Факультет компьютерных технологий и прикладной математики

Кафедра вычислительных технологий

02.03.02

Информационная безопасность

Лабораторная работа № 2

Тема: Модели информационной безопасности.

Цель работы: получить знания и практические навыки построения моделей информационной безопасности.

Задание: необходимо разработать алгоритмы и программы решения двух задач на языке C++.

Указания к работе. Каждый студент должен выполнить задачи по вариантам, изложенным ниже. За проделанную работу студент может получить оценку от «неудовлетворительно» до «отлично». На занятии студенты решают первую задачу по вариантам, изложенным ниже. Студент разрабатывает алгоритм и программу решения задачи на языке C++ на лабораторном занятии. За работу на занятии студент может получить оценку от «неудовлетворительно» до «хорошо». При построении алгоритма первой задачи на занятии студент получает оценку «удовлетворительно» или «хорошо». Для получения оценки «хорошо» на занятии студент должен выполнить первую задачу и ответить на теоретические вопросы. Оценка «удовлетворительно» ставиться, если студент на занятии решает первую задачу полностью, но не может ответить на теоретические вопросы или выполняет как минимум половину первой задачи и в состоянии ответить на теоретические вопросы. Оценку «неудовлетворительно» студент получает, если выполняет минимум половину первой задачи или в состоянии ответить на теоретические вопросы.

Оценка «отлично» может быть выставлена ТОЛЬКО если студент получил оценку «хорошо» на занятии и сделал вторую задачу на следующее занятие.

Если студент не получил оценку, то ему будет необходимо подготовить отчёт по задаче и реализовать программу, используя систему управления версиями Git и размещая её репозиторий на GitHub.

При защите отчёта ОБЯЗАТЕЛЬНО ответить на теоретические вопросы по прошедшим лекциям.

ОЦЕНКУ ЗА ПРОДЕЛАННУЮ РАБОТУ МОЖНО ПОВЫСИТЬ ДО СДАЧИ СЛЕДУЮЩЕЙ ЛАБОРАТОРНОЙ РАБОТЫ.

Вариант: ((N - 1) % 17) + 1, где % – деление с остатком, N – номер в подгруппе.

Вариант 1. Задачи 1, 9

Вариант 2. Задачи 3, 8

Вариант 3. Задачи 5, 6

Вариант 4. Задачи 7, 4

Вариант 5. Задачи 1, 8

Вариант 6. Задачи 3, 6

Вариант 7. Задачи 5, 4

Вариант 8. Задачи 7, 2

Вариант 9. Задачи 1, 6

Вариант 10. Задачи 3, 2

Вариант 11. Задачи 1, 4

Вариант 12. Задачи 5, 2

Вариант 13. Задачи 1, 2

Вариант 14. Задачи 3, 9

Вариант 15. Задачи 5, 8

Вариант 16. Задачи 7, 6

Вариант 17. Задачи 5, 9

Дискреционная модель доступа с условием Белла-Лападуллы

1. Для заданных натуральных чисел n и m матрица A[1:n; 1:m] прав доступа субъектов $s_1, s_2, \ldots, s_n$ к объектам $o_1, o_2, \ldots, o_m$ предполагается заданной так, что P = A[i, j], где $P \subseteq \{$‘r’, ‘w’$\}$; ‘r’ – полномочие чтения субъектом s_i объекта o_j, ‘w’ – полномочие записи субъектом s_i в объект o_j. Кроме того, предполагаются заданными целочисленные массивы LS и LO: LS[1:n] так что LS[i] – значение уровня допуска субъекта s_i; LO[1:m] так что LO[j] – значение уровня секретности объекта o_j. Составить программу, которая проверяет, верно ли что компьютерная система CS = (n, m, A, LS, LO) соответствует критерию безопасности Белла-Лападулы.

2. Для заданных натуральных чисел n и m матрица A[1:n; 1:m] прав доступа субъектов $s_1, s_2, \ldots, s_n$ к объектам $o_1, o_2, \ldots, o_m$ предполагается заданной так, что P = A[i, j], где $P \subseteq \{$‘r’, ‘w’$\}$; ‘r’ – полномочие чтения субъектом s_i объекта o_j, ‘w’ – полномочие записи субъектом s_i в объект o_j. Пусть целочисленный массив LS[1:n] предназначен для хранения уровней допуска (это числа) субъектов, а LO[1:m] – целочисленный массив, предназначенный для хранения уровней секретности объектов (это числа). Составить программу, которая по заданным n, m и A[1:n; 1:m] подбирает такие значения для элементов массивов LS и LO, чтобы компьютерная система CS = (n, m, A, LS, LO) удовлетворяла критерию безопасности Белла-Лападулы или сообщала, что при заданной матрице A[1:n; 1:m] требуемых наполнений массивов LS и LO не существует.

Дискреционная модель доступа Харрисона-Руззо-Ульмана, модели на основе матрицы доступа

3. Пусть задана матрица прав доступа A[1:n; 1:m] для заданных натуральных n, m, определяющих полномочия субъектов $s_1, s_2, \ldots, s_n$ к объектам $o_1, o_2, \ldots, o_m$ так что P = A[i, j], где $P \subseteq \{$‘r’, ‘w’, ‘o’, ‘x’$\}$ ‘r’ – код права чтения, ‘w’ – код права записи, 'o' – код права владения, 'x' – код права

исполнения. Элементарные команды модели HRU можно кодировать следующим образом:

команду Create object o_j как cco = ‘O+’ image_j

команду Create subject s_j как ccs = ‘S+’ image_j

команду Destroy object o_j как cdo = ‘O-’ image_j

команду Destroy subject s_j как cds = ‘S-’ image_j

команду Enter p into A[i, j] как cer_p = ‘p+’ image_i image_j, где p – символ из алфавита {‘r’, ‘w’, ‘o’, ‘x’}

команду Delete p from A[i, j] как cdr_p = ‘p-’ image_i image_j, где p – символ из алфавита {‘r’, ‘w’, ‘o’, ‘x’}

Команда с условием из модели HRU:

K = **if** p in A[i, j] **then** Elem_Command,

где Elem_Command – это элементарная команда из модели HRU (их список перечислен выше) можно кодировать в виде

Code_K : 'p' image_i image_j ‘–‘ Code_Element_Command,

где Code_Element_Command – это элемент из множества {cco, ccs, cdo, cds, cer_p, cdr_p}.

Тогда программа в модели HRU – это ее представление в виде последовательности:

$Code\_K_1$

$Code\_K_2$

…

$Code\_K_t$

Составить программу-интерпретатор, которая в качестве входных данных принимает натуральные числа n, m, матрицу прав доступа A[1:n; 1:m] – все это задается в одном файле environ – это файл среды; представление программы:

$Code\_K_1$

$Code\_K_2$

…

$Code\_K_t$

Это содержимое другого файла prgrm. Программа-интерпретатор исполняет команды из файла prgrm, а результирующую матрицу A выводит в rzlt – файл.

4. Составить программу, которая на основе данных:

1) n – натуральное число, количество субъектов;

2) m – натуральное число, количество объектов;

3) A[1:n; 1:m] – матрица прав доступа, такая, что $1 \le i \le n$, $1 \le j \le m$, $P = A[i, j]$, $P \subseteq \{\text{‘r’ , ‘w’}\}$, ‘r’ – право чтения субъектом s_i объекта o_j, ‘w’ – право записи в объект o_j субъектом s_i.

Данные заданы в текстовом файле access_matr при их подходящем представлении (например, клетка матрицы – это строка символов – прав доступа) формирует следующие результаты:

1) Список объектов, доступ к которым не задан никакому субъекту (ни по чтению, ни по записи).

2) Список субъектов, не обращающихся ни к каким объектам (ни по чтению, ни по записи).

3) Список субъектов, имеющих полный доступ ко всем объектам.

4) Список $C_1, C_2, \ldots, C_t$, где C_i – список субъектов, которые все имеют право записи в некоторый один объект o_j (т.е. все субъекты из C_i могут вступить в потенциальный конфликт при одновременной записи в объект o_j).

5) Список субъектов, каждый из которых имеет полный доступ только к одному объекту, а к другим объектам доступа либо нет, либо только право чтения.

Модель на основе матрицы доступа

5. Составить программу, которая по входным данным из одного или нескольких текстовых файлов формирует в оперативной памяти «разряженную» матрицу A[1:n; 1:m] прав доступа, n – количество субъектов

доступа, $3 \le n \le 10000$; m – количество объектов доступа, $3 \le m \le 10000$; для $1 \le i \le n$, $1 \le j \le m$, P = A[i, j], $P \subseteq$ {‘r’, ‘w’}, ‘r’ – код права чтения, ‘w’ – код права записи. Матрица считается “разряженной”, если более половины ее клеток хранят одинаковое значение (например, отсутствие любого доступа, назовем такие клетки стандартными), которые, таким образом, можно не хранить вообще в матрице, считая его известным заранее. Существуют: списочная организация хранения нестандартных клеток, векторная организация (одномерные массивы). В зависимости от выбранной организации представления A[1:n; 1:m] определяется способ задания входных данных. Программа должна обеспечивать:

1) Редактирование содержимого любой (т.е. стандартной или нестандартной) клетки A[i, j] матрицы, т.е. добавление/удаление заданного права в эту клетку.

2) Добавление нового субъекта в матрицу с указанием пользователю под каким номером зарегистрирован новый субъект в матрице. Например, команда:

add Bell.

Ответ программы: Bell has been registrated as 1111.

3) Удаление существующего субъекта из матрицы, например, delete 1111.

4) Вычисление процента заполненности матрицы (с учетом стандартных клеток при подсчете общего числа клеток, но исключая стандартные клетки при подсчете реально хранимых клеток).

5) Формирование списка субъектов, имеющих некоторый доступ к заданному объекту t (t – входное данное).

Модель Chinese Wall (Китайская стена)

6. Составить программу, реализующую политику Chinese Wall (Китайская стена). Входными данными для программы являются: n – количество субъектов s_1, s_2, …, s_n доступа к объектам; m – количество

объектов доступа o_1, o_2, …, o_m; f – число фирм F_1, F_2, …, F_f, являющихся собственниками объектов. Каждая фирма представлена своим портфелем, в который входят какие-то из объектов, так, что каждый объект должен входить в портфель только одной фирмы. Фирмы могут быть конкурентами, CI_1, CI_2, …, CI_s – классы конфликта интересов; в каждый класс CI_j входят номера (или названия) тех фирм F_s, которые конфликтуют между собой. Какие-то фирмы могут быть членами разных классов CI, какие-то фирмы могут не членствовать ни в каких классах (они ни с кем не конфликтуют). Программа должна выполнять директивы:

1) **start** – очищает историю выполненных доступов субъектов к объектам;

2) **read s_i o_j** – выполнить чтение объекта o_j субъектом s_i, чтение возможно, если объект o_j принадлежит к такому классу CI, к которому s_i еще не обращался, или o_j принадлежит портфелю данных той F_t, к которому s_i уже обращался; программа должна вывести:

1) **accepted** – подтверждение выполнения;

2) **refused** – в случае отказа.

3) **write s_i o_j** – выполнить запись в объект o_j субъектом s_i, запись возможна, если s_i может выполнить чтение o_j согласно правилу чтения и s_i не читал никакого объекта из портфеля данных другой фирмы (отличной от собственника o_j), которая входит в некоторый в один класс CI с фирмой-собственником o_j; Программа должна вывести:

1) **accepted** – подтверждение выполнения;

2) **refused** – в случае отказа.

4) **report** s_i – выдать отчет по выполненным операциям субъекта s_i с указанием объектов и фирм-собственников этих объектов;

5) **report** o_j – выдать отчет по операциям, выполненным с этим объектом разными субъектами;

6) **brief_case** F_i – выдать список объектов, находящихся в портфеле фирмы F_i.

Модель дискреционной политики безопасности Take-Grant

7. Модель дискреционной политики безопасности Take-Grant основана на графе доступов G = (S, O, E), где S – конечное множество субъектов доступа, O – конечное множество объектов доступа (считается, что субъект – это активный объект, так что $S \subset O$), $E \subseteq S \times O$. Кроме того, задана конечная совокупность R полномочий (прав доступа), среди которых выделяются права:

1) g – право предоставления субъекту некоторой совокупности прав S от другого субъекта, который обладает S в отношении определенного объекта;

2) t – право взятия субъектом определенной совокупности прав S у другого субъекта, который обладает S в отношении определенного объекта.

Каждая дуга из E снабжается пометкой – подмножеством прав из R. Информационные процессы, подчиняющиеся политике безопасности, соответствующей модели Take-Grant, меняют разметку дуг в графе доступов с течением времени. В этой последовательности графов доступов G_0, G_1, G_2, …, в которых изменяются множества S_i, O_i, E_i и разметка дуг, интерес представляет проверка истинности следующего условия:

Possible_Access(A, x, y), где $x \in S$, $y \in O$, $A \subseteq R$, которое истинно тогда и только тогда, когда существует такая конечная последовательность переходов графов доступа G_0, G_1, G_2, …, G_n благодаря подходящей последовательности команд модели Take-Grant, в результате которой, если $(x, y, A) \notin E_0$ (т.е. в начале x не обладал набором полномочий A по отношению к y), то $(x, y, A) \in E_n$. В случае, когда в графе доступов все вершины представляют субъектов, проверку Possible_Access(A, x, y) обеспечивает следующий Критерий (теорема): Possible_Access(A, x, y) истинно ⇔

1) существует конечное множество субъектов s_1, s_2, …, s_t, таких что $(s_i, y_i, A_i) \in E_0$, $1 \le i \le t$, и $A = A_1 \cup A_2 \cup \ldots \cup A_t$

2) субъект x соединен tg-путем в графе G_0 с каждым из s_i.

Определение tg-пути: вершины a и b соединены дугами, возможно через конечное число промежуточных вершин, при этом каждая дуга на таком пути помечена (в числе прочих прав) правом t или g, а направления самих дуг безразличны.

Составить программу, которая по исходной конфигурации графа доступов G, заданной во входном текстовом файле, по произвольной тройке A, x, y могла бы вычислить значение

Possible_Access(A, x, y) применяя описанный критерий. В процессе сеанса программа должна обеспечивать возможность вычисления Possible_Access для различных троек A, x, y.

8. Нахождение «островов» в модели Take-Grant. Политика безопасности описывается графом доступов и его изменениями с помощью подходящих команд модели. См. введение к задаче № 7. В графе доступов G = (S, O, E) островом Island называется максимальный tg-связный подграф, вершины которого состоят только из вершин-субъектов. В tg-связный графе любая пара его вершин соединена tg-путем (см. определение к задаче № 7). Максимальность подграфа означает, что множество его вершин не входит как собственное подмножество ни в какое другое множество вершин графа с таким же свойством. Составить программу, которая по графу доступов G, заданному во входном текстовом файле, вычисляет все «острова» I_1, I_2, …, I_t этого графа G. При этом описание каждого острова $I_j = (S_j, E_j)$, где дуги из E_j помечаются правами t или g и, возможно, еще какими-то правами, записываются в отдельном выходном текстовом файле island_j.

9. Алгоритм поиска моста в графе доступов, когда острова уже выявлены. Разновидностью дискреционных моделей безопасности является модель Take-Grant. В этой модели система правоотношений содержит два особых права: take (сокращенно t) – брать права на объект у другого субъекта и grant (сокращенно g) – давать права на объект другому субъекту. Сама

компьютерная система представляется в виде ориентированного графа (графа доступов), в котором вершинами являются субъекты и объекты, а дугами – права доступов, установленные между соответствующими субъектами и объектами. Анализ информационной безопасности в модели Take-Grant проводится путем исследования графа доступов и поиска в нем определенных структур. Примерами таких структур являются острова и мосты.

Определение 1. Островом в графе доступов называется подграф, состоящий из вершин-субъектов, причем эти вершины соединены между собой дугами, помеченными правом take либо grant (направление дуг не учитывается).

Определение 2. Мостом в графе доступов называется проходящий по вершинам-объектам (субъекты считаются объектами) путь, каждая дуга которого содержит метку take или grant, при этом словарная запись пути имеет вид $\vec{t}^*$, $\overleftarrow{t}^*$, $\vec{t}\vec{g}\overleftarrow{t}^*$, либо $\vec{t}\overleftarrow{g}\overleftarrow{t}^*$. Ниже приводится алгоритм поиска мостов, если острова в графе уже найдены.

Пусть в графе доступов G = (V, E) (где V =O – множество вершин-объектов, E – множество дуг) уже известны острова и необходимо найти мост между островами $I_1 \subset S$ (S – множество субъектов) и $I_2 \subset S$. Будем искать мост между вершинами $s \in I_1$ и $f \in I_2$. Опишем алгоритм поиска моста, основанный на поиске в глубину.

Введем шесть цветов для раскраски вершин графа: *красный*, *зеленый*, *синий*, *белый*, *серый* и *черный*. Пусть в начале работы алгоритма все вершины графа G окрашены в *белый* цвет. Окраску вершины x обозначим c(x). Для временного запоминания посещенных вершин используется стек.

Утверждение 1. Алгоритм корректно распознает наличие моста в графе доступов и заканчивает свою работу за конечное число шагов независимо от того, существует мост в графе или нет.

Алгоритм. Поиск_моста_между(s – стартовая вершина-субъект, f – финальная вершина-субъект)

1. Сделать текущей вершину s.

2. Окрасить текущую вершину в *серый.*

3. **Если** вершина f не *белая* **то** завершить алгоритм – мост существует.

4. **Для всех** вершин u, смежных с текущей, выполнить:

5. **Если** c(u) = *белый* и текущая вершина *серая* или *красная* и существует дуга $\vec{t}$

от текущей вершины до u, **то**

окрасить u в *красный* и перейти на шаг 11.

6. **Если** c(u) = *белый* и текущая вершина *серая* или *красная* и существует дуга $\overleftarrow{g}$

или дуга $\vec{g}$ от текущей вершины до u, **то** окрасить u в *зеленый* и перейти на шаг 11.

7. **Если** c(u) = *белый* и текущая вершина *серая* или *зеленая* или *синяя*, и существует дуга $\overleftarrow{t}$

от текущей вершины до u, **то** окрасить u в *синий* и перейти на шаг 11.

8. В остальных случаях окрасить текущую вершину в *черный*.

9. **Если** c(s) = *черный* **то** завершить алгоритм – «Моста не существует».

10. **иначе** сделать текущей первую вершину из стека, перейти на шаг 3.

11. Положить текущую вершину в стек, сделать текущей вершину u, перейти на шаг 3.

Составить программу, входными данными для которой является граф доступов, заданный как в задачах № 7 или № 8, файлы, представляющие острова I_p, см. описание задачи № 8, пара натуральных чисел a и b – номера двух произвольных островов I_a и I_b. Программа определяет факт наличия моста между островами I_a и I_b согласно представленному алгоритму.