УДК 004.056
DOI: 10.52190/2073-2600_2021_3_12

Алгоритм тестирования функций безопасности программно-аппаратных СЗИ, основанный на использовании теории графов

Т. М. Каннер
ЗАО "ОКБ САПР", Москва, Россия

Рассмотрены существующие подходы к тестированию программно-аппаратных средств защиты информации (СЗИ), позволяющие обеспечить полноту тестирования, но не решающие вопрос его оптимальности. Дано определение задачи тестирования программно-аппаратных СЗИ. Предложен подход к проверке ее выполнимости с использованием теории графов. Представлен основанный на данном подходе алгоритм тестирования функций безопасности программно-аппаратных СЗИ, позволяющий обеспечить его полноту и оптимальность.

Ключевые слова: тестирование программно-аппаратных СЗИ, полнота и оптимальность тестирования, ориентированный граф без петель и кратных дуг, алгоритм тестирования функций безопасности программно-аппаратных СЗИ, задача китайского почтальона, эйлеров путь.

При разработке программно-аппаратных средств защиты информации, как и любых других программных или программно-аппаратных средств, должно быть проведено их тестирование в целях подтверждения соответствия реализованных функциональных возможностей заявленным характеристикам.

Существующие подходы к тестированию программных и программно-аппаратных средств предполагают построение математических моделей с применением какого-либо математического аппарата. Во многих случаях (см., например [1—4]) в качестве такого аппарата используют теорию автоматов и поведение СЗИ моделируется при помощи выполнения переходов автомата и получения его выходных значений. В работах [5, 6] автор также выбрал подход к тестированию программно-аппаратных СЗИ, основанный на построении его математической модели. В [5] сформулирована описательная модель, и на ее основе предложена формальная модель произвольного программно-аппаратного СЗИ. В [6] на основании полученной математической модели предложено представление программно-аппаратного СЗИ в виде конечного детерминированного автомата:

$\tilde{m} = (V, I, O, f, g)$ — конечный детерминированный автомат, моделирующий $m \in M$,

где M — множество всех программно-аппаратных СЗИ;

m — произвольное программно-аппаратное СЗИ;

V — множество состояний автомата, v_0 — начальное состояние;

I — множество входов (стимулов) — функций m, которые могут выполняться в $v \in V$;

$O = \{1, 0\}$ — множество выходов (реакций) — результатов выполнения стимулов в $v \in V$ (успешное или неуспешное выполнение);

$f : I \times V \to V$ — функция переходов: если $f((i, v)) = v'$, то по стимулу $i \in I$ из состояния $v \in V$ автомат переходит в состояние $v' \in V$;

$g : I \times V \to O$ — функция выходов: если $g((i, v)) = o$, то по стимулу $i \in I$ из состояния $v \in V$ на выход автомата поступает $o \in O$.

Предложенный подход, как и другие подходы, основанные на использовании теории автоматов, позволяет обеспечить полноту тестирования программно-аппаратных средств защиты информации. Однако существенным недостатком этих подходов является отсутствие ответа на вопрос об оптимальности выполняемого тестирования. Для обеспечения не только полноты, но и оптимальности в [6] автором введены следующие понятия и сформулирована задача тестирования программно-аппаратного СЗИ:

$V_{ФБ}$ — множество состояний автомата с потенциально вычислимыми ФБ — состояний, в которых необходимо проверить функции безопасности m, $V_{ФБ} \subseteq V$;

T — множество всевозможных переходов автомата; при каждом переходе (v, i, o, v') выполняется $f((i, v)) = v'$ и $g((i, v)) = o$, $T \subseteq V \times I \times O \times V$;

Каннер Татьяна Михайловна, руководитель учебного центра.
E-mail: tatianash@okbsapr.ru

Статья поступила в редакцию 1 июля 2021 г.

$\bar{s}$ — последовательность переходов тестирования $s_0, \ldots, s_{l-1} \in T$ длины $\text{len}(\bar{s}) = l \in \mathbb{N}$;

S — множество последовательностей переходов тестирования различной длины.

Для моделируемого автоматом $\tilde{m} = (V, I, O, f, g)$ программно-аппаратного СЗИ m решена задача тестирования с помощью последовательности переходов тестирования $\bar{s} \in S$, когда одновременно выполняются условия:

- возможности проведения тестирования: $m \in M_p$ (где M_p — множество программно-аппаратных СЗИ, для которых вне зависимости от человеческого фактора возможно выполнение ручного тестирования, $M_p \subseteq M$);
- полноты тестирования: $\forall s_j \in T$, для которого $v'_{j+1} \in V_{\text{ФБ}}$ является элементом последовательности $\bar{s}$;
- оптимальности тестирования: $\text{len}(\bar{s}) = \min\{\text{len}(\bar{s}') : \forall \bar{s}' \in S$ выполняется предыдущий пункт$\}$.

В данной работе представлен 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.

Алгоритм тестирования функций безопасности программно-аппаратных СЗИ

По аналогии с [6] введем граф:

$G_m = (V, E)$ — ориентированный граф без петель и кратных дуг (простой орграф), соответствующий программно-аппаратному СЗИ, представленному в виде формальной модели в [5], где:

V — множество вершин графа, соответствующих состояниям программной или аппаратной компоненты СЗИ;

$E \subseteq V \times V$ — множество ориентированных ребер (дуг) графа — переходов СЗИ из одного состояния в другое при выполнении нецелевых функций или функций безопасности.

Из условия полноты в задаче тестирования следует, что из некоторой начальной вершины $v_0 \in V$ должен осуществляться обход только тех дуг графа, входящих в вершины, в которых могут выполняться какие-либо функции безопасности СЗИ: $V_{\text{ФБ}} \subseteq V$, а не по всему множеству V. Поэтому вместо графа G_m будем рассматривать производный от него граф G'_m, который строится с помощью удаления из оригинального графа неиспользуемых при решении задачи тестирования вершин и дуг. Удаление таких вершин и дуг выполняется с использованием приведенных в [6] правил для сохранения связности оставшихся вершин.

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 (Chinese postman problem), также известной как задача инспекции дорог (Route Inspection Problem), либо, как частный случай, — к задаче поиска эйлерова пути (Eulerian path / Eulerian trail).

В соответствии с этим на основе [6] можно предложить 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 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. Это можно сделать с использованием алгоритма Косараджу—Шарира задва 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, поскольку также невозможно будет выполнить условие полноты из [6].

4. Для всех вершин графа G'_m необходимо вычислить разницу полустепеней выхода и 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 требуется рассмотреть отдельно несбалансированные вершины в целях 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 равной 0).

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

Сложность предложенного алгоритма 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 его сложность алгоритма равна максимальной сложности используемых в нем известных алгоритмов на графах. Сложности используемых алгоритмов следующие [6, 7, 9]:

- Сложность построения графа G'_m для представления графа в виде матрицы смежности $O\left(|V'|^3\right)$. За $O\left(|V'|^3\right)$ выполняется первая проверка при построении G'_m, за $O\left(|V'|^2\right)$ — вторая и третья, четвертая — за $O\left(|V'|^3\right)$. Соответственно общая сложность не превосходит сложности первой или четвертой проверки.
- Алгоритм Косараджу—Шарира — $O\left(|V'|^2\right)$ для матрицы смежности.
- Расчет полустепеней выхода и входа и их разницы для каждой вершины — $O\left(|V'|^2\right)$ для матрицы смежности.
- Алгоритм Флойда—Уоршелла — $O\left(|V'|^3\right)$.
- Венгерский алгоритм — $O\left(|V'|^3\right)$.
- Алгоритм Хиерхольцера — $O\left(|E'|\right)$.

Таким образом, в случае, если G'_m содержит эйлеров путь (цикл) или не содержит, сложность алгоритма составляет $O\left(|V'|^3\right)$ 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, в графе имеется несколько несвязанных ветвей. Следовательно, 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.

Апробация предложенного алгоритма тестирования функций безопасности программно-аппаратных СЗИ

Для апробации данного алгоритма были произвольно выбраны два программно-аппаратных СЗИ, относящихся к различным видам [5]:

- ШИПКА, функции безопасности которого реализованы на базе мобильной аппаратной компоненты и взаимодействуют со средой ОС средства вычислительной техники (СВТ);

- 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.

Для перечисленных средств защиты информации построены соответствующие графы и применен предложенный алгоритм. Показано, что оба графа для данных СЗИ являются сильно связными, т. е. может быть построен оптимальный путь, проходящий по всем ребрам. Это значит, что с использованием предложенного алгоритма может быть получено решение задачи тестирования выбранных СЗИ. При этом будут обеспечены его полнота и оптимальность.

Заключение

Предложенный алгоритм подразумевает обход вершин графа с помощью одной последовательности переходов, если обход графа по приведенному алгоритму невозможно выполнить с помощью одной последовательности переходов, в графе имеется несколько несвязанных ветвей. Следовательно, в СЗИ есть ошибка, так как нельзя добиться, чтобы все функции безопасности выполнялись корректно без нарушения работы других функций безопасности. Поэтому в статье не рассматривается вариант с несколькими последовательными обходами графа СЗИ.

Таким образом, предложен подход к проверке выполнимости задачи тестирования произвольного программно-аппаратного СЗИ с использованием положений теории графов. В соответствии с данным подходом для некоторых программно-аппаратных средств защиты эту задачу решить невозможно (не все вершины графа принадлежат орцепи или компоненте сильной связности). Для остальных СЗИ задача имеет решение в худшем случае за полиномиальное время с использованием известных алгоритмов на графах.

В качестве развития проведенного исследования можно предложить изменение задачи тестирования путем включения в нее негативного тестирования, т. е. проверки невыполнимости функций безопасности в состояниях $V_{\text{ФБ}} \setminus V$. Подходы и алгоритмы решения такой задачи тестирования будут полностью аналогичными за исключением необходимости их применения ко всему графу G_m, а не к производному графу G'_m, полученному после исключения ненужных для изначальной задачи тестирования вершин и дуг.

Литература

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: 29.05.2021).
4. *Бурдонов И. Б., Косачев А. С., Кулямин В. В.* 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. 2016. V. 12(1). P. 167—190.
6. *Kanner A. M., Kanner T. M.* Testing Software and Hardware Data Security Tools Using the Automata Theory and the Graph Theory: Proceedings of Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology. 2020. P. 615—618.
7. *Седжвик Р.* Фундаментальные алгоритмы на С. Ч. 5: Алгоритмы на графах. Изд. 3. — СПб.: ДиаСофтЮП, 2003. — 496 с.
8. *Edmonds J., Johnson E. L.* Matching Euler tours and the chinese postman // Mathematical programming. 1973. V. 5(1). P. 88—124.
9. *Скиена С. С.* Алгоритмы. Руководство по разработке. Изд. 2. / Пер. с англ. — СПб.: БХВ-Петербург, 2018.
10. *Кормен Т. и др.* Алгоритмы: построение и анализ. Изд. 3. — М.: Вильямс, 2013. — 1328 с.
11. *Sharir M.* A strong connectivity algorithm and its applications to data flow analysis // Computers and Mathematics with Applications. 1981. V. 7(1). P. 67—72.

Algorithm for testing security functions of software and hardware data security tools based on the use of the graph theory

T. M. Kanner
JSC "OKB SAPR", Moscow, Russia

The article discusses existing approaches to testing software and hardware data security tools (DST), which allow to ensure the completeness of testing, but do not solve the issue of its optimality. The definition of the problem of testing software and hardware DST is given and an approach to checking its feasibility using graph theory is proposed. The algorithm based on this approach is considered for testing security functions of software and hardware DST, which makes it possible to ensure its completeness and optimality.

Keywords: software and hardware DST testing, completeness and optimality of testing, directed loop-free graph without multiple edges, algorithm for testing security functions of software and hardware DST, Chinese postman problem, Eulerian path.

Bibliography — 11 references. *Received July 1, 2021*