилах C2 и E2. Это лицо не имеет права выполнять какие-либо действия, если оно уполномочено изменять регламентирующие эти действия списки.

Публикация описания модели Кларка-Вилсона вызвала широкий отклик среди исследователей, занимающихся проблемой контроля целостности. В ряде научных статей рассматриваются практические аспекты применения модели, предложены некоторые ее расширения и способы интеграции с другими моделями безопасности.

Роль каждого из девяти правил модели Кларка-Вилсона в обеспечении целостности информации можно пояснить, показав, каким из теоретических принципов политики контроля целостности отвечает данное правило:

1. корректность транзакций;
2. аутентификация пользователей;
3. минимизация привилегий;
4. разграничение функциональных обязанностей;
5. аудит произошедших событий;
6. объективный контроль.

Соответствие правил модели Кларка-Вилсона перечисленным принципам показано в таблице. Как видно из таблицы, принципы 1 (корректность транзакций) и 4 (разграничение функциональных обязанностей) реализуются большинством правил, что соответствует основной идее модели.

Правило модели Кларка-Вилсона	Принципы политики контроля целостности, реализуемые правилом
C1	1,6
C2	1

E1	3,4
E2	1,2,3,4
C3	4
E3	2
C4	5
C5	1
E4	4

Пример применения модели

В качестве примера рассмотрим систему форумовFORUM.TOMSK.RU

- CDI – контролируемые элементы данных: логин и пароль.
- UDI – неконтролируемые элементы данных: вводимые данные через интерфейс пользователя.
- IVP – процедуры контроля целостности проверяют соответствие введенных пользователем логина и пароля с зарегистрированным логином и паролем (которые хранятся в базе данных системы). Процедуры проверки корректности данных ввода и хранимой информации.
- TP – процедуры преобразования: процедуры преобразования изменяют состав множества всех контролируемых элементов данных путем редактирования, создания, ввода, удаления и т.п. В частности – редактирование писем и их распределение по папкам.

C1: все процедуры преобразования данных соответствуют определенному пользователю, что в свою очередь обеспечивает конфиденциальность хранимой пользователем информации.

C2: все процедуры преобразования реализованы таким образом, чтобы не изменять системные файлы и папки. Для процедуры преобразования «удаление» установлен список тех данных, на которые эта процедура не сможет воздействовать и сможет влиять только в рамках тех прав, которые были установлены для пользователя с конкретными логином и паролем (пользователь не может удалить информацию о другом сеансе).

E1: должна обеспечиваться на уровне программного средства, должен быть установлен контроль доступа в соответствии со списками, которые были установлены в правиле C2, о применении процедур преобразования к соответствующим CDI.

E2: соответствие логина и пароля с доступом к определенной информации, отождествление пользователей с предназначенным для них списком прав. Четкое разграничение функциональных возможностей и обязанностей для каждого пользователя системы.

C3: администратор имеет право изменять логин и пароль, а также права доступа к информации, но не может менять данные в базе.

E3: при попытке пользователя совершить какую-либо операцию, каждый раз производиться аутентификация пользователя (более того, аутентификация пользователя происходит каждые 10 мин).

C4: Каждая операция, совершаемая пользователем, записывается в журнал историй.

C5: Сначала производиться поиск логина , далее соответствующего пароля в базе, а затем определяются права доступа к информации.

E4: должно быть определено уполномоченное лицо – администратор системы. Он определяет права и контролирует работу системы.

Задание к работе

Необходимо для заданной информационной системы определить:

- CDI – контролируемые элементы данных.
- UDI – неконтролируемые элементы данных.
- IVP – процедуры контроля целостности.
- TP – процедуры преобразования.

Выделить объект(ы), в которых хранятся списки, определенные в правилах C2 и E2, а также хранится журнал регистрации событий. Указать кто может изменять списки, определенные с правилах C2 и E2.

Создать правила, которые соответствовали ли бы девяти правилам, определяющим взаимоотношения элементов данных и процедур в процессе функционирования системы. Провести проверку целостности системы с использованием теоретических принципов политики контроля целостности.

Варианты объектов исследования:

1. www-сервер;
2. ftp-сервер;
3. Почтовая база данных;
4. База данных Microsoft Access;
5. Операционная система;
6. Жесткий диск;
7. Сотовый телефон.

Контрольные вопросы

1. Каково назначение модели Кларка – Вилсона;
2. Какие подмножества объектов выделяются в модели Кларка-Вилсона;
3. Какие типы функций определены в модели Кларка-Вилсона. Какие требования и ограничение вводит модель Кларка-Вилсона на функции;
4. Перечислите правила модели Кларка-Вилсона.
5. Какова роль Администратора в модели Кларка-Вилсона.

Лабораторная работа 3

Изучение скрытого канала передачи данных

Краткая теория

Особый класс программ–шпионов могут составлять «недобросовестные» программы, которым пользователь добровольно доверяет свою персональную информацию, надеясь, что реализуемая в системе политика безопасности не позволит получить несанкционированный доступ к ней.

Примером могут быть какие–либо серверные программы, которые ведут обработку клиентской информации через сеть. Предположим, что политика безопасности такова, что данная серверная программа не может создавать файлы, в которые она теоретически могла бы записать приватную информацию клиентов и передать её в дальнейшем третьим лицам. Или, например, может создавать файлы, но доступ к ним никто кроме неё получить не может. Возникает иллюзия, что «недобросовестный» сервер не может распространить обрабатываемую им информацию. На практике же каналом передачи данных могут быть не только логические области памяти или диска (сегменты данных или файлов), но и реальные физические объекты, наличие или отсутствие которых может кодировать «0» и «1» соответственно. Существование или отсутствие данного объекта может периодически фиксироваться третьей стороной, тем самым она воспринимает какую–то кодовую последовательность, которая и может нести секретную информацию клиента.

В качестве физических объектов могут быть использованы:

- файлы – некоторые ОС, например UNIX, позволяют программе, которая обращается к файлу, получить информацию о том, существует ли он реально, даже если доступ к нему запрещён;
- загруженность процессора – сервер специально загружает процессор пустыми операциями, соответственно у остальных процессов увеличивается время отклика;
- интенсивность сетевого трафика;
- интенсивность обращений к диску.

Недостатками скрытого канала являются низкая скорость передачи данных, сильная зашумлённость и возможные ошибки синхронизации. Первый не так важен, а 2–й и 3–й устраняют, используя помехоустойчивое кодирование. Очевидно, что такой сложный метод шпионажа, как скрытые каналы, может быть использован лишь для получения особо ценных сведений.

Методов борьбы со скрытыми каналами как таковых не существуют. Все они носят скорее организационный характер и не направлены непосредственно на борьбу с данным видом угроз. Это:

- использование сертифицированных программных продуктов,
- отслеживания состояния работы вычислительного комплекса или отдельного ПК (периодическая спонтанная активность процесса, создание или удаление файлов может вызывать подозрения)
- минимизация передаваемых по сети секретных данных (их обработка по возможности должна вестись локально).

Пример скрытого канала

Пусть клиент вызывает некоторый сервис, передавая ему в качестве параметров конфиденциальную информацию, утечка которой нежелательна. Как следует ограничить поведение произвольного сервиса

Подчеркнем, что речь идет о создании "песочницы" для произвольной программы. Если ограничения окажутся нарушенными, выполнение сервиса должно аварийно завершаться.

Чтобы понять характер налагаемых ограничений, сначала исследуем возможные каналы утечки информации, выделяя следующие:

Если у сервиса есть память, он может сохранить клиентскую информацию, дождаться, когда его вызовет хозяин, и передать тому сохраненную информацию;

Сервис может записать информацию в постоянный файл в каталоге хозяина;

Сервис может создать временный файл (что само по себе вполне законно: как же без временных файлов?); хозяин может периодически проверять его (файла) существование и прочитать информацию прежде, чем сервис завершит работу и удалит все временное; Сервис может послать сообщение процессу, контролируемому хозяином;

Сервис может закодировать информацию в счете за свои услуги, поскольку хозяин должен получить копию этого счета; ограничения на формат счета способны свести объем утечки к нескольким десяткам бит, но полностью ликвидировать ее нереально;

Если могут возникать конфликты по ресурсам, сам факт конфликта может быть использован для передачи информации (чуть ниже мы подробнее рассмотрим этот канал утечки);

Сервис может варьировать отношение вычислительной активности к темпу подкачки страниц или числу операций ввода/вывода, кодируя таким способом информацию; параллельно работающий процесс способен наблюдать поведение системы и получать передаваемую информацию; это зашумленный канал, его пропускная способность невелика, но он есть, и им можно воспользоваться.

Всегда интересно не просто теоретически знать, что каналы утечки существуют, но и понять на практике, как они могут быть устроены. Пусть, например, один и тот же файл нельзя параллельно открыть из двух процессов (при попытке сделать это фиксируется ошибка). Данный факт можно использовать для побитной передачи информации. Две следующие

процедуры, написанные на АЛГОЛоподобном языке обеспечивают, соответственно, установку нужного бита и его проверку.

Листинг 1 setbit (file, bit);

```
  Boolean bit;
begin
  if bit = true then
    loop1: open (file, loop1)
  else
    close (file)
end;

Boolean procedure testbit (file);
begin
  testbit : = true;
  open (file, loop2);
  testbit := false;
  close (file)
loop2: end;
```

(К своему стыду, автор не знает, как окружение АЛГОЛ-программ реагирует на попытку закрыть неоткрытый файл).

Сейчас далеко не все помнят тонкости АЛГОЛа-60, поэтому нужно пояснить по крайней мере два момента. Во-первых, метки в АЛГОЛе являются допустимым типом данных, значения которого можно передавать в качестве параметров. Во-вторых, если (занятый) файл открыть не удалось, управление нелокально передается на метку, заданную в качестве второго аргумента процедуры open. Кстати, это более практичный способ обработки исключительных ситуаций, чем проверка кодов возврата, которую зачастую забывают сделать, что является одним из источников уязвимостей программных систем.

Теперь, располагая элементарными операциями, можно организовать передачу бита между процессами. Для этого используем три файла: data, sendclock и receiveclock.

В программе-сервисе может присутствовать такой фрагмент (написанный нами с некоторыми вольностями):

Листинг 2:

```
  setbit (data, bitbeingsent);
  while testbit (receiveclock) = false do
    ;
  setbit (sendclock, false);
```

```
  while testbit (receiveclock) = true do
    ;
```

Процесс-получатель может осуществлять прием бита так:

Листинг 3:

```
  while testbit (sendclock) = false do
    ;
  receivedbit : = testbit (data);
  setbit (receiveclock, true);
  while testbit (sendclock) = true do
    ;
  setbit (receiveclock, false);
```

Переходя к правилам ограничения (контролируемого выполнения), Лэмпсон прежде всего указывает, что ограничиваемая программа не должна иметь возможности сохранять информацию между вызовами. Для процедур это условие легко проверяется: оно означает отсутствие обращений к нелокальным переменным.

Если нелокальная память отсутствует, то для успешной реализации ограничения достаточно, чтобы ограничиваемая программа не вызывала других программ. Это правило полной изоляции по сути совпадает с моделью "песочницы" в версии JDK 1.0. Как показывают два последних из перечисленных выше примеров утечек, в числе прочих должны быть запрещены явные и неявные вызовы супервизора (ядра ОС). Попытка применить правило полной изоляции "насквозь" и потребовать, чтобы ограничивались все вызываемые программы, не проходит, поскольку супервизор нельзя ограничивать.

Чтобы исправить ситуацию, предлагается, как и следовало ожидать, поделить всех на чистых и нечистых, то есть на тех, кому доверяют, и тех, кого ограничивают. В результате получается следующее правило: если ограничиваемая программа вызывает недоверенную, то последнюю также необходимо ограничивать.

Дело осталось за малым — написать доверенный супервизор. Как показывают два последних из перечисленных выше примеров утечек, дело это непростое, поскольку используемые для этого каналы могут быть самыми неожиданными. Каналов утечки, конечно, на удивление много, но все-таки конечное число. Необходимо их перенумеровать, а потом и заблокировать. В качестве отправной точки предлагается следующая классификация каналов:

Разного рода память, управляемая супервизором, в которую может писать ограничиваемая программа (в рассматриваемых примерах — сервис), а читать — неограничиваемая (вскоре после записи или позднее); все приведенные выше примеры, кроме двух, относятся к этому классу;

Легальные каналы, используемые ограничиваемой программой (например, счет за обслуживания);

Скрытые каналы, вообще не предназначенные для передачи информации, (например, влияние сервиса на загруженность системы).

Предотвратить утечки через память довольно просто. Например, чтобы избавиться от блокировок при совместном доступе к ресурсам, можно при попытке записи копировать файл и бесконфликтно предоставлять копию для чтения ограничиваемой программой. (Заметим, что идея эта весьма глубока, только лучше при попытке записи открывать новую версию файла и писать в нее, применяя в дальнейшем механизмы конфигурационного управления.)

Для блокирования легальных и скрытых каналов предлагается следующий принцип маскирования: ограничиваемая программа должна позволять вызывающему полностью определять вывод в легальные и скрытые каналы. Таким образом каналы маскируются вызывающим (клиентом).

Задание на лабораторную работу

1) Предложить вариант реализации скрытого канала по памяти либо реализовать один из предложенных (по вариантам)

Варианты:

1. Загруженность процессора
2. Существование файла на диске.
3. Интенсивность IP-пакетов.
4. Интенсивность обращений к диску
5. Загруженность ОЗУ
6. Объём файлов каталога
7. Цвет рабочего стола

2) Реализовать предложенный вариант скрытого канала в виде программ. Программы должны быть снабжены интерфейсом для просмотра состояния передачи данных по скрытому каналу

3) Разработать программу, вносящую произвольные шумы в разработанный скрытый канал

4) Проанализировать возможность передачи данных по скрытому каналу в условиях зашумления, сделать вывод о влиянии интенсивности шума в скрытом канале на передачу данных

Контрольные вопросы

1) Дайте определение скрытого канала
2) Отличие скрытых каналов по времени и скрытых каналов по данным
3) Методы борьбы со скрытыми каналами по данным
4) Теоретические проблемы передачи данных по скрытым каналам
5) Какие угрозы, помимо утечек данных, несут скрытые каналы?

Указания к самостоятельной работе студентов

Содержание курса

№ п/п	Раздел (тема) дисциплины	Содержание
1	Основы формальной теории защиты информации	Авторизация. Методы идентификации и аутентификации пользователя. Классификация задач, решаемых механизмами идентификации и аутентификации. Парольная защита. Механизмы парольной защиты. Угрозы преодоления парольной защиты. Способы усиления парольной защиты. Классификация задач, решаемых механизмами идентификации и аутентификации.
2	Формальные модели управления доступом	Формальные модели управления доступом. Дискреционная модель Харрисона-Руззо-Ульмана Формальные модели управления доступом. Мандатная модель Белла-ЛаПадулы. Формальные модели управления доступом. Мандатная модель Мак-Лина.
3	Формальные модели целостности	Формальные модели управления целостности. Модель политики контроля целостности. Защищенные домены. Концепция защищенного ядра. Исследование корректности систем защиты. Методы верификации. Модель Кларка-Вилсона.
4	Совместное использование моделей безопасности	Основные способы и каналы утечки информации в КС. Преодоление парольной защиты. Основные способы несанкционированного доступа. Способы внедрения программных закладок и компьютерных вирусов. Архитектура защиты современных операционных систем. Основные механизмы защиты ОС.
5	Скрытые каналы передачи информации	Контроль корректности функционирования механизмов защиты. Методы контроля целостности. Метод уровневого контроля списков санкционированных событий. Контроль за действиями пользователей.

График выполнения СРС
Самостоятельная работа студентов

№	Наименование раздела учебной дисциплины	Срок выполнения	Время, затрачиваемое на выполнение СРС, час.
1.	Основы формальной теории защиты информации	1-3 недели	8
2.	Формальные модели управления доступом	4-7 недели	8
3.	Формальные модели целостности	8-11 недели	14
4.	Совместное использование моделей безопасности	12-14 недели	10
5.	Скрытые каналы передачи информации	15-18 недели	14

Итого	46

Методические указания для обучающихся по освоению дисциплины

Основными видами аудиторной работы студента при изучении дисциплины «Теоретические основы компьютерной безопасности» являются лекции, лабораторные занятия. Студент не имеет права пропускать занятия без уважительных причин.

На лекциях излагаются и разъясняются основные понятия темы, связанные с ней теоретические и практические проблемы, даются рекомендации для самостоятельной работы. В ходе лекции студент должен внимательно слушать и конспектировать материал.

Изучение наиболее важных тем или разделов дисциплины завершают лабораторные и практические занятия, которые обеспечивают: контроль подготовленности студента; закрепление учебного материала; приобретение опыта устных публичных выступлений, ведения дискуссии, в том числе аргументации и защиты выдвигаемых положений и тезисов.

Лабораторному занятию предшествует самостоятельная работа студента, связанная с освоением материала, полученного на лекциях, и материалов, изложенных в учебниках и учебных пособиях, а также литературе, рекомендованной преподавателем.

Качество учебной работы студентов преподаватель оценивает по результатам тестирования, собеседования, защиты отчетов по лабораторным и практическим работам.

Преподаватель уже на первых занятиях объясняет студентам, какие формы обучения следует использовать при самостоятельном изучении дисциплины «Теоретические основы компьютерной безопасности»: конспектирование учебной литературы и лекции, составление словарей понятий и терминов и т. п.

В процессе обучения преподаватели используют активные формы работы со студентами: чтение лекций, привлечение студентов к творческому процессу на лекциях, промежуточный контроль путем отработки студентами пропущенных лекции, участие в групповых и индивидуальных консультациях (собеседовании). Эти формы способствуют выработке у студентов умения работать с учебником и литературой. Изучение литературы и справочной документации составляет значительную часть самостоятельной работы студента. Это большой труд, требующий усилий и желания студента. В самом начале работы над книгой важно определить цель и направление этой работы. Прочитанное следует закрепить в памяти. Одним из приемов закрепление освоенного материала является конспектирование, без которого немыслима серьезная работа над литературой. Систематическое конспектирование помогает научиться правильно, кратко и четко излагать своими словами прочитанный материал.

Самостоятельную работу следует начинать с первых занятий. От занятия к занятию нужно регулярно прочитывать конспект лекций, знакомиться с соответствующими разделами учебника, читать и конспектировать литературу по каждой теме дисциплины. Самостоятельная работа дает студентам возможность равномерно распределить нагрузку, способствует более глубокому и качественному усвоению учебного материала. В случае необходимости студенты обращаются за консультацией к преподавателю по вопросам дисциплины «Теоретические основы компьютерной безопасности» с целью усвоения и закрепления компетенций.

Основная цель самостоятельной работы студента при изучении дисциплины «Теоретические основы компьютерной безопасности» - закрепить теоретические знания, полученные в процессе лекционных занятий, а также сформировать практические навыки самостоятельного анализа особенностей дисциплины.

Вопросы для самоконтроля

Тема 1. Основы формальной теории защиты информации.

1. Что такое формальный язык, из чего он состоит?
2. Основные виды доступа к информации и их интерпретация?
3. Что такое монитор безопасности обращений и какие требования к нему предъявляются?
4. Что такое информационный поток?
5. Дайте определение понятию «преобразование информации»

Тема 2. Формальные модели управления доступом.

1. Что такое формальная модель безопасности?
2. Какие постулаты положены в основу модели Харрисона-Руззо-Ульмана?
3. Опишите элементарные операции модели Харрисона-Руззо-Ульмана.
4. Как происходит создание файла на языке модели Харрисона-Руззо-Ульмана?
5. Как происходит редактирование файла на языке модели Харрисона-Руззо-Ульмана?
6. Что такое «утечка данных» на языке модели Харрисона-Руззо-Ульмана?
7. Какой принцип управления доступом положен в основу модели Беллы-ЛаПадулы?
8. Опишите правила модели Беллы-ЛаПадулы.
9. Что такое уровень секретности?
10. Применимость модели Беллы-ЛаПадулы в реальных информационных ситсемах.

Тема 3. Формальные модели целостности.

1. Назначение модели Кларка-Вилсона.
2. Примитивы модели Кларка-Вилсона.
3. Правила модели Кларка-Вилсона.
4. Базовые правила модели Биба.

5. Изобразите диаграмму информационных потоков в модели Биба.

Тема 4. Совместное использование моделей безопасности.

1. Причины, вынуждающие использовать несколько моделей безопасности
2. Как выделить общие компоненты при использовании моделей Белла-Лападулы и Биба?
3. Как построить единую решётку уровней целостности и секретности при использовании моделей Белла-Лападулы и Биба?
4. Что такое ролевое управление доступом?
5. Критерии безопасности системы при использовании ролевой модели.

Тема 5. Скрытые каналы передачи информации.

1. Понятие скрытого канала
2. Назовите виды скрытых каналов и приведите их примеры
3. Особенности реальных информационных систем, позволяющие реализовывать скрытые каналы
4. Опишите метод разделяемых ресурсов Кемерера
5. Назовите особенности использования методов сигнатурного анализа при выявлении скрытых каналов по данным
6. Предложите несколько вариантов монитора, обнаруживающего скрытые каналы по данным