Математические структуры и моделирование
2011, вып. 23, с. 99–104

УДК 004.056.53

АЛГОРИТМ ПОИСКА МОСТОВ ТИПА $\overrightarrow{t^*}$ И $\overleftarrow{t^*}$ В ГРАФЕ ДОСТУПОВ ДЛЯ ДИСКРЕЦИОННОЙ МОДЕЛИ БЕЗОПАСНОСТИ TAKE-GRANT

Д.М. Бречка

В статье приводится алгоритм поиска мостов типа $\overrightarrow{t^*}$ и $\overleftarrow{t^*}$ в графе доступов для дискреционной модели безопасности Take-Grant, основанный на классическом алгоритме поиска в ширину.

Введение

Модель Take-Grant является одной из первых и наиболее глубоко проработанных в математическом плане дискреционных моделей безопасности [1–3]. Одним из наиболее существенных достижений классической модели Take-Grant можно считать выработанные условия, при соблюдении которых компьютерная система считается защищённой. Эти условия можно найти в двух теоремах об истинности предиката *«возможен доступ»*, сформулированных и доказанных для модели Take-Grant [2, 3]. В классической модели способ проверки выполнимости этих условий никак не оговаривается. Наиболее простым способом проверки, в концептуальном плане, можно считать полный перебор вершин и дуг графа доступов. Однако, как известно, алгоритм перебора имеет экспоненциальную вычислительную сложность, это означает, что выполняться такой алгоритм на обычной ЭВМ будет слишком медленно.

В более поздних работах, посвящённых модели Take-Grant, предлагаются альтернативные способы проверки защищённости компьютерной системы, не основывающиеся на предложенных в классической модели условиях. В работах [4, 5] предлагаются алгоритмы, имеющие полиномиальную сложность $O(N^3)$.

Данная статья является продолжением работ [6, 7], в которых описываются способы проверки безопасности компьютерной системы, основанные на тех условиях, что сформулированы в классической модели Take-Grant. В статье описывается полиномиальный алгоритм для поиска мостов типа $\overrightarrow{t^*}$ и $\overleftarrow{t^*}$ в графе доступов. Предлагаемый алгоритм основан на классическом алгоритме поиска в ширину [8].

Copyright © 2011 **Д.М. Бречка**
Омский государственный университет им. Ф.М. Достоевского
E-mail: dbrechkawork@yandex.ru

1. Начальные условия и обозначения

Исходя из условий, сформулированных в модели Take-Grant, для того чтобы предикат *«возможен доступ»* был истинен в произвольном графе, необходимо, чтобы в этом графе были известны острова (tg-связные подграфы из вершин-субъектов), мосты, начальные и конечные пролёты мостов (пути в графе заданного вида). В данной статье ограничимся поиском мостов типа $\overrightarrow{t^*}$ и $\overleftarrow{t^*}$ — путей в графе, каждая дуга которых содержит метку $\overrightarrow{t}$ или $\overleftarrow{t}$ соответственно.

Пусть в графе уже известны острова, способ их нахождения приводится в [6, 7]. Нам нужно найти мост между двумя островами I_1 и I_2. Введём ряд обозначений. Будем обозначать дугу, содержащую метку $\overrightarrow{t}$, между вершинами графа e_i и e_j через $\overrightarrow{t}(e_i, e_j)$. Начальную вершину моста будем обозначать через s, а конечную — через f. Множество всех вершин-объектов исходного графа обозначим через O.

2. Мост $\overrightarrow{t^*}$

Рассмотрим поиск моста типа $\overrightarrow{t^*}$. Для начала опишем алгоритм неформально. В начале работы алгоритм разбивает все множество вершин-объектов исходного графа на два подмножества O_r и O_i. В множество O_r попадают те вершины, до которых существует мост заданного вида, все остальные вершины заносятся в множество O_i. В самом начале работы алгоритма в множестве O_r находится только одна вершина — s (начальная вершина). Далее алгоритм просматривает все дуги графа, инцидентные вершинам из множества O_r, и если обнаруживается дуга вида $\overrightarrow{t}$, соединяющая вершину e_r из O_r с вершиной e_i из O_i, то e_i удаляется из O_i и заносится в O_r. После просмотра всех дуг графа в O_r может быть занесено сразу несколько вершин, то есть мощность O_r увеличится на некоторое число, а мощность O_i уменьшится на это же число. Если вершина f попадает в множество O_r, то это значит, что в графе существует путь заданного вида и алгоритм заканчивает работу. Иначе, процедура повторяется для измеренных O_r и O_i. Однако возможна ситуация, когда после просмотра всех дуг в O_r не будет добавлено ни одной вершины. Это возможно, когда между вершинами из O_r и вершинами из O_i не существует дуг вида $\overrightarrow{t}$. Чтобы отловить данную ситуацию, нужно проверить мощности множеств, которыми мы оперируем: если мощности множеств не изменились, значит, необходимо закончить работу алгоритма с выдачей сообщения о том, что моста, заданного между рассматриваемыми островами, не существует.

Формально алгоритм будет состоять из трёх основных этапов. Перед началом работы алгоритма разобьём множество O на два подмножества O_r и O_i, причём $O = O_r \cup O_i$.

Этап 1. Вершина s заносится в множество O_r, все остальные вершины заносятся в O_i.

Этап 2. Просматриваются все дуги графа, началом которых являются вершины из O_r, если существует $\overrightarrow{t}(e_r, e_i)$, то e_i заносится в множество O_r и

удаляется из O_i. Здесь $e_r \in O_r$, $e_i \in O_i$. Когда все дуги, инцидентные вершинам из O_r множества, будут просмотрены, переходим на третий этап.

Этап 3. Если после выполнения этапа 2 вершина f оказалась во множестве O_r, то алгоритм заканчивает свою работу: мост заданного вида в графе существует. Если после выполнения этапа 2 мощности множеств O_r и O_i не изменились, алгоритм также заканчивает работу: моста заданного вида в графе не существует. В противном случае — возвращаемся на этап 2.

Замечание 1. В зависимости от задачи можно предусмотреть разные реализации алгоритма. Например, если требуется лишь показать наличие или отсутствие моста между указанными вершинами, то приведённого выше описания будет достаточно — алгоритм выдаёт сообщение о результатах своей работы. Если же требуется найти сам мост, то необходимо дополнительно каким-либо образом поддерживать множество отмеченных вершин и дуг. Например, после каждого шага второго этапа строить граф путей (который не обязательно будет деревом) или определённым образом окрашивать выбранные вершины и дуги.

Проведём оценку работы предложенного алгоритма. Пусть исходный граф содержит N вершин. Так как граф ориентирован, то в случае его полносвязности количество дуг в нем будет равно $N(N-1)$. Количество повторений этапа 2 можно ограничить количеством вершин графа, так как в случае если после каждого выполнения этапа в множество O_r будет заноситься по одной вершине, то мост в графе будет найден за N шагов. В общем случае в множество O_r за каждое выполнение этапа 2 будет заноситься более одной вершины, то есть мост будет найден меньше чем за N шагов. Если же моста не существует, то на каком-то шаге не найдётся дуги нужного вида и алгоритм ничего не занесёт в множество O_r, то есть мощность множества останется неизменной, тогда алгоритм прервётся с выдачей соответствующего сообщения. Таким образом, сложность работы алгоритма можно оценить как или $O(N^3)$.

Теорема 1. *Приведённый выше алгоритм корректно находит мост типа* $\overrightarrow{t^*}$.

Доказательство. Во-первых нужно доказать, что алгоритм вообще закончит свою работу (сходимость алгоритма), во-вторых — что алгоритм найдёт мост нужного вида (корректность алгоритма).

Сходимость можно показать исходя из того, что множество вершин-объектов исходного графа конечно. Алгоритм разбивает множество O на O_i и O_r, причём $O = O_r \cup O_i$. На каждом шаге второго этапа алгоритма либо какая-то вершина удаляется из O_i и заносится в O_r, при этом мощности обоих множеств соответственно изменяются, либо перемещения вершин не происходит — мощности множеств не меняются, и алгоритм заканчивает работу.

Пусть $|O| = M$. Возможно три варианта развития событий.

1. Все вершины из O_i будут перенесены в O_r, тогда на следующем шаге алгоритм закончит свою работу, так как мощности множеств менять будет уже невозможно. Всего алгоритм совершит $M + 1$ шагов.

2. На шаге $k < M$ алгоритм обнаружит мост. Алгоритм закончит работу, при этом совершит k шагов.

3. На шаге $k < M$ алгоритм обнаружит, что мощности множеств O_i и O_r не изменились. Алгоритм закончит работу, при этом совершит k шагов.

Таким образом, алгоритм в любом случае закончит работу независимо от того, присутствуют в графе мосты или нет.

Корректность алгоритма покажем индукцией по длине моста (n). В качестве базы индукции мы не можем выбрать $n = 1$, так как это бы означало, что начальная и конечная вершина моста соединены дугой $\vec{t}$ и, следовательно, принадлежат одному острову. Поэтому в качестве базы выберем $n = 2$ и покажем, что алгоритм найдёт такой мост.

Ситуация, когда в графе присутствует мост длины 2, изображена на рисунке 1. С вершиной s связана как минимум одна дуга вида $\vec{t}$, соединяющая вершины s и x. С вершиной x, в свою очередь, связана как минимум одна дуга вида $\vec{t}$, соединяющая x и f.

Вершина s заносится в множество O_r на первом этапе алгоритма. На втором этапе, при проверке всех дуг вида $\vec{t}$, инцидентных вершинам из множества O_r, будет обнаружена вершина x. Эта вершина будет занесена в O_r и удалена из O_i. Однако мост еще не будет найден. Он обнаружится только после повторного выполнения второго этапа, когда вновь будут просматриваться дуги вида $\vec{t}$, инцидентные вершинам из множества O_r, т.е. будет обнаружена дуга, связывающая x и f.

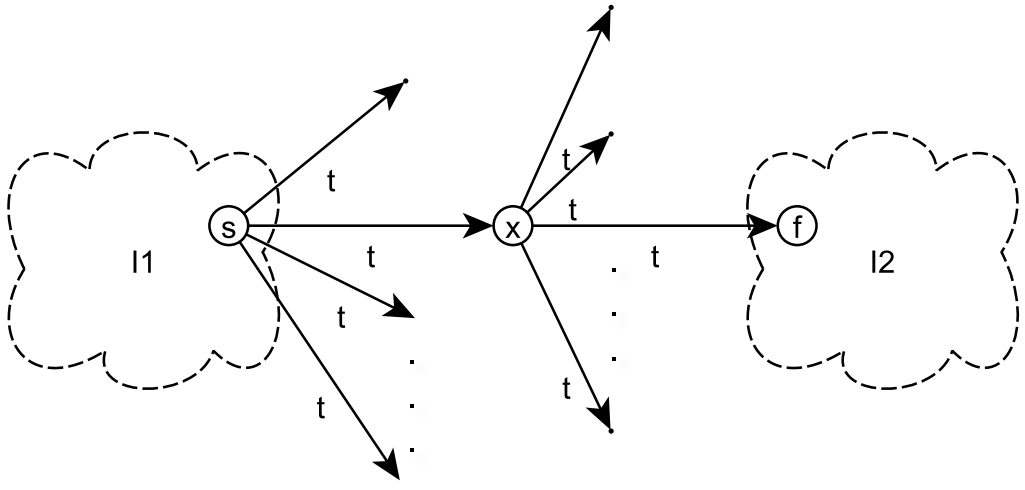

Рис. 1. Мост $\vec{t}^*$ длины 2

В качестве предположения индукции выберем утверждение, что для длины моста $n < k$, где $k > 2$ алгоритм находит мост правильно.

Шаг индукции: пусть длина моста равна k, алгоритм выполнил $k-2$й шаг, и на этом шаге в множество O_r были занесены вершины $x_1, x_2, \ldots x_m$ (рис. 2). Согласно предположению индукции, до каждой из вершин x_j мост найден правильно. Так как длина моста равна k, это означает, что как минимум между одной из x_j и f существуют дуги $\overrightarrow{t}(x_j, y)$ и $\overrightarrow{t}(y, f)$, являющиеся последними дугами искомого моста.

Применим алгоритм для каждой из x_j, как показано для базы индукции, алгоритм способен правильно найти мост, состоящий их двух дуг.

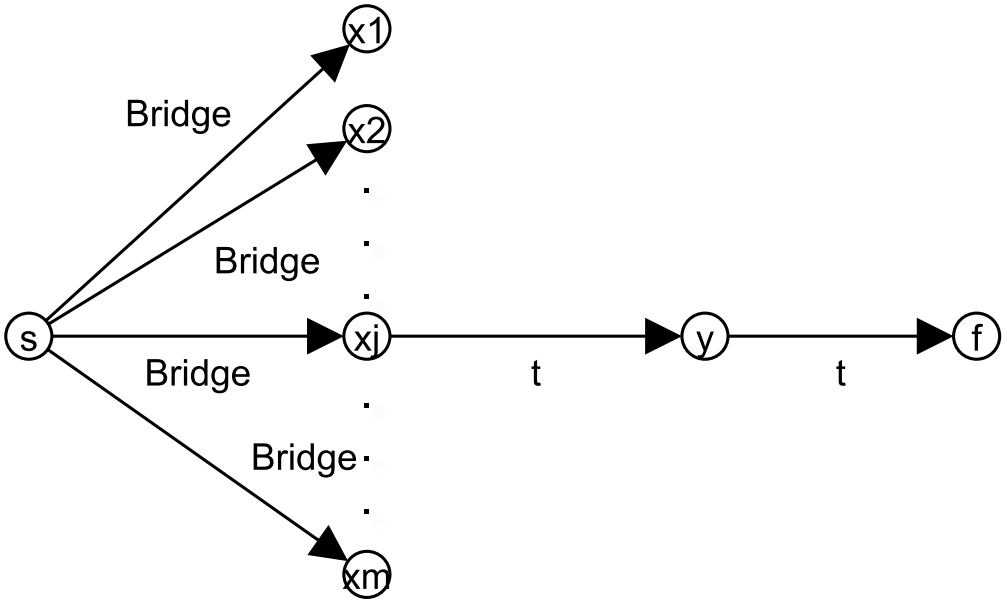

Рис. 2. Шаг индукции

■

3. Мост $\overleftarrow{t^*}$

Для моста типа $\overleftarrow{t^*}$ очевидно можно использовать описанный выше алгоритм, если на втором этапе вместо дуг типа $\overrightarrow{t^*}$ искать дуги типа $\overleftarrow{t^*}$. При этом все сказанное выше будет справедливо и для моста $\overleftarrow{t^*}$, в том числе трудоёмкость алгоритма также будет оцениваться как $O(N^3)$.

Заключение

Возможность проверки предиката *«возможен доступ»* на истинность позволяет установить наличие или отсутствие каналов утечки информации между двумя выделенными субъектами компьютерной системы. Для того чтобы

совершить проверку истинности данного предиката, необходимо также предусмотреть способы поиска мостов типа $\overrightarrow{t^*}\overrightarrow{g}\overleftarrow{t^*}$ и $\overrightarrow{t^*}\overleftarrow{g}\overleftarrow{t^*}$, а также начального и конечного пролётов моста. В данной статье способы поиска этих структур не рассматриваются. Однако разработка полиномиальных алгоритмов поиска указанных структур может лечь в основу программных средств анализа безопасности информационных систем с целью выявления каналов утечки информации.

Литература

1. Lipton R.J., Snayder L. A linear time algorithm for deciding subject security // Journal of ACM (Addison-Wesley). N.3. 1977. P.455-464
2. Теоретические основы компьютерной безопасности: Учебное пособие для вузов / Девянин П.Н. и др. М. : Радио и связь, 2001. 192 с.
3. Гайдамакин Н.А. Разграничение доступа к информации в компьютерных системах. Е. : Издательство Уральского Университета, 2003. 328 с.
4. Frank J., Bishop M. Extending the Take-Grant protection system. Technical report. Department of Computer science, University of California in Devis, 1996. 14 p.
5. Bishop M. Theft of information in the Take-Grant protection model // Computer security 3 (4). 1994. P.283–309.
6. Бречка Д.М. Алгоритмы анализа безопасности состояний компьютерной системы для модели Take-Grant // Математические структуры и моделирование. 2009. №20. С.160-172.
7. Бречка Д.М. Алгоритмы проверки безопасности состояний компьютерной системы в модели Take-Grant // Проблемы обработки и защиты информации. Книга 1. Модели политик безопасности компьютерных систем. Коллективная монография / Под общей редакцией С.В. Белима. Омск : ООО «Полиграфический центр КАН», 2010.
8. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. М. : МЦНМО, 2000. 960 с.