ДОВЕРЕННАЯ СРЕДА

УДК 004.056: 061.68

Применимость методов тестирования ПО к программно-аппаратным СЗИ

Т. М. Каннер
Закрытое акционерное общество «ОКБ САПР», Москва, Россия

Рассмотрена применимость методов тестирования программного обеспечения к программно-аппаратным средствам защиты информации, выявлены особенности функционирования программно-аппаратных СЗИ в зависимости от их принципа работы, сформированы критерии возможности использования существующих методов тестирования ПО для тестирования различных видов программно-аппаратных СЗИ.

Ключевые слова: тестирование, методы тестирования ПО, особенности программно-аппаратных СЗИ, критерии возможности применения методов тестирования ПО для программно-аппаратных СЗИ.

Применимость методов тестирования программного обеспечения (ПО) к программно-аппаратным средствам защиты информации (СЗИ) целесообразно рассматривать в аспекте существования разных видов тестирования. Обычно принято выделять следующие признаки для классификации видов тестирования [1, 2]:

- время проведения;
- позитивность сценариев;
- степень автоматизированности тестирования;
- объект тестирования.

С точки зрения времени проведения тестирования (альфа и бета-тестирование) и позитивности сценариев тестирования (позитивное или негативное тестирование) в целом нет существенной разницы в том, что является предметом тестирования — ПО, программное или программно-аппаратное СЗИ. В связи с этим при выполнении таких видов тестирования любые методы их проведения будут применимы как к программным, так и к программно-аппаратным продуктам (в том числе и СЗИ).

Гораздо интересней рассмотреть виды тестирования, классифицированные по таким признакам, как объект тестирования и степень автоматизированности тестирования, так как в этих случаях как раз важно, что является предметом тестирования.

С точки зрения автоматизации тестирование ПО может быть ручным, автоматизированным и полуавтоматизированным [1, 2].

Каннер Татьяна Михайловна, начальник отдела верификации и сопровождения продуктов.
Тел. 8 (499) 235-71-24.
E-mail: tatianash@okbsapr.ru

Статья поступила в редакцию 8 сентября 2014 г.

В случае ручного тестирования метод тестирования ПО состоит из следующих шагов:

- создание программы и методики испытаний (ПМИ);
- определение входных данных для ПМИ (например, в каких ОС проводится тестирование, какое прикладное ПО дополнительно установлено и т. п.);
- проведение тестирования по ПМИ, которое, в свою очередь, включает:
  - выявление некорректного поведения продукта, указывающего на наличие в нем ошибок;
  - фиксацию проявления ошибок;
  - локализацию зафиксированных проявлений ошибок (поиск других проявлений и взаимосвязей);
  - анализ локализованных проявлений ошибок;
  - фиксацию ошибок и особенностей;
- анализ полученных результатов тестирования.

Программа и методика испытаний составляется на основании требований, предъявляемых к продукту. Если ПМИ составлена грамотно, то она служит хорошим помощником для тестировщика, описывая четкую последовательность его действий. После выполнения всех пунктов ПМИ на основании полученных результатов можно сделать вывод о соответствии продукта заявленным требованиям, а также оценить найденные ошибки, их критичность и возможность выпуска ПО.

При автоматизированном тестировании используются скрипты, разработанные при помощи специальных программ автоматизации на основании разработанных ПМИ и повторяющие их проверки автоматизированным способом. В этом слу-

чае последовательность шагов точно такая же, как и при ручном тестировании ПО, за исключением того, что его проводит не человек, а автоматизированный тест, им запущенный. Анализ результатов тестирования, в том числе и полученных ошибок, все равно делает человек.

Полуавтоматизированное тестирование включает использование автоматизированных тестов, но с частичным использованием человеческого ресурса, оно используется тогда, когда не получается полностью автоматизировать весь процесс тестирования и часть действий необходимо выполнять вручную. Для тестирования ПО такие ситуации достаточно редкие и встречаются чаще всего только в том случае, когда по каким-то причинам нет возможности выполнить полную автоматизацию.

Также тестирование ПО (по объекту тестирования) можно условно разделить на следующие группы:

– функциональное (тестирование реализуемого функционала продукта, включая взаимодействие его компонент);

– нефункциональное (нагрузочное, стрессовое тестирование, тестирование удобства использования и т. п.);

– тестирование, связанное с изменениями (регрессионное, санитарное тестирование и т. п.).

Необходимо отметить, что в общем случае при проведении любых видов тестирования ПО применяются следующие обобщенные методы:

- ручное тестирование по ПМИ (как функциональных, так и нефункциональных требований);
- автоматизированное тестирование с применением скриптов автоматизации (в основном нагрузочное и стрессовое тестирование, регрессионное и т. п.);
- полуавтоматизированное в случаях, когда нет возможности полноценно использовать скрипты автоматизации.

Рассмотрим особенности, которые возникают при попытках применения указанных методов проведения тестирования ПО к программно-аппаратным СЗИ на конкретных примерах.

Программно-аппаратные СЗИ, функционирующие до старта ОС

Как правило, СЗИ, функционирующие до старта ОС, представляют собой аппаратный модуль, подключаемый в средство вычислительной техники (СВТ), который либо перехватывает на себя управление до момента старта ОС СВТ, либо с него необходимо инициировать загрузку до (вместо) ОС СВТ. Примером такого СЗИ является средство защиты информации от несанкционированного доступа (НСД) «Аккорд-АМДЗ».

Принцип работы СЗИ НСД «Аккорд-АМДЗ» следующий: контроллер перехватывает управление сразу после выполнения штатного BIOS СВТ и позволяет выполнить дальнейшую загрузку ОС только с заранее определенных постоянных носителей информации и только после прохождения ряда контрольных процедур: идентификации/аутентификации пользователя и проверки целостности технических и программных средств СВТ.

Попробуем применить существующие методы тестирования ПО к тестированию данного СЗИ.

Тестирование «Аккорд-АМДЗ», выполняемое вручную, в общем случае не отличается от ручного тестирования ПО, за исключением того, что необходимо обязательно учитывать в ПМИ некоторые особенности этого СЗИ, такие как [3]:

- сложность фиксации ошибок в функционировании СЗИ (обычно связанная с невозможностью записи большого объема информации в память СЗИ или с тем, что любое возникновение серьезных ошибок, как правило, приводит к перезагрузке СВТ без сохранения каких-либо результатов);
- невозможность в общем случае использовать режим отладки или еще каким-либо образом изменять программную составляющую СЗИ (с сохранением изменений до следующей перезагрузки);
- СЗИ может быть реализовано на базе различных контроллеров (в рассматриваемом случае — Аккорд-5MX, 5.5, 5.5e, GX, GXM, GXMH), т. е. входные данные в ПМИ необходимо дополнить различными вариантами исполнения аппаратной базы;
- существование необходимости проведения перекрестного функционального тестирования составляющих программно-аппаратного СЗИ, т. е. тестирование их взаимодействия (надежности интерфейса управления, надежности каких-либо аппаратных компонент — флеш-памяти и т. д.);
- существование необходимости проведения тестирования на разнообразном оборудовании (с различными версиями BIOS/UEFI, различными прерываниями и питанием).

При рассмотрении возможности автоматизации тестирования «Аккорд-АМДЗ» сразу же возникает проблема, связанная с тем, что скрипты автоматизации могут функционировать только в ОС с установленным ПО автоматизации (достаточно большого объема с точки зрения объема памяти «Аккорд-АМДЗ»), т. е. при наличии среды функционирования скриптов автоматизации. Кроме того, невозможно автоматизировать тестирование «Аккорд-АМДЗ» на всем разнообразии оборудова-

ния (вручную устанавливая параметры BIOS и т. п.), это нарушает идеологию проведения автоматизированных тестов, которые должны проводиться без участия человека. Поэтому автоматизированные тесты для «Аккорд-АМДЗ» в любом случае необходимо дополнять ручной проверкой работоспособности на различном оборудовании.

В соответствии с вышесказанным автоматизация тестирования «Аккорд-АМДЗ» крайне затруднена, возможна только частичная автоматизация и только при выполнении следующих условий:

Условие 1.1. Наличие достаточных условий для работы среды функционирования скриптов автоматизации (работоспособность в среде ОС СЗИ, увеличение объема памяти «Аккорд-АМДЗ», т. е. изменение схемотехники).

Условие 1.2. Существование возможности фиксации результатов автоматизированного тестирования (например, с записью на внутренний или внешний носитель информации).

Условие 1.3. Наличие скриптов автоматизации перекрестного функционального тестирования составляющих программно-аппаратного СЗИ.

Из описанного видно, что автоматизированные тесты целесообразно проводить на некотором опытном/доработанном образце аппаратной составляющей, удовлетворяющей данным критериям, что, конечно же, не гарантирует точно такой же работоспособности стандартного исполнения аппаратной составляющей.

При рассмотрении тестирования «Аккорд-АМДЗ» несколько с другой стороны, его можно разделить на две части: тестирование исполняемого кода СЗИ отдельно от аппаратной составляющей (например, в среде ОС СВТ, а не СЗИ) и тестирование конечного функционала СЗИ, доступного пользователю (т. е. исполняемого кода вместе с аппаратной составляющей).

Исполняемый код СЗИ может быть запущен на выполнение не в среде ОС аппаратного устройства (в данном случае контроллера «Аккорд-АМДЗ»), а в среде аналогичной ОС, установленной на СВТ. Для исполнения этого кода не в операционной системе, а в виртуальной машине требуется возможность "пробрасывания" аппаратного компонента в конкретной среде виртуализации, что часто бывает невозможно. Таким образом, в рамках ОС СВТ можно автоматизировать процесс проверки корректности исполнения различных модулей, провести нагрузочное и стрессовое тестирование (интерфейса взаимодействия с контроллером и т. п.), проверить корректную работу интерфейса. Но в данном случае не будет ответа на главный вопрос: как будет работать финальный продукт, ведь весь этот код должен также корректно выполняться и в среде ОС контроллера «Аккорд-АМДЗ». Таким образом, дополнительно необходимо выполнять ручное тестирование.

На основании вышесказанного можно сделать вывод, что в случае с программно-аппаратными СЗИ, функционирующими до старта ОС (на примере «Аккорд-АМДЗ»), можно использовать либо только полуавтоматизированное тестирование без внесения каких-либо доработок в конструктив самого СЗИ, либо автоматизированное тестирование, для которого потребуется изменять как программную, так и аппаратную компоненту СЗИ. Однако в обоих случаях потребуется дополнительно проводить проверки, как минимум, работоспособности СЗИ на различных СВТ.

Программно-аппаратные СЗИ, функционирующие в ОС

Рассмотрим СЗИ, функционирующие в ОС, в конструктив которых входит флеш-память (далее — СЗИ с флеш-памятью). Такие СЗИ представляют собой комплекс программного обеспечения и составного аппаратного устройства (как правило, отчуждаемого), включающего в себя:

– компоненты, непосредственно реализующие функции по защите информации (микропроцессор, внутреннее ПО, ПО в ОС и т. п.);

– прочие компоненты, не реализующие функции защиты напрямую, в данном случае — флеш-память.

Например, к средствам защиты с флеш-памятью можно отнести ПСКЗИ ШИПКА на базе ШИПКА-2.0 и служебные носители (СН) Секрет. При тестировании таких СЗИ (как ручном, так и автоматизированном) необходимо дополнительно к привычным методам тестирования ПО провести такие проверки, как:

- тестирование флеш-памяти (определение скоростных характеристик, нагрузочные тесты и т. п.);
- тестирование взаимодействия компонент, реализующих защитные функции, с флеш-памятью.

Таким образом, при тестировании СЗИ, функционирующих в ОС, в конструктив которых входит флеш-память, необходимо дополнить ПМИ приведенными выше проверками. При этом следует отметить, что, например, тестирование флеш-памяти в аспекте определения ее надежности провести вручную можно, но, как минимум, неразумно по временным затратам, так как это требует выполнения большого количества циклов атомарных операций чтения/записи различных по объему блоков данных. Для тестирования флеш-памяти целесообразно применять существующие и широкодоступные специализированные средства, вместо разработки собственных тестов [4].

При рассмотрении возможности автоматизации тестирования программно-аппаратных СЗИ с флеш-памятью возникает нюанс, связанный с необходимостью переподключения аппаратной составляющей в процессе тестирования функционала продукта, используемого конечными пользователями. Для решения такой проблемы необходимо использовать специальные USB-прерыватели, позволяющие выполнять отключение питания на шине USB (для самого СЗИ), без физического отключения устройства, т. е. применение такого прерывателя позволяет полноценно эмулировать работу реального пользователя программно-аппаратного СЗИ.

В соответствии с вышесказанным автоматизация тестирования СЗИ с флеш-памятью в общем случае возможна, но требует выполнения ряда условий, таких как:

Условие 2.1. Применение USB-прерывателя в случае необходимости переподключения СЗИ в ходе настройки или использования.

Условие 2.2. Поддержка среды функционирования скриптов автоматизации в целевой ОС.

Условие 2.3. Наличие скриптов автоматизации или прочих средств тестирования флеш-памяти и взаимодействия компонент программно-аппаратных СЗИ.

Ручное же тестирование таких СЗИ достаточно затруднено и требует, как минимум, использования сторонних программ для тестирования флеш-памяти (т. е. желательна, как минимум, частичная автоматизация).

Формализация критериев применимости методов тестирования ПО к программно-аппаратным СЗИ

Обозначим как $M = \{m_1, m_2, m_3, m_4, ...\}$ — множество всех программно-аппаратных СЗИ, где m_1, m_2, m_3, m_4, … — некоторые программно-аппаратные СЗИ (например, средства защиты, рассмотренные выше — ШИПКА и «Аккорд-АМДЗ»).

Обозначим $M_р$ как множество программно-аппаратных СЗИ, для которых возможно ручное тестирование.

$M = M_р$ для любого непротиворечивого СЗИ ($M_р \subseteq M$ по определению, т. е. все программно-аппаратные СЗИ могут быть проверены вручную).

В соответствии с этим множество M можно представить в следующем виде:

$$M_а U M_{па} \subseteq M = M_р,$$

где $M_а$ — множество программно-аппаратных СЗИ, тестирование которых можно проводить автоматизировано;

$M_{па}$ — множество программно-аппаратных СЗИ, тестирование которых возможно автоматизировать только частично.

При этом:

1. $M_а \subseteq M_р = M$, $M_{па} \subseteq M_р = M$ — подмножества множества M.
2. Подмножества $M_а$ и $M_{па}$ не пересекаются — $M_а \cap M_{па} = \varnothing$.

Множество программно-аппаратных СЗИ (M) представлено на рис. 1.

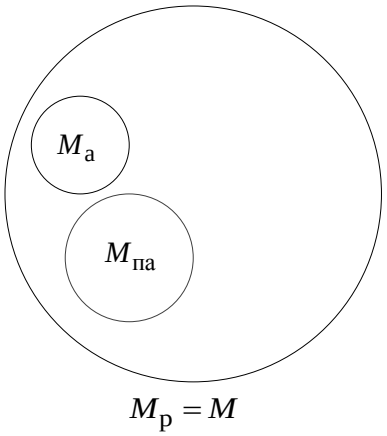

Рис. 1. Множество программно-аппаратных СЗИ (M)

Также множество M может быть представлено следующим образом:

$$M = P_{ос} U P_{доос},$$

где $P_{ос}$ — множество программно-аппаратных СЗИ, функционирующих в среде ОС;

$P_{доос}$ — множество программно-аппаратных СЗИ, функционирующих до старта ОС.

Множество $P_{ос}$ состоит из двух подмножеств:

$$P_{ос} = P_{usb} U P_{др},$$

где P_{usb} — множество программно-аппаратных СЗИ, функционирующих в среде ОС, аппаратная часть которых подключается в USB-порт СВТ;

$P_{др}$ — множество других программно-аппаратных СЗИ, функционирующих в среде ОС, не входящих в P_{usb}.

При этом $P_{usb} \cap P_{др} = \varnothing$.

Определение 1.

Определим математическую модель некоторого абстрактного программно-аппаратного СЗИ следующим образом: $\forall\ m \in M$ может быть представлено в виде $m = \sum(F, V, OP)$ — набора следующих множеств:

F — множество целевых функций, которые выполняет программно-аппаратное СЗИ (например: шифрование, подпись, контроль доступа и т. п.);

V — множество состояний, в которых может находиться аппаратная компонента (при этом $v_0 \in V$ — начальное состояние, например, когда аппаратная составляющая не инициализирована);

OP — множество функций переходов программно-аппаратного СЗИ из одного состояния в другое (например, инициализация аппаратной компоненты и ее переход в инициализированное состояние).

В каждый момент времени СЗИ может находиться в определенном состоянии (F', v, OP), где $v \in V$ — текущее состояние m, F' — множество выполнимых в данном состоянии v целевых функций m. При этом объединение $UF` = F$, $\forall v \in V$.

В общем случае $\forall F' \subseteq F, F'(\text{input}) =$ | True (если F' выполнена успешно) / False (если произошла ошибка),

где input — набор формальных параметров функции F', зависит от реализации функции (например, для шифрования файла — это файл, который нужно зашифровать, куда сохранить зашифрованный результат, и ключ шифрования).

$op \in OP: (F', v_1, OP) \rightarrow (F'', v_2, OP)$, т. е. m переходит из одного состояния (F', v_1, OP) в другое (F'', v_2, OP).

При этом из-за особенностей конкретных программно-аппаратных СЗИ некоторые функции, которые выполняет СЗИ, могут быть выполнены только в определенных состояниях, в которых находится аппаратная компонента СЗИ. Например, как правило, нельзя выполнять никакой функционал СЗИ до того, как выполнится первоначальная инициализация его аппаратной компоненты, нельзя использовать функции шифрования криптографического СЗИ, если не сгенерированы ключи шифрования и т. п.

Таким образом, *определение 1* подразумевает, что некоторые функции из множества F могут быть вычислимы в понятиях теории алгоритмов [5] только в определенных состояниях из множества V, в других состояниях данные функции могут быть невычислимы.

Также функции из множества ОР могут быть как функциями, реализованными непосредственно в самом программно-аппаратном СЗИ (нецелевые функции, например, функция инициализации устройства), так и функциями, внешними по отношению к нему, которые необходимо реализовывать дополнительно (например, ввод пин-кода, переподключение аппаратной компоненты и т. д.)

Определим множество функций ручных проверок для программно-аппаратных СЗИ.

Определение 2.

Пусть $m \in M$ и $m = \sum(F, V, OP)$ в соответствии с *определением 1*. Пусть $F = \{F_1, ..., F_n\}$, где $n \in N$ — конечное множество целевых функций, которые выполняет m.

Так как $M = M_\text{р}$, то для m существует $R = R_1 U \ldots U R_n$, где $n \in N$ — конечное множество функций ручных проверок, соответствующих функциям $F = \{F_1, ..., F_n\}$.

При этом, если представить все функции в F в виде набора выполняющихся друг за другом подфункций f (шагов, необходимых для выполнения самой функции):

$\forall F_i \subseteq F$ ($i = 1, \ldots, n$), $F_i(\text{input}) =$ | $f_1(\text{input}_1)$ / ... / $f_k(\text{input}_k)$,

при этом $\forall f_j \subseteq F_i$ ($j = 1, \ldots, k$), $f_i(\text{input}_j) =$ | True (если f_j выполнена успешно) / False (если произошла ошибка),

если f_i = False, то F_i = False;
если f_i = True, то F_i = True, если $j = k$

то в таком случае можно определить функции ручных проверок следующим образом:

В общем случае $\forall R_i \subseteq R$, R_i (input, output) = где $i = 1,...,n$. | True (если F_i выполнена успешно) / False (если произошла ошибка),

$\forall r_j \in R_i$, r_j (input_j, output_j) = где $j = 1,...,k$. | True (если f_i выполнена успешно) / False (если произошла ошибка),

где
output_i — набор выходных данных (лог работы, код ошибки, дополнительный результат и т. д.);
output = output_i, если r_i = False (R_i = False);
output = output_{i+1}, если r_i = True, $j \neq k$ (R_i = True, output = output_i, если $j = k$).

Определение 2.1.

$\forall r_j \in R_i$ вычислима, если соответствующая ей $f_j \in F_i$ вычислима для $\forall j = 1, \ldots, k$.

Определение 2.2.

Пусть $m = \sum(F, V, OP) \in M$. Тогда $m \in M_\text{р}$, если $\forall R_i \subseteq R$ (множество функций ручных проверок по *определению 2*), R_i — множество вычислимых функций, $i = 1, \ldots, n$.

Основываясь на *определениях 1, 2, 2.1* и *2.2* формализуем общий критерий возможности проведения ручного тестирования программно-аппаратных СЗИ.

Утверждение 1 (общий критерий возможности проведения ручного тестирования программно-аппаратных СЗИ).

Если $m = \sum(F,V,OP) \in M$, в состоянии $v_0 \in V$, то $m \in M_p \Leftarrow\Rightarrow$:

1. либо $\forall r_j \in R_i$ вычислимы в состоянии $v_0 \in V$ для $R_i \subseteq R$;

2. либо $\exists op_1,\ldots, op_L \in OP$ (функции перехода в состояния $v_1,\ldots, v_L$): $\forall r_j \in R_i$, невычислимые в состоянии v_0, вычислимы в одном из состояний $v_1,\ldots,v_L$ для $\forall R_i \subseteq R$.

То есть R_i(input,output)	r_1(input$_1$, output$_1$) … **op_l** **r_i(input$_j$, output$_j$)** … r_k(input$_k$, output$_k$),

где op_l — функция перехода в состояние, где r_j(input$_j$, output$_j$) вычислима.

Определим множество функций (скриптов) автоматизации для программно-аппаратных СЗИ.

Определение 3.

Пусть $m \in M$ и $m = \sum(F,V,OP)$ в соответствии с *определением 1*. Пусть $F = \{F_1, ..., F_n\}$, где $n \in N$ — конечное множество целевых функций, которые выполняет *m*.

Для m определим множество функций (скриптов) автоматизации, применимых как для ПО, так и для программно-аппаратных СЗИ, как $S = S_1 U \ldots US_n$, где $n \in N$ — конечное множество функций автоматизированных проверок, соответствующих целевым функциям $F = \{F_1, ..., F_n\}$.

При этом, если представить все функции в *F* в виде набора выполняющихся друг за другом подфункций *f* (аналогично *определению 2*), то в таком случае можно определить функции автоматизированных проверок следующим образом:

$\forall S_i \subseteq S, S_i$ (input, output, count)=	True (если F_i выполнена успешно) False (если произошла ошибка),

где i = 1,...,n; count — количество повторов выполнения автоматизированного скрипта.

S_i (input, output, count) =	for (i = 0,…, count) {
	s_1(input$_1$, output$_1$)
	…
	s_k(input$_k$, output$_k$)
	}

$\forall s_j \in S_i, s_j$(input$_j$,output$_j$) =	True (если f_j выполнена успешно) False (если произошла ошибка),

где j = 1,...,k; output$_i$ — набор выходных данных (лог работы, код ошибки, дополнительный результат и т. д.), необходимый для анализа и отчетности средства автоматизации.

output =	{ output′$_1$ … output′$_{count}$ },

где

output′$_l$ = output$_j$, l = 1,…,count, если s_j = False на итерации l;

output′$_l$ = output$_{j+1}$, l = 1,…,count, если s_j = True, $j \neq k$ на итерации l (output′$_l$ = output$_j$, если $j = k$).

Определение 3.1.

$\forall s_j \in S_i$ — вычислимая функция автоматизации, если соответствующая ей $f_j \in F_i$ вычислима, для $\forall j = 1,\ldots,k$ и существуют достаточные условия для выполнения s_j и возможность фиксации результата s_j.

Определение 3.2.

Пусть $m = \sum(F,V,OP) \in M$. Тогда $m \in M_a$, если $\forall S_i \subseteq S$ (множество функций автоматизации по *определению 3*), S_i — множество вычислимых функций автоматизации, $i = 1,\ldots,n$.

Основываясь на *определениях 1, 3, 3.1* и *3.2*, формализуем общие критерии возможности полной и частичной автоматизации тестирования программно-аппаратных СЗИ.

Утверждение 2 (общий критерий возможности автоматизации тестирования программно-аппаратных СЗИ).

Если $m = \sum(F,V,OP) \in M$ в состоянии $v_0 \in V$, то $m \in M_a \Leftarrow\Rightarrow$:

1. либо $\forall s_j \in S_i$ является вычислимой функцией автоматизации в состоянии $v_0 \in V$ для $\forall S_i \subseteq S$;

2. либо $\exists op_1,\ldots,op_L \in OP$ (автоматизированные функции перехода в состояния $v_1,\ldots,v_L$): $\forall s_j \in S_i$, невычислимые функции автоматизации в состоянии v_0, являются вычислимыми функциями автоматизации в одном из состояний $v_1,\ldots,v_L$ для $\forall S_i \subseteq S$.

То есть S_i (input, output, count) =	for (i = 0,…, count) { s_1(input$_1$, output$_1$) … **op_l** **s_j(input$_j$, output$_j$)** … s_k(input$_k$, output$_k$) } ,

где op_l — функция перехода в состояние, где s_j(input$_j$, output$_j$) вычислима.

Следствие (общий критерий возможности частичной автоматизации тестирования программно-аппаратных СЗИ):

Если $m = \sum(F,V,OP) \in M$ в состоянии $v_0 \in V$, то $m \in M_{\text{па}} \Longleftrightarrow \exists S_i \subseteq S, S_i$ — множество вычислимых функций автоматизации, $i = 1,\ldots, n \Longleftrightarrow$:

1. либо $\forall s_j \in S_i$ является вычислимой функцией автоматизации в состоянии $v_0 \in V$ для $S_i \subseteq S$;

2. либо $\exists op_1,\ldots,op_L \in OP$ (автоматизированные функции перехода в состояния $v_1,\ldots,v_L$): $\forall s_j \in S_i$, невычислимые функции автоматизации в состоянии v_0, являются вычислимыми функциями автоматизации в одном из состояний $v_1,\ldots,v_L$ для $S_i \subseteq S$.

Применим общий критерий возможности проведения ручного тестирования из *утверждения 1* к таким программно-аппаратным СЗИ, как «Аккорд-АМДЗ» и ПСКЗИ ШИПКА.

Обозначим СЗИ НСД «Аккорд-АМДЗ» как $m_{\text{амдз}}$, ПСКЗИ ШИПКА как $m_{\text{шипка}}$ ($m_{\text{амдз}}$, $m_{\text{шипка}} \in M$).

Аксиома 1. $m_{\text{амдз}} \in P_{\text{доос}}$.

Аксиома 2. $m_{\text{шипка}} \in P_{usb} \subseteq P_{\text{ос}}$.

Утверждение 3. Пусть $m_{\text{амдз}} = \sum(F_{\text{амдз}}, V_{\text{амдз}}, OP_{\text{амдз}}) \in M$ — СЗИ НСД «Аккорд-АМДЗ», а $m_{\text{шипка}} = \sum(F_{\text{шипка}}, V_{\text{шипка}}, OP_{\text{шипка}}) \in M$ — ПСКЗИ ШИПКА.

Тогда $m_{\text{амдз}} = \sum(F_{\text{амдз}}, V_{\text{амдз}}, OP_{\text{амдз}}) \in M_p$, $m_{\text{шипка}} = \sum(F_{шипка}, V_{\text{шипка}}, OP_{\text{шипка}}) \in M_p$.

Докажем *утверждение 3* для «Аккорд-АМДЗ» и ПСКЗИ ШИПКА.

«Аккорд-АМДЗ»

$m_{\text{амдз}}$ может находиться в следующих состояниях $v_0, v_1, v_2, v_3 \in V_{\text{амдз}}$:

v_0 — начальное состояние, когда контроллер неинициализирован (пустая БД пользователей, необходимо создать гл. Администратора);

v_1 — состояние, когда контроллер готов к работе (инициализирован, зарегистрирован гл. Администратор, могут быть созданы пользователи и списки КЦ);

v_2 — состояние, когда контроллер завершил работу (проведена и/а, проведены контрольные процедуры, результаты переданы в ОС);

v_3 — состояние, когда контроллер находится в технологическом режиме.

Из принципа работы «Аккорд-АМДЗ» следует, что все его целевые функции вычислимы либо в состоянии v_1, либо в состоянии v_2.

В начальный момент времени работы контроллера он не может находиться в состоянии v_2, при этом он может находится в одном из состояний — v_0, v_1, v_3. Поэтому, по *утверждению 1*, чтобы сделать вычислимой задачу ручного тестирования «Аккорд-АМДЗ» должны существовать функции перехода $op_1,\ldots,op_5 \in OP$ из одних состояний $m_{\text{амдз}}$ в другие:

- $op_1 : v_0 \to v_1$;
- $op_2 : v_1 \to v_2$;
- $op_3 : v_0 \to v_2$;
- $op_4 : v_3 \to v_1$;
- $op_5 : v_3 \to v_2$.

Возможные состояния и функции переходов $m_{\text{амдз}}$ изображены на рис. 2.

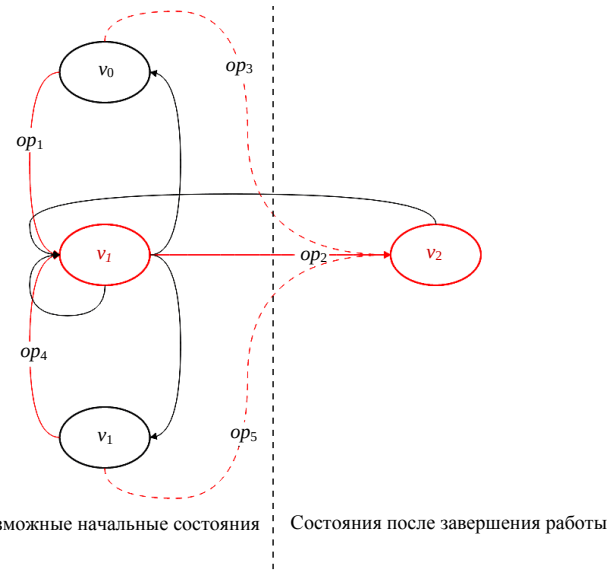

Рис. 2. Возможные состояния и функции переходов $m_{\text{амдз}}$

Для $m_{\text{амдз}}$ такими функциями перехода являются его штатные возможности, а именно:

op_1 — функция инициализации контроллера (инициализация БД пользователей, регистрация гл. Администратора);

op_2 — функция перехода при успешном прохождении процедуры и/а пользователя и контроля целостности (в случае неуспешной и/а пользователя или непрохождения КЦ контроллер переходит в состояние v_1);

op_3 — составная функция перехода, состоящая из op_1 и op_2;

op_4 — перевод контроллера из технологического режима в рабочий (ручная установка или снятие перемычки и т. п.);

op_4 — составная функция перехода, состоящая из op_4 и op_2.

Все перечисленные функции переходов могут быть выполнены вручную, в соответствии с чем все целевые функции могут быть вычислимы после выполнения соответствующих переходов при

ручном тестировании «Аккорд-АМДЗ», т. е. $m_{\text{амдз}} \in M_p$, что и требовалось доказать.

ПСКЗИ ШИПКА

$m_{\text{шипка}}$ может находиться в следующих состояниях $v_0, v_1, v_2, v_3, v_4 \in V_{\text{шипка}}$:

v_0 — начальное состояние, когда устройство неинициализировано (не форматировано, не заданы параметры авторизации, пароль администратора, PIN и PUK-код пользователя);

v_1 — состояние, когда устройство инициализировано и подготовлено к работе (отформатировано, заданы параметры авторизации, пароль администратора, PIN и PUK-код пользователя);

v_2 — состояние, когда устройство готово к работе и не введен PIN-код пользователя (сгенерированы или импортированы криптографические ключи и сертификаты, необходимые для выполнения процедур шифрования и подписи, или зарегистрирован пользователь для осуществления защищенного входа в ОС, или созданы пасскарты, или существует одновременное сочетание двух или более перечисленных пунктов);

v_3 — состояние, когда устройство готово к работе аналогично v_2, но при этом введен корректный PIN-код пользователя;

v_4 — состояние, когда устройство находится в технологическом режиме.

В начальный момент времени работы ПСКЗИ ШИПКА может находиться в состояниях v_0, v_1, v_2, v_4 и не может находится в состоянии v_3.

При этом, так как ПСКЗИ ШИПКА является отчуждаемым устройством ($m_{\text{шипка}} \in P_{usb}$), то для $m_{\text{шипка}}$ существуют состояния $v'_0, v'_1, v'_2, v'_4 \in V_{\text{шипка}}$, аналогичные состояниям $v_0, v_1, v_2, v_4 \in V_{\text{шипка}}$, но в которых устройство не подключено в USB-порт СВТ. В начальный момент времени работы ПСКЗИ ШИПКА также может находиться в состояниях v'_0, v'_1, v'_2, v'_4.

Из принципа работы ПСКЗИ ШИПКА следует, что все его целевые функции вычислимы только в состоянии v_3.

Поэтому, по *утверждению 1*, чтобы сделать вычислимой задачу ручного тестирования для ПСКЗИ ШИПКА должны существовать функции перехода $op_1, \ldots, op_5 \in OP$ из одних состояний $m_{\text{шипка}}$ в другие:

- $op_1 : v_2 \to v_3$;
- $op_2 : v'_2 \to v_3$;
- $op_3 : v_1 \to v_3$;
- $op_4 : v'_1 \to v_3$;
- $op_5 : v_0 \to v_3$;
- $op_6 : v'_0 \to v_3$;
- $op_7 : v_4 \to v_3$;
- $op_8 : v'_4 \to v_3$.

Возможные состояния и функции переходов $m_{\text{шипка}}$ изображены на рис. 3.

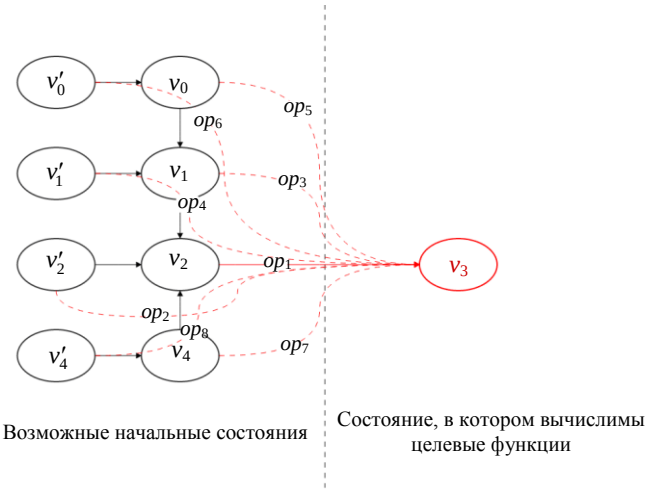

Рис. 3. Возможные состояния и функции переходов $m_{\text{шипка}}$

Для $m_{\text{шипка}}$ такими функциями перехода являются:

op_1 — ввод PIN-кода;

op_2 — составная функция перехода, состоящая из подключения устройства в USB-порт СВТ и выполнения op_1;

op_3 — составная функция перехода, состоящая из процедур генерации или импорта криптографических ключей и сертификатов, или регистрации пользователя для осуществления защищенного входа в ОС, или создания пасскарты, или одновременного сочетания двух или более перечисленных пунктов, а также из выполнения op_1;

op_4 — составная функция перехода, состоящая из подключения устройства в USB-порт СВТ и выполнения op_3;

op_5 — составная функция перехода, состоящая из инициализации устройства и выполнения op_3;

op_6 — составная функция перехода, состоящая из подключения устройства в USB-порт СВТ и выполнения op_5;

op_7 — составная функция перехода, состоящая из перевода устройства из технологического режима в состояние v_0, v_1 или v_2 и выполнения op_5, op_3 или op_1 соответственно;

op_8 – составная функция перехода, состоящая из подключения устройства в USB-порт СВТ и выполнения op_7;

Все перечисленные функции переходов могут быть выполнены вручную, в соответствии с чем все целевые функции могут быть вычислимы по-

сле выполнения соответствующих переходов при ручном тестировании ПСКЗИ ШИПКА, что и требовалось доказать.

Формализуем частные критерии возможности автоматизации тестирования «Аккорд-АМДЗ» и ПСКЗИ ШИПКА, используя *условия 1.1—1.3* и *условия 2.1—2.3* соответственно и общий критерий из *утверждения 2*.

Утверждение 4 (частный критерий возможности автоматизации тестирования «Аккорд-АМДЗ»).

Пусть $m_{\text{амдз}} = \sum(F_{\text{амдз}}, V_{\text{амдз}}, OP_{\text{амдз}}) \in M$ — СЗИ НСД «Аккорд-АМДЗ».

Тогда $m_{\text{амдз}} = \sum(F_{\text{амдз}}, V_{\text{амдз}}, OP_{\text{амдз}}) \in M_{\text{а}} \Longleftrightarrow \forall S_i \subseteq S$ (определенных в соответствии с *определением 3*) выполняются следующие условия:

1. $\forall s_j \in S_i$ являются вычислимыми функциями автоматизации в каком-либо состоянии $v \in V_{\text{амдз}}$.

2. $\exists op_4 \in OP$, позволяющая автоматизировано переходить из состояния v_3 (технологический режим) в состояние v_1 (рабочий режим), в следствии чего невычислимые в состоянии v_3 функции автоматизации $s_j \in S_i$ становятся вычислимыми функциями автоматизации в состоянии v_1 или v_2.

Из *определения 3, 3.1, 3.2* и *утверждения 3* следует, что $\forall s_j \in S_i$ должны быть вычислимыми функциями автоматизации в каком-либо состоянии $v \in V_{\text{амдз}}$.

Из *утверждения 3* следует, что все функции переходов из $OP_{\text{амдз}}$, кроме op_4, могут быть автоматизированы, в соответствии с чем, если каким-либо образом автоматизировать op_4, то для вычислимости задачи автоматизированного тестирования «Аккорд-АМДЗ» достаточно будет вызывать функции автоматизированного перехода из одного состояния (в котором некоторая s_j невычислима) в другое (в котором эта функция s_j вычислима).

Утверждение 5 (частный критерий возможности автоматизации тестирования ПСКЗИ ШИПКА).

Пусть $m_{\text{шипка}} = \sum(F_{\text{шипка}}, V_{\text{шипка}}, OP_{\text{шипка}}) \in M$ — ПСКЗИ ШИПКА.

Тогда $m_{\text{шипка}} = \sum(F_{\text{шипка}}, V_{\text{шипка}}, OP_{\text{шипка}}) \in$ $\in M_{\text{а}} \Longleftrightarrow \forall S_i \subseteq S$ (определенных в соответствии с *определением 3*) выполняются следующие условия:

1. $\forall s_j \in S_i$ являются вычислимыми функциями автоматизации в каком-либо состоянии $v \in V_{\text{шипка}}$.

2. $\exists op_2, op_4, op_6, op_8 \in OP$, позволяющие автоматизировано подключать/переподключать устройство в USB-порт СВТ (с дополнительными дальнейшими действиями), вследствие чего невычислимые функции автоматизации $s_j \in S_i$ становятся вычислимыми.

Доказательство аналогично *утверждению 4*. В качестве указанных в условиях $op_2, op_4, op_6, op_8 \in OP$ может выступать USB-прерыватель, способный подавать или прерывать питание на USB-порт СВТ. Поэтому для вычислимости задачи автоматизированного тестирования ПСКЗИ ШИПКА достаточно включать и выключать USB-прерыватель до или после определенных функций автоматизации.

Таким образом, можно сделать вывод, что и для программно-аппаратных СЗИ, функционирующих до момента страта ОС, и для программно-аппаратных СЗИ, функционирующих в ОС, применимы все те же методы тестирования программного обеспечения, но с учетом некоторых особенностей, индивидуальных для каждого СЗИ. Для того чтобы выполнять автоматизированное тестирование рассмотренных программно-аппаратных СЗИ, необходимо обязательное выполнение описанных выше критериев.

Литература

1. *Тамре Л.* Введение в тестирование программного обеспечения // Вильямс. 2003. — 368 с.
2. *Котляров В. П.* Основы тестирование программного обеспечения // Интернет-университет информационных технологий. БИНОМ. Лаборатория знаний. — М., 2006. — 285 с.
3. *Борисова Т. М., Кузнецов А. В.* Проблемы тестирования СЗИ, функционирующих до старта ОС // Комплексная защита информации. Электроника инфо. Материалы XVIII Междунар. конф. 21—24 мая 2013 года. — Брест (Республика Беларусь), 2013. С. 114—115.
4. *Борисова Т. М., Обломова А. И.* Способы автоматизации тестирования СЗИ, функционирующих в ОС, на примере ПСКЗИ ШИПКА // Комплексная защита информации. Электроника инфо. Материалы Электроника инфо. Материалы XVIII Междунар. конф. 21—24 мая 2013 года. — Брест (Республика Беларусь), 2013. С. 117—118.
5. *Мальцев А. Н.* Алгоритмы и рекурсивные функции. — М., Наука, 1986. — 366 с.

Applicability of software testing methods for software and hardware DST

T. M. Kanner
Closed Joint Stock Company "OKB SAPR", Moscow, Russia

The article considers the applicability of software testing methods for software and hardware data security tools (DST), identifies features of the software and hardware DST according to their operation principles, forms criteria possibility of using existing software testing methods for testing various types of software and hardware DST.

Keywords: testing, software testing methods, features of the software and hardware DST, criteria possibility of using existing software testing methods for testing software and hardware DST.

Bibliography — 5 references. *Received September 8, 2014*