УДК 004.056

ПРИМЕНЕНИЕ ИЗВЕСТНЫХ АЛГОРИТМОВ ТЕОРИИ ГРАФОВ ДЛЯ ТЕСТИРОВАНИЯ ФУНКЦИЙ БЕЗОПАСНОСТИ ПРОГРАММНО-АППАРАТНЫХ СЗИ

Т.М. КАННЕР

Закрытое акционерное общество «ОКБ САПР»,
г. Москва, Россия

В настоящее время все чаще подходы к тестированию программных или программно-аппаратных комплексов, в том числе и средств защиты информации (СЗИ), предполагают построение математических моделей с использованием некоторого математического аппарата, чаще всего теории автоматов [1-4]. При этом поведение СЗИ моделируется с помощью выполнения переходов автомата и получения выходных значений. По аналогии с этими работами в одной из предыдущих работ автора [5] разработана описательная и на ее основе формальная модель произвольного программно-аппаратного СЗИ, для которой сформированы необходимые и достаточные условия принципиальной возможности проведения тестирования. В следующей работе автора [6] на основании математической модели программно-аппаратного СЗИ предложено его представление в виде автомата.

Подходы, основанные на использовании теории автоматов, в том числе и описанный автором в [6], можно использовать для обеспечения полноты тестирования программно-аппаратных средств защиты информации. Однако вопрос оптимальности тестирования при применении этих подходов остается открытым. Для обеспечения не только полноты, но и оптимальности в [6] автором сформулирована задача тестирования программно-аппаратного СЗИ, на основании которой предложен подход к проверке ее выполнимости с использованием положений теории графов.

В данной статье автором предлагается основанный на предложенном в [6] подходе алгоритм тестирования функций безопасности программно-аппаратных СЗИ, использующий известные алгоритмы теории графов, и позволяющий провести тестирование функций безопасности таких СЗИ, а также обеспечить его полноту и оптимальность.

Введем по аналогии с [6] граф, соответствующий программно-аппаратному СЗИ, представленному в виде формальной модели в [5]:

$G_m = (V, E)$ – ориентированный граф без петель и кратных дуг (простой орграф), где V – множество вершин графа, соответствующих состояниям программной или аппаратной компоненты СЗИ, а $E \subseteq V \times V$ – множество ориентированных ребер (дуг) графа – переходов СЗИ из одного состояния в другое при выполнении нецелевых функций или функций безопасности.

Задача тестирования заключается в обходе из некоторой начальной вершины $v_0 \in V$ всех дуг графа, входящих в вершины, в которых могут выполняться какие-либо функции безопасности СЗИ – $V_{\text{фб}} \subseteq V$. При этом обеспечение полноты и оптимальности тестирования основано на поиске пути, проходящего через все дуги хотя бы по одному разу за минимальное количество переходов. Таким образом, в соответствии с [6-9] задача тестирования сводится к задаче китайского почтальона (англ. Chinese postman problem), также известной как задача инспекции дорог (англ. Route Inspection Problem), либо, как частный случай – к задаче поиска Эйлерова пути (англ. Eulerian path / Eulerian trail).

На основе введенного представления программно-аппаратного СЗИ в виде графа предложен алгоритм решения задачи тестирования функций безопасности программно-аппаратного СЗИ, который заключается в выполнении следующих шагов:

1. Построить из изначального графа программно-аппаратного СЗИ G_m соответствующий граф G'_m с помощью удаления неиспользуемых при решении задачи тестирования некоторых вершин и дуг, используя приведенные в [6] правила для сохранения связности оставшихся вершин.

2. Если в G'_m есть изолированные вершины (не удаленные при построении графа G_m) – задача тестирования не может быть решена, так как невозможно будет выполнить условие полноты тестирования из [6].

3. Проверить, что в G'_m любая вершина $v \in V$ либо принадлежит орцепи, либо лежит в компоненте сильной связности, то есть имеет вид как на рисунке 1. Это можно сделать с использованием алгоритма Косараджу-Шарира за два обхода графа в глубину [7, 10, 11]: найти все компоненты связности и проверить связанность начальной вершины v_0 со всеми остальными вершинами. В противном случае задача тестирования не может быть решена, так как также невозможно будет выполнить условие полноты из [6].

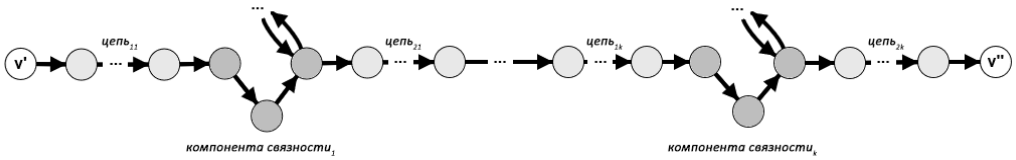

Рисунок 1 – Общий вид графа для решения задачи тестирования – отдельные вершины, цепи или компоненты связности могут отсутствовать, $k \in N_0$

4. Для всех вершин графа G'_m необходимо вычислить разницу полустепеней выхода от полустепеней входа. Если в графе есть только сбалансированные вершины (разница равна *«0»)*, то в графе существует Эйлеров цикл. Если в графе есть только две несбалансированные вершины с разницей *«1»* и *«-1»*, а остальные вершины сбалансированные, то в графе существует Эйлеров путь. В этих случаях необходимо продолжить алгоритм с пункта 9.

5. В противном случае рассмотреть отдельно несбалансированные вершины с целью нахождения путей, которые необходимо пройти повторно с сохранением требования по минимизации обхода дуг из условия оптимальности тестирования в [6]. Несбалансированные вершины можно представить в виде биграфа.

6. С помощью алгоритма поиска кратчайших путей для всех вершин в полученном биграфе – алгоритма Флойда-Уоршелла [7, 10], вычислить длину кратчайших путей от вершин с отрицательной разницей полустепеней выхода и входа к вершинам с положительной разницей.

7. Выбрать с использованием Венгерского алгоритма, алгоритма Куна-Манкреса или алгоритма Форда-Фалкерсона [7, 10] из всех сочетаний возможных кратчайших путей те пути, при повторном использовании которых все вершины станут сбалансированными, но при этом суммарная длина этих путей будет минимальна. При добавлении пути от вершины с отрицательной разницей полустепеней выхода и захода к вершине с положительной разницей, разница в обеих вершинах изменится ровно на «1» (для первой увеличится на *«1»*, для второй уменьшится на *«1»*). При этом для всех остальных вершин разница не изменится, так как могут добавиться только входящая и исходящая дуга (суммарная разница останется *«0»*).

8. Добавить дуги для выбранных на предыдущем шаге дополнительных путей в граф G'_m. Так как теперь все вершины сбалансированы, то будет существовать Эйлеров цикл.

9. С помощью алгоритма на основе циклов, также известного как алгоритм Хиерхольцера [9], построить Эйлеров цикл в полученном графе. Посещение достроенных на предыдущем шаге дуг эквивалентно повторному посещению соответствующих дуг графа G'_m. Некоторые итерации Эйлерова цикла можно менять местами, так как оптимальное решение может быть не единственным.

Блок-схема предложенного алгоритма тестирования приведена на рисунке 2.

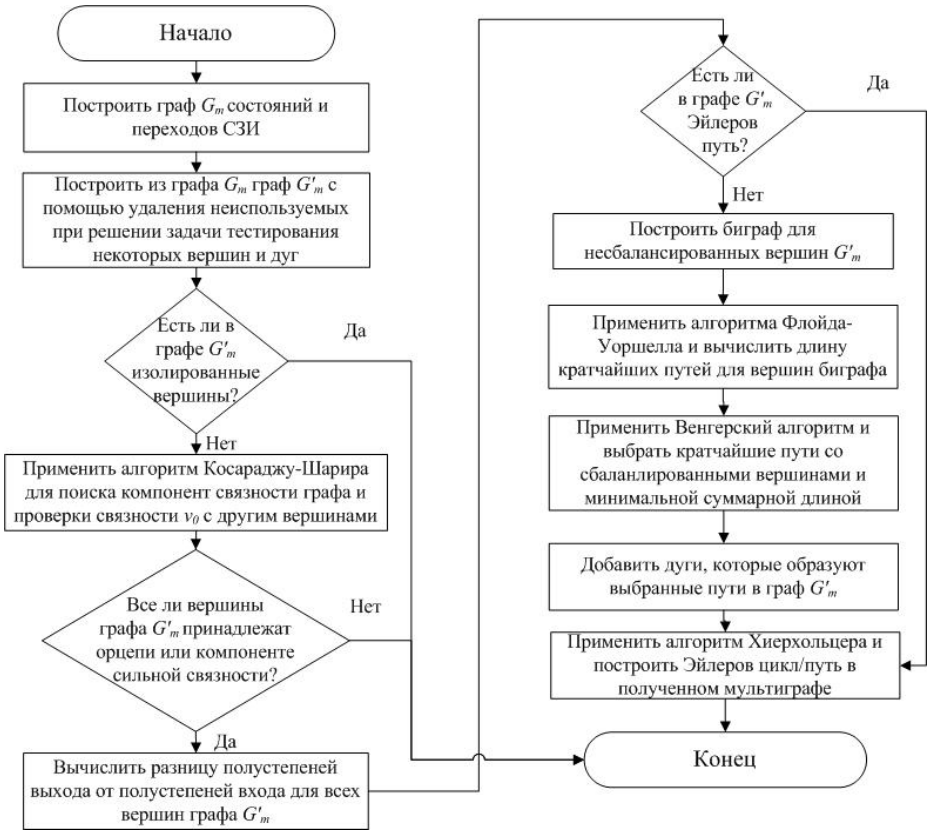

Рисунок 2 – Блок-схема алгоритма решения задачи тестирования функций безопасности программно-аппаратных СЗИ

Из предложенного алгоритма решения задачи тестирования программно-аппаратных СЗИ видно, что его шаги выполняются последовательно, и для них не используется вложенность. Это означает, что сложность представленного алгоритма равна максимальной сложности используемых в нем известных алгоритмов на графах.

Сложности используемых алгоритмов следующие [6, 7, 9]:

1. Сложность построения графа G'_m для представления графа в виде матрицы смежности $O(|V'|^3)$. За $O(|V'|^3)$ выполняется первая проверка при построении G'_m, за $O(|V'|^2)$ – вторая и третья, а четвертая проверка – за $O(|V'|^3)$. Соответственно общая сложность не будет превосходить сложности первой или четвертой проверки.
2. Алгоритм Косараджу-Шарира – $O(|V'|^2)$ для матрицы смежности;
3. Рассчет полустепеней выхода и входа и их разницы для каждой вершины – $O(|V'|^2)$ для матрицы смежности;
4. Алгоритм Флойда–Уоршелла – $O(|V'|^3)$;
5. Венгерский алгоритм – $O(|V'|^3)$;
6. Алгоритм Хиерхольцера – $O(|E'|)$.

То есть в случае если G'_m содержит Эйлеров путь (цикл) или не содержит – сложность алгоритма будет $O(|V'|^3)$ для представления графа в виде матрицы смежности. То есть задача тестирования программно-аппаратных СЗИ принадлежит классу задач P, а не NP и существует алгоритм, решающий эту задачу за полиномиальное время [9].

Предложенный алгоритм подразумевает обход вершин графа с помощью одной последовательности переходов. В случае если обход графа по приведенному алгоритму невозможно выполнить с помощью одной последовательности переходов, то это означает, что в графе есть несколько несвязанных ветвей. А это значит, что в СЗИ есть ошибка, так как нельзя добиться, чтобы все функции безопасности выполнялись корректно без

нарушения работы других функций безопасности. Поэтому в работе не рассматривается вариант с несколькими последовательными обходами графа СЗИ.

Проведена апробация предложенного алгоритма на практике – для решения задачи тестирования двух произвольно выбранных программно-аппаратных СЗИ различных видов: ШИПКА – функции безопасности которого реализованы на базе мобильной аппаратной компоненты и взаимодействуют со средой ОС средства вычислительной техники (СВТ) и СЗИ от несанкционированного доступа «Аккорд-АМДЗ» – функции безопасности которого реализованы на базе стационарной аппаратной компоненты и не взаимодействуют с ОС СВТ, выполняются независимо от ОС в составе СВТ [5]. Для данных СЗИ построены соответствующие графы, применен предложенный алгоритм и показано, что задача тестирования для данных СЗИ может быть решена, так как оба графа для них являются сильно связными. Это значит, что по предложенному алгоритму для них можно построить оптимальный путь, проходящий по всем ребрам, то есть обеспечить полноту и оптимальность тестирования.

Таким образом, в статье предложен алгоритм решения задачи тестирования произвольного программно-аппаратного СЗИ с использованием положений теории графов и известных алгоритмов на графах. В соответствии с данным алгоритмом для некоторых программно-аппаратных средств защиты эту задачу решить невозможно, так как не все вершины графа принадлежат орцепи или компоненте сильной связности, для остальных СЗИ задача имеет решение в худшем случае за полиномиальное время.

Список литературы

1. Beizer, B. Software testing techniques, 2nd ed. / Dreamtech, 2003.
2. Broy, M. Jonsson, B., Katoen, J. P., Leucker, M., Pretschner, A. Model based testing of reactive systems / LNCS 3472,Springer Berlin Heidelberg, 2005.
3. Кулямин, В. В. Тестирование на основе моделей. Курс лекции ВМиК МГУ. [Электронный ресурс] – URL: http://panda.ispras.ru/~kuliamin/mbt-course.html (дата обращения: 7 апреля 2021 г.).
4. Бурдонов, И. Б. Использование конечных автоматов для тестирования программ/ Программирование. 2000. № 2. с. 12-28.
5. Kanner, T. M. Applicability of software testing methods to software and hardware data security tools // Glob. J. Pure Appl. Math., vol. 12(1), 2016. pp. 167-190.
6. Kanner, T. M. Testing Software and Hardware Data Security Tools Using the Automata Theory and the Graph Theory / A. M. Kanner, T. M. Kanner // Proceedings of Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology, 2020. pp. 615-618.
7. Седжвик, Р. Фундаментальные алгоритмы на С. Часть 5: Алгоритмы на графах 3-е изд. / СПб.: ДиаСофтЮП, 2003. 496 с.
8. Edmonds, J., Johnson, E. L. Matching Euler tours and the chinese postman // Mathematical programming, vol. 5(1), 1973. pp. 88-124.
9. Скиена, С. С. Алгоритмы. Руководство по разработке [пер. с англ.] 2 изд. / С-П.: БХВ-Петербург, 2018.
10. Кормен, Т. Алгоритмы: построение и анализ / Т. Кормен [и др.]. // 3-е изд. М.: Вильямс, 2013. 1328 с.
11. Sharir, M. A strong connectivity algorithm and its applications to data flow analysis // Computers and Mathematics with Applications, vol. 7(1), 1981, pp. 67-72.