НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ «МИФИ»

На правах рукописи

Коркин Игорь Юрьевич

МЕТОДИКА ОБНАРУЖЕНИЯ НЕЛЕГИТИМНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, ИСПОЛЬЗУЮЩЕГО ТЕХНОЛОГИЮ АППАРАТНОЙ ВИРТУАЛИЗАЦИИ

Специальность 05.13.19 – Методы и системы защиты информации, информационная безопасность

Диссертация на соискание ученой степени кандидата технических наук

Автор: 

Научный руководитель – к.т.н., доцент Петрова Т.В.

Москва – 2011

Оглавление

Введение

Актуальность темы. На современном этапе развития информационных технологий и массового внедрения средств вычислительной техники в различные сферы деятельности человека задача обеспечения информационной безопасности приобретает всё более актуальное значение.

Важным аспектом обеспечения информационной безопасности ЭВМ является обнаружение нелегитимного программного обеспечения, которое может быть потенциальным носителем программных закладок.

Появление новых процессоров с поддержкой технологии аппаратной виртуализации расширило возможности как для средств защиты информации [34], [83], [99], [128], [133], [155], так и для средств, нарушающих информационную безопасность [18], [41], [42], [78]. Программное обеспечение, использующее технологию аппаратной виртуализации (ПОАВ), работает в новом, более привилегированном, чем ОС, режиме. С одной стороны, ПОАВ, выполняющее функции монитора виртуальных машин (МВМ), повышает сервисные возможности ЭВМ и снижает стоимость её эксплуатации. Но, с другой стороны, появляется возможность для негласного внедрения образца ПОАВ – программной закладки с бесконтрольными возможностями. Бизнес-консультант по безопасности А. Лукацкий в своём докладе «Что будет актуальным в области ИБ в ближайшие годы?» на конференции «Информационная безопасность» в 2008 г. призвал обратить пристальное внимание на появление угроз информационной безопасности, связанных с технологией виртуализации в течение следующих 3-5 лет [29].

Согласно данным компании «Liliputing» [117] в течение последних пяти лет наблюдается устойчивый рост продаж процессоров фирмы Intel для настольных ЭВМ с поддержкой технологии аппаратной виртуализации с 160 до 190 миллионов единиц в год. Общее число процессоров с поддержкой технологии аппаратной виртуализации к 2012 г. по прогнозу может составить более миллиарда. Объём продаж процессоров AMD имеет схожую тенденцию.

Потенциально все ЭВМ с такими процессорами подвержены угрозам нарушения информационной безопасности, поскольку становится возможной реализация вредоносного ПОАВ для активного добывания информации и деструктивного информационного воздействия на ЭВМ различного назначения, в том числе на информационно-телекоммуникационные системы критически важных объектов [2], [118], [139].

Примечательно, что эти угрозы могут исходить от производителей как аппаратного, так и программного обеспечения. При этом внедрение ПОАВ может осуществляться посредством установки драйвера ОС, а также модификации микропрограммы BIOS и загрузочной записи жёсткого диска. Некоторые из них могут реализоваться как до ввода в эксплуатацию, так и на стадии эксплуатации ЭВМ.

Важно отметить, что штатные средства обнаружения ПОАВ отсутствуют, а опубликованные средства не позволяют принять решение о наличии ПОАВ в системе в случаях, когда ПОАВ противодействует своему обнаружению посредством компрометации показаний процессорного счётчика тактов либо временной выгрузки ПОАВ из памяти.

Обнаружением ПОАВ занимались как целые компании: Komoku, North Security Labs и др., так и отдельные специалисты: Ю. Булыгин, А. Луценко, К. Адамс (K. Adams), Э. Барбоса (E. Barbosa), Э. Филиол (E. Filiol), Х. Фритш (H. Fritsch), Н. Лоусон (N. Lawson), М. Майерс (M. Myers), Д. Медли (D. Medley) и др. Анализ опубликованных подходов и реализованных продуктов по обнаружению ПОАВ выявил такие их недостатки, как отсутствие возможности выявить ПОАВ в случае его противодействия обнаружению, а также неудобство использования и тиражирования ряда средств.

В связи с широким распространением легитимного ПО на базе технологии аппаратной виртуализации, присутствие одного образца ПОАВ уже не может рассматриваться как возможное присутствие вредоносного ПО. Особую опасность представляет нелегальный образец ПОАВ, который для своего сокрытия посредством вложенной виртуализации использует

легитимный образец ПОАВ. Методы обнаружения вложенных образцов ПОАВ отсутствуют.

Для обеспечения информационной безопасности необходимо выявить присутствие нелегитимного программного обеспечения на всех возможных архитектурных уровнях работы ЭВМ. Разработанная ранее автором в рамках дипломной работы система обнаружения скрытого программного обеспечения в 32-х битных ОС семейства Windows не позволяет выявлять скрытые объекты вне операционной системы [19].

Таким образом, обнаружение нелегитимного программного обеспечения, использующего технологию аппаратной виртуализации, является актуальной задачей для обеспечения информационной безопасности новейших ЭВМ. Поэтому разработка методики обнаружения нелегитимного ПОАВ, отвечающей современным требованиям, является актуальной и приоритетной задачей.

Целью данной работы было восполнить пробел в решении этой задачи.

Существуют инструкции, при выполнении которых в ОС управление всегда передаётся ПОАВ. По результатам измерения длительности выполнения таких инструкций, например, с помощью процессорного счётчика тактов, можно принять решение о наличии ПОАВ, поскольку под управлением ПОАВ эта длительность на порядок превышает длительность в случае его отсутствия. Однако нарушитель может использовать функциональные возможности ПОАВ по компрометации процессорного счётчика, в результате чего средние длительности выполнения набора процессорных инструкций (трассы), безусловно перехватываемых ПОАВ в случаях отсутствия и присутствия ПОАВ, совпадут. В таких ситуациях обнаружить ПОАВ не представляется возможным.

Автором предлагается методика обнаружения нелегитимного ПОАВ для ситуаций, когда противодействие выявлению со стороны ПОАВ осуществляется указанным выше способом, а также путём временной выгрузки его из памяти. Методика основана на измерении длительности выполнения

набора процессорных инструкций (трассы), безусловно перехватываемых ПОАВ.

Тема работы удовлетворяет пунктам 3, 6, 13 и 14 паспорта специальности 05.13.19 «Методы и системы защиты информации, информационная безопасность».

Объект исследования. Операционная среда ЭВМ в случаях отсутствия (присутствия) одного образца (нескольких образцов) ПОАВ.

Предмет исследования. Статистические характеристики длительности выполнения инструкций процессора, измеряемой в операционной среде с использованием процессорного счётчика тактов.

Цель диссертационной работы. Разработка методики обнаружения нелегитимного программного обеспечения, использующего технологию аппаратной виртуализации в классе процессоров Intel и AMD.

Научная задача заключается в анализе статистических характеристик длительности выполнения трассы в случаях отсутствия (присутствия) ПОАВ и в синтезе критерия присутствия ПОАВ в условиях близости моментов первого порядка длительности выполнения трассы.

В рамках решения научной задачи необходимо:

- провести сравнительный анализ и классификацию существующих способов и средств обнаружения ПОАВ;
- построить модель нарушителя, провести анализ угроз и возможных способов противодействия обнаружению ПОАВ;
- исследовать длительность выполнения трассы в случаях отсутствия (присутствия) одного образца (нескольких образцов) ПОАВ, построить модели выполнения трассы и провести их анализ;
- разработать критерий присутствия образца ПОАВ;
- разработать методику обнаружения нелегитимного ПОАВ;
- реализовать программное средство обнаружения ПОАВ.

Методы исследований. В работе используются методы теории графов, теории вероятностей и математической статистики.

Научная новизна работы состоит в следующем:

– представлены модели выполнения трассы в терминах теории графов, которые позволили выявить особенности статистических характеристик длительности выполнения трассы в случаях отсутствия (присутствия) одного образца (нескольких образцов) ПОАВ;
– синтезирован критерий присутствия образца ПОАВ на основе моментов 2-го и 4-го порядков, а также длины вариационного ряда длительности выполнения трассы;
– предложена методика обнаружения нелегитимного ПОАВ, позволяющая выявлять как один, так и несколько вложенных образцов ПОАВ.

Практическая значимость результатов заключается в следующем:

– разработанная методика позволяет обнаружить как один, так и несколько вложенных образцов ПОАВ, которые могут быть загружены из BIOS или из ОС в персональных или серверных ЭВМ;
– реализованное программное средство обнаружения ПОАВ позволяет принять решение о наличии ПОАВ в условиях его противодействия выявлению.

Результаты работы представляют практическую ценность для реализации систем защиты от вредоносного программного обеспечения.

Достоверность результатов. Достоверность результатов теоретических исследований обеспечивается корректным применением математического аппарата, а также их согласованностью с результатами экспериментальных исследований. Достоверность экспериментальных данных подкрепляется проведением опытов в полном соответствии с общепринятыми методическими рекомендациями. Допущения и ограничения в доказательствах утверждений достаточно обоснованы.

Внедрение результатов исследований. Программное средство обнаружения ПОАВ использовано в составе системы обеспечения информационной безопасности в Государственном научном центре Российской

Федерации федеральном государственном унитарном предприятии «Центральный научно-исследовательский институт химии и механики им. Д.И. Менделеева» (ГНЦ РФ ФГУП «ЦНИИХМ»).

Программное средство обнаружения было использовано как самостоятельная программа для контроля отсутствия нелегитимного ПОАВ при подготовке к вводу в эксплуатацию части парка ЭВМ федерального государственного унитарного предприятия «Научно-исследовательского института стандартизации и унификации» (ФГУП «НИИСУ»).

Теоретические и практические результаты, полученные в ходе выполнения диссертационной работы, использованы в учебном курсе «Безопасность операционных систем» кафедры «Криптология и дискретная математика» НИЯУ МИФИ для подготовки материалов лабораторных работ.

Публикации и апробация работы. Результаты диссертации изложены в 10 публикациях, 4 из которых опубликованы в рецензируемых журналах ВАК РФ. Результаты работы докладывались на конференциях и семинарах различного уровня:

- XVI, XVII, XVIII Всероссийская научно-практическая конференция «Проблемы информационной безопасности в системе высшей школы», г. Москва, 2009-2011 гг. (Инфофорум);
- XIV Международная телекоммуникационная конференция молодых ученых и студентов «Молодежь и наука», г. Москва, 2011 г.;
- XIX и XX Общероссийская научно-техническая конференция «Методы и технические средства обеспечения безопасности информации», г. Санкт-Петербург, 2010-2011 гг.;
- XV конференция «Телекоммуникации и новые информационные технологии в образовании», г. Москва, 2011 г.;
- семинары в Институте системного программирования Российской академии наук, г. Москва (21 февраля и 19 сентября 2011 года);
- семинар в Московском Государственном Техническом Университете им. Н.Э. Баумана (1 марта 2011 года);

– XIII Международная конференция «РусКрипто» в 2011 году; представленная работа вышла в финал конкурса докладов.

Структура работы. Работа состоит из введения, пяти глав, заключения, списка литературы, включающего 173 наименования, и 2 приложений. Текст диссертации изложен на 140 страницах, включая 42 рисунка и 10 таблиц.

Логически, работа состоит из пяти глав. Первая глава посвящена рассмотрению возможности применения технологии аппаратной виртуализации для сокрытия программного обеспечения и анализу существующих способов обнаружения ПОАВ. В качестве примеров ПОАВ рассматриваются программные образцы «Blue Pill» и «Vitriol». Проводится классификация существующих способов обнаружения ПОАВ, а также сравнение программных и аппаратных средств, реализованных отечественными и зарубежными исследователями. Отмечаются различные программные средства обнаружения ПОАВ. Анализируются различные аппаратные средства обнаружения ПОАВ, а также экспериментальные образцы программного обеспечения, опубликованные в электронных изданиях. Показано, что для обнаружения ПОАВ возможно использование различия статистик длительности выполнения набора процессорных инструкций в случаях отсутствия (присутствия) ПОАВ. При этом предлагается использовать безусловно перехватываемые ПОАВ инструкции (БП-инструкции), например, CPUID.

Во второй главе описаны теоретические предпосылки обнаружения ПОАВ. Проводится построение модели нарушителя и анализ угроз, описываются возможные способы сокрытия ПОАВ, в том числе путём компрометации длительности выполнения инструкций. Приводятся схемы переключения между режимами процессора при выполнении инструкций и их анализ. Представляются модели выполнения трассы в терминах теории графов для случаев отсутствия (присутствия) одного образца (нескольких образцов) ПОАВ. Осуществляется анализ построенных моделей и проверка их адекватности экспериментально полученным данным, на основании чего предлагается критерий присутствия ПОАВ, основанный на сравнении

статистик длительности выполнения инструкций для случаев присутствия и отсутствия ПОАВ: длина вариационного ряда, дисперсия и моменты 4-ого порядка длительности выполнения трассы и др. Предложенный критерий уточняется и экспериментально проверяется в главе 3.

Третья глава описывает статистические исследования длительности выполнения трассы и разработку методики обнаружения нелегитимного ПОАВ. Описана методика измерения длительности выполнения трассы и организация проведения опытов. Проанализированы возможности различных подходов к обработке получаемого из опытов статистического материала. Экспериментально подтверждён вывод из анализа моделей работы процессора о том, что значения вариационных показателей длительности выполнения трассы зависят от наличия ПОАВ: в случае его присутствия значение и вариабельность длительности выполнения трассы существенно больше, чем в случае отсутствия. Уточнён ряд показателей варьирования длительности выполнения трассы для обнаружения ПОАВ. Представлены методы и алгоритмы обработки данных и расчёта идентификационных характеристик в этих условиях. Описана методика обнаружения нелегитимного ПОАВ. Решается проблема недостаточной сходимости и воспроизводимости результатов испытаний.

В четвертой главе сформулированы требования к программному средству обнаружения ПОАВ, предложена архитектура и описана его реализация. Средство обнаружения состоит из трёх подсистем: выработки пороговых значений, имитации действий нарушителя и подсистемы выявления образцов ПОАВ. В результате взаимодействия первых двух подсистем вырабатываются пороговые значения статистик для обнаружения ПОАВ. Для минимизации внесения хаотичности в характеристики распределения длительности выполнения процессорных инструкций предложен авторский образец ПОАВ, содержащий минимальный набор инструкций и обеспечивающий компрометацию счётчиков тактов. Это наиболее сложный случай для обнаружения, который может встретиться на практике.

В пятой главе приведено подтверждение практической значимости полученных результатов, заключающееся в практическом использовании разработанного программного средства обнаружения ПОАВ на различных предприятиях. Использование результатов работы в трёх проектах подтвердило её работоспособность.

1 Проблема обнаружения программного обеспечения, использующего технологию аппаратной виртуализации

В данной главе анализируется технология аппаратной виртуализации процессоров Intel и AMD с точки зрения возможности реализации скрытого ПО, рассматриваются известные экспериментальные образцы ПОАВ, анализируются существующие подходы к выявлению их присутствия в ЭВМ.

1.1 Применение технологии аппаратной виртуализации для сокрытия программного обеспечения

В данном разделе проводится систематизированный обзор технологии аппаратной виртуализации настольных и серверных ЭВМ, работающих под управлением процессоров Intel и AMD. Описывается возможность реализации ПО, использующего технологию аппаратной виртуализации, и рассматриваются существующие образцы ПОАВ.

1.1.1 Особенности работы программного обеспечения, использующего технологию аппаратной виртуализации процессоров Intel и AMD

Рассматривается ПОАВ, которое работает в новом, более привилегированном, чем ОС, режиме. Технология аппаратной виртуализации позволяет запускать несколько различных образцов ПОАВ во вложенном виде.

ПОАВ, выполняющее функции МВМ, повышает сервисные возможности ЭВМ и снижает её эксплуатационные расходы. Благодаря МВМ на одной ЭВМ может быть запущено одновременно несколько ОС в разных виртуальных машинах. МВМ контролирует совместное использование системных ресурсов, изоляцию виртуальных машин и все их ключевые функции, в результате чего создаётся иллюзия, что гостевые ОС взаимодействуют непосредственно с аппаратным обеспечением (АО) [138].

Требования, которым должна соответствовать система, находящаяся под управлением МВМ, были сформулированы более 30-ти лет назад в работе

американских исследователей Д. Попека (G. Popek) и Р. Голдберга (R. Goldberg) [138]. Технология аппаратной виртуализации, внедрённая в 2006 г. в процессоры Intel и AMD, позволила удовлетворить всем поставленным требованиям [93], [145].

Выделяют два вида виртуализации – программную и аппаратную [47]. В случае программной виртуализации МВМ представляет собой драйвер, управляющий изолированием виртуальной машины от основной ОС. Примерами таких МВМ являются продукты Xen ESX Server, VMWare Workstation, Sun VirtualBox, экспериментальный образец программного обеспечения SubVirt [162], [164]. В случае аппаратной виртуализации ПО работает на специальном привилегированном уровне ниже ОС, используя предусмотренные возможности процессора. Примеры этого типа ПО включают в себя Hyper-V, Acronis True Image 2010, а также экспериментальные образцы программного обеспечения «Blue Pill» и «Vitriol», разработанные для процессоров AMD и Intel соответственно [104], [161].

Рассмотрим особенности применения виртуализации для сокрытия вредоносного программного обеспечения (ВПО). В книге М. Дэвиса (M. Davis) и Ш. Бодмера (S. Bodmer) «Hacking exposed malware & rootkits: malware & rootkits security secrets & solutions» [80] и в работе Т. Шилдса (T. Shields) «Survey of Rootkit Technologies and Their Impact on Digital Forensics» [160] выделены следующие виды вредоносного ПО, использующего виртуализацию: Virtualization-aware malware (VAM), Virtual machine-based rootkits (VMBR), Hardware-based virtual machine (HVM) rootkits.

ВПО типа VAM изменяет свою работу при запуске в виртуальной среде, что позволяет противодействовать его исследованию в популярных средствах, таких как VMWare [164] и VirtualBox [162]. В работе Т. Гарфинкеля (T. Garfinkel) [98] приводится подробное описание обнаружения этого ВПО, и в данной работе оно не освещается.

ВПО типа VMBR основано на программной виртуализации и способно переместить основную ОС в виртуальную машину, например, с помощью

изменения последовательности загрузки ЭВМ. Примером этого является проект SubVirt, разработанный в 2006 г. учёными из университета Мичигана и исследовательского центра Microsoft [113]. Аналитик С. Байнг (S. Bing) отмечает [65], что для запуска SubVirt необходимо внести изменения в используемое ПО – VMWare или Virtual PC, а также в ОС Windows XP либо Linux. Аспекты обнаружения ВПО типа VMBR представлены в следующих работах: в статье А. Кухара «Rootkit: новые угрозы и средства борьбы с ними» [28], в статье Д. Франклина (J. Franklin), М. Люка (M. Luk), Д. МакКуна (J. McCune) «Detecting the Presence of a VMM through Side-Effect Analysis» [94]. В представляемой работе этот тип ВПО не описывается.

ВПО типа HVM основано на технологии аппаратной виртуализации. Главное его отличие от описанных выше заключается в том, что внедрение ПОАВ производится прозрачно для пользователя, без внесения каких-либо изменений в ОС и ПО.

Поскольку представляемая работа посвящена анализу и разработке способов обнаружения программного обеспечения, использующего технологию аппаратной виртуализации, опишем его главные особенности.

В результате проведённых исследований авторы публикаций [38], [53], [75], [92], [153], [124] отмечают различные способы внедрения ПОАВ, реализуемые посредством модификации микропрограммы BIOS, изменения загрузочной записи жёсткого диска (MBR), установки драйвера в ОС. Без потери общности в данной работе описан только последний вариант.

ПОАВ инсталлируется в систему в виде драйвера ОС и прозрачно перемещает существующую ОС в виртуальную машину. В результате этого ПОАВ получает возможность контролировать события, происходящие в ОС. В книге М. Девиса (M. Davis), С. Будмера (S. Bodmer), А. ЛеМастера (A. LeMaster) «Hacking exposed malware & rootkits: malware & rootkits security secrets & solutions» [80] процесс установки ПОАВ также называется – «захват системы гипервизором» (англ. hijacking the hypervisor, hyperjacking).

Впервые установка и работа ПОАВ были продемонстрированы в 2006 г. на конференции Black Hat в США [173] специалистами Д. Зови (D. Zovi) и Дж. Рутковской (J. Rutkowska). Анализ соответствующих программных образцов приведён в подразделе 1.1.2.

В случае отсутствия ПОАВ (далее ОТ) все запросы из ОС непосредственно обрабатываются аппаратным обеспечением, а в случае присутствия ПОАВ (далее ПР) определённые запросы перехватываются ПОАВ, которое взаимодействует с аппаратным обеспечением, как это показано на рис. 1 [125]. Отметим, что ПОАВ может содержать программную закладку, несущую угрозу информационной безопасности ЭВМ, на схеме это условно обозначено червём.

Для установки ПОАВ необходимо выполнить следующие основные шаги:

- установить драйвер режима ядра ОС;
- проверить и инициализировать поддержку технологии аппаратной виртуализации;
- загрузить код ПОАВ в память из драйвера;
- создать новую виртуальную машину для ОС;
- установить связь между созданной виртуальной машиной и ПОАВ;
- запустить виртуальную машину, переключив исходную ОС в гостевой режим.

Для поддержки технологии аппаратной виртуализации производители процессоров внедрили дополнения, включающие новые режимы выполнения кода, инструкции и управляющие флаги регистров. Компания AMD назвала эти дополнения – AMD-V Secure Virtual Machine (SVM, кодовое имя «Pacifica»), а Intel – Virtualization Technology extensions (VT-x, кодовое имя «Vanderpool») [43], [84], [107], [142], [154].

Опишем особенности виртуализации посредством Intel VT-x. Эти расширения добавляют два новых режима выполнения кода – привилегированный (VMX root mode, V-режим) и непривилегированный (VMX non-root mode, R-режим). ПОАВ выполняется в высоко привилегированном V-

режиме. При этом ОС, помещённая в виртуальную машину, выполняется в менее привилегированном R-режиме.

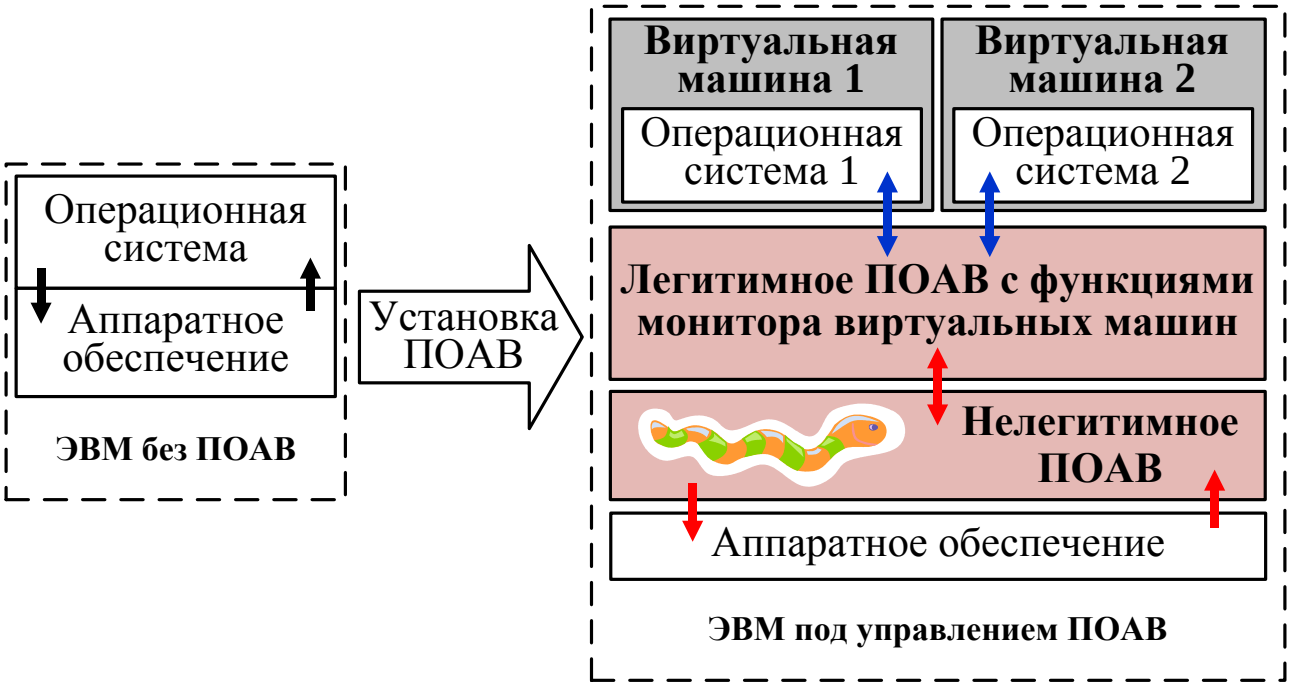


Рисунок 1 — Схема взаимодействия операционной системы с аппаратным обеспечением в случаях отсутствия (присутствия) двух образцов ПОАВ: легитимного с функциями МВМ и нелегитимного, находящихся во вложенном виде

Настройки каждой виртуальной машины сохраняются в специальной области памяти, называемой управляющая структура виртуальной машины (УСВМ, virtual-machine control structure, VMCS), доступ к которой можно получить с помощью определённых VMX-инструкций. УСВМ содержит следующие поля: область состояния виртуальной машины (Guest-State Area), область состояния ПОАВ (Host-State Area), поля управления исполнением виртуальной машины (VM-execution control fields), поля управления остановом виртуальной машины (VM-exit control fields), поля управления запуском виртуальной машины (VM-entry control fields), поля информации об остановах виртуальной машины (VM-exit information fields) [108].

Переключение из R- в V-режим называется «ВМ вход» (VM entry), который происходит при запуске или возобновлении работы виртуальной машины. Переключение из V- в R-режим называется «ВМ выход» (VM exit).

События, вызывающие «ВМ выход», будем называть демаскирующими. Список событий, выполнение которых в виртуальной машине приводит к «ВМ выходу», задаётся установкой соответствующих бит в УСВМ. В результате переключения «ВМ выход» гостевая ОС, запущенная в виртуальной машине, приостанавливает свою работу, а управление передаётся ПОАВ.

Предоставляемые возможности технологии аппаратной виртуализации позволяют разрабатывать ПОАВ для различных областей применения. Например, в ЭВМ, где не требуется работа ПО с использованием технологии аппаратной виртуализации, разработанный образец ПОАВ может предотвращать установку постороннего ПОАВ, контролируя доступ к расширениям аппаратной виртуализации процессора. ПОАВ может быть использовано в различных системах безопасности, таких как антивирусные средства, системы предотвращения вторжений и многие другие [81], [112], [117], [141], [157]. Однако все эти средства подвержены атаке «человек посередине» («Man-In-The-Middle») со стороны вредоносного ПОАВ, поскольку оно может внедриться на более ранних этапах загрузки ЭВМ и нарушить работу средств защиты [114].

1.1.2 Примеры программного обеспечения, использующего технологию аппаратной виртуализации

Рассмотрим наиболее известные примеры ПОАВ, такие как «Blue Pill», «Vitriol», которые были разработаны для демонстрации угроз информационной безопасности, исходящих со стороны технологии аппаратной виртуализации.

Проект «Blue Pill». Ярким примером ПОАВ является проект «Blue Pill», разработанный исследователями Дж. Рутковской (J. Rutkowska), А. Терешкиным (A. Tereshkin), Р. Войтчуком (R. Wojtczuk) и Р. Фаном (R. Fan) компании «Invisible Things Labs» для демонстрации возможностей технологии аппаратной виртуализации процессоров AMD-V [95]. ПОАВ «Blue Pill» было успешно опробовано на ЭВМ под управлением процессора AMD с установленной 64-разрядной ОС Windows Vista [79].

«Blue Pill» был представлен на конференции Black Hat USA в 2006 г. и с тех пор развился шире своей начальной концепции экспериментального образца. Изначально «Blue Pill» представлял собой драйвер режима ядра ОС. Последняя опубликованная на сайте [66] версия «Blue Pill» 0.32 содержит лишь неполную версию драйвера ПОАВ, разработанную для процессоров AMD. Финальная неопубликованная версия «Blue Pill» включает в себя boot-загрузчик, поддержку вложенных образцов ПОАВ и технологию «Blue Chicken», также отмечается поддержка процессоров AMD и Intel [153].

Boot-загрузчик обеспечивает запуск ПОАВ до ОС, обеспечивая более устойчивое функционирование.

Поддержка вложенных образцов ПОАВ обеспечивает прозрачную установку и работу вновь загружаемых образцов ПОАВ.

Технология «Blue chicken» обеспечивает сокрытие ПОАВ от временны́х способов обнаружения. Суть этой технологии состоит в определении того, что происходит большее число обращений к ПОАВ за сравнительно короткое время и принятии мер для предотвращения обнаружения ПОАВ. В своей статье автор [151] предлагает в качестве мер противодействия обнаружению – временную деинсталляцию ПОАВ, в результате чего длительность выполнения демаскирующих инструкций будет производиться в ЭВМ с выгруженным ПОАВ, в результате чего вне зависимости от используемого счётчика времени образец ПОАВ не будет выявлен [152], [156].

Х. Фритш (H. Fritsch) в своей работе считает важным отметить, что ПОАВ «Blue Pill» использует отдельные таблицы страниц для обеспечения сокрытия собственного кода в ОС, однако этот механизм отсутствует в опубликованной версии 0.32 [95].

Следует отметить, что публикация исходного кода «Blue Pill» вызвала широкий резонанс в мире компьютерной безопасности и явилась импульсом к созданию различных способов обнаружения ПОАВ.

Проект «Vitriol». Вторым примером ПОАВ является проект «Vitriol», разработанный исследователем Д. Зови (D. Zovi) из компании «Matasano

Security» одновременно с «Blue Pill» в 2006 г. и представленный на конференции Black Hat. «Vitriol» был успешно опробован на ЭВМ с установленной Mac OS X Tiger под управлением процессора Intel [143]. Исходные коды «Vitriol» прилагаются к книге «The Mac Hacker's Handbook» [125] и доступны на сайте издательства «John Wiley & Sons» [124].

«Blue Pill» и «Vitriol» имеют общие черты: оба ПОАВ реализованы в виде драйвера ОС и демонстрируют возможности ПО, разработанного на базе технологии аппаратной виртуализации. Однако ПОАВ «Vitriol» также осуществляет следующие деструктивные действия и действия по активному добыванию информации: сокрытие файлов и папок от пользователя, чтение и изменение сетевого трафика, регистрация всех нажатых клавиш. Следует отметить, что ПОАВ «Vitriol» не содержит таких мощных средств сокрытия, как технология «Blue Chicken», boot-загрузчик и поддержка вложенных ПОАВ, разработанных в «Blue Pill», однако осуществляет деструктивные воздействия, что демонстрирует высокую опасность ПОАВ.

1.2 Сравнительный анализ существующих способов обнаружения программного обеспечения, использующего технологию аппаратной виртуализации

В данном разделе проводится систематизированный обзор и даётся классификация описанных в литературе способов обнаружения ПОАВ. Анализируются возможности различных счётчиков ЭВМ при использовании их для измерения длительности выполнения процессорных инструкций.

1.2.1 Классификация существующих способов обнаружения ПОАВ

В первую очередь необходимо отметить, что штатные средства для обнаружения ПОАВ отсутствуют. В руководстве программиста ЗВ компании Intel об этом сказано так: «Не существует бита, доступного программному обеспечению, который показывал бы, что логический процессор находится в

режиме «VMX non-root operation» (англ. «There is no software-visible bit whose setting indicates whether a logical processor is in VMX non-root operation.») [108].

Вместе с тем имеется ряд публикаций с описанием способов обнаружения ПОАВ в ЭВМ. Для удобства анализа их особенностей предлагается следующая схема классификации (рис. 2).

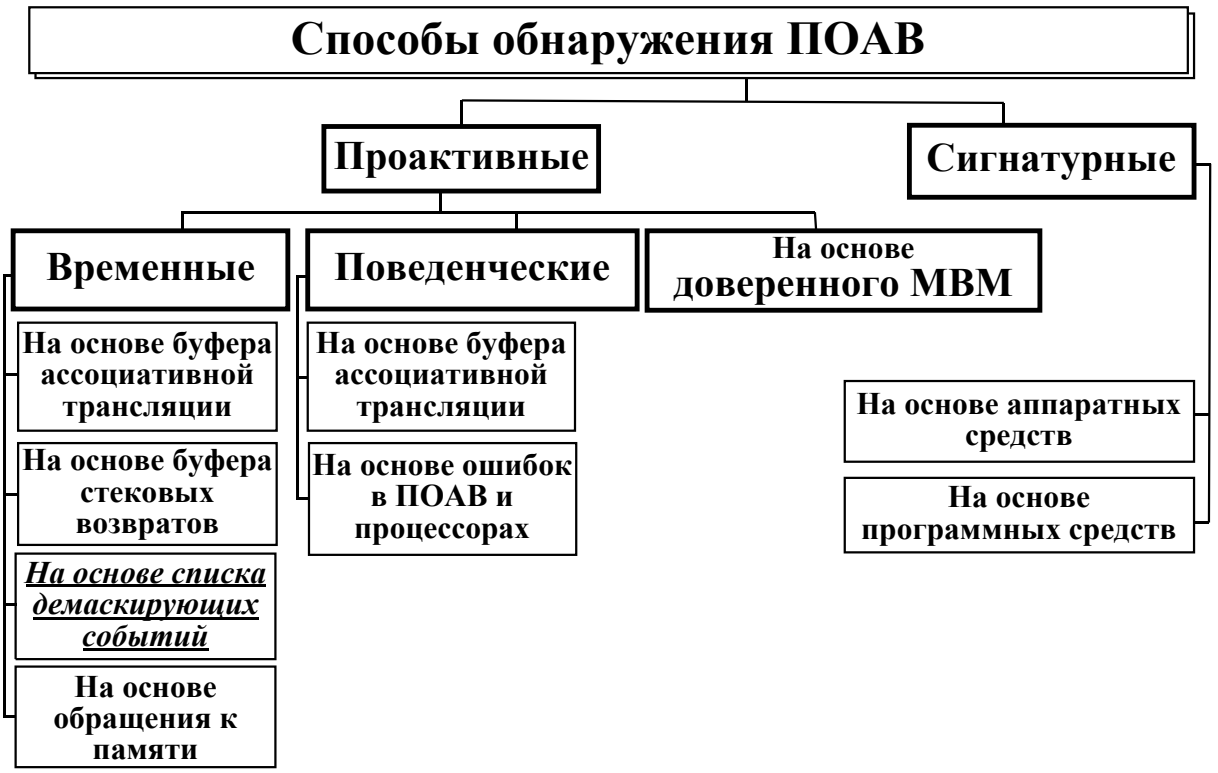


Рисунок 2 — Классификация способов обнаружения ПОАВ

Все способы можно сгруппировать в две группы: проактивные и сигнатурные. В свою очередь, группа проактивных способов подразделяется на подгруппы временны́х, поведенческих способов и с использованием доверенного МВМ. К сигнатурной группе относятся способы, использующие поиск в памяти фрагментов ПОАВ и его структур.

1.2.2 Временны́е способы обнаружения ПОАВ

Все временны́е способы обнаружения основаны на измерении длительности выполнения определённых манипуляций в гостевой ОС. В зависимости от вида манипуляций выделяют временны́е способы на основе буфера ассоциативной трансляции, буфера стековых возвратов, списка демаскирующих событий и обращения к памяти.

В случае присутствия ПОАВ выполнение манипуляций в гостевой ОС приводит к передаче управления ПОАВ, в результате чего длительность их выполнения возрастает по сравнению со случаем отсутствия ПОАВ. Поэтому имеется возможность путём сравнения наблюдаемой длительности выполнения манипуляций с эталонной решить вопрос о присутствии ПОАВ в обследуемой системе. Кроме того, как будет показано, вариабельность длительности при повторных манипуляциях различна, что также можно использовать в целях обнаружения ПОАВ.

Однако все временны́е способы имеют общий недостаток – им можно противодействовать путём искажения показаний счётчика длительности и временной выгрузкой ПОАВ из памяти [20], [22].

Способ обнаружения на основе буфера ассоциативной трансляции

Буфер ассоциативной трансляции (Translation lookaside buffer, БАТ, буфер преобразования адреса, далее «БАТ») представляет собой аппаратно реализованную в процессоре кэш-память, увеличивающую скорость работы ЭВМ за счёт ускорения преобразования адресов памяти. Использование этого буфера для обнаружения ПОАВ описано в ряде источников [58], [62], [79], [82], [91], [93], [95], [98], [131], [143], [148], [151], [152].

БАТ включает в себя записи, состоящие из ряда полей, в том числе данные о виртуальных и физических адресах страниц памяти (на рис. 3 они обозначены в блоке БАТ символами «ВА» и «ФА» соответственно).

Рисунок 3 — Схема работы страничной адресации с использованием БАТ

При каждом обращении ОС к некоторому виртуальному адресу «ВА», кэш-контроллер проверяет присутствие необходимой записи в БАТ. В случае её

обнаружения данные извлекаются из этого буфера и выполняется обращение к соответствующему физическому адресу «ФА».

В случае отсутствия записи осуществляется поиск адреса физической страницы путём длительного прохода по ряду системных таблиц – каталогам таблиц и страниц. После нахождения соответствующего физического адреса в БАТ добавляется запись о сводной информации, об исходном виртуальном и найденном физическом адресах. В результате при повторном обращении по данному виртуальному адресу произойдёт переход к физическому адресу на основе записи в БАТ.

Обращение к памяти посредством БАТ выполняется намного быстрее, чем с использованием прохода по системным таблицам. Добавление очередных записей в БАТ осуществляется по определённому алгоритму [4], [110].

При передаче управления ПОАВ происходит очистка буфера БАТ, что может быть использовано для принятия решения о наличии ПОАВ [131].

Для обнаружения ПОАВ необходимо выполнить следующие шаги:

- прочитать значения по заданному адресу памяти;
- произвести повторное обращение к ранее посещённому адресу памяти и измерить затрачиваемое время доступа;
- выполнить демаскирующее событие;
- повторно измерить время обращения к ранее посещенному адресу;
- на основе сравнения длительности операции доступа к памяти до и после выполнения демаскирующего события принять решение о присутствии ПОАВ.

При первом обращении к адресу памяти на основе прохода по системным таблицам происходит заполнение буфера БАТ. В качестве участка памяти может выступать как предварительно выделенный фрагмент, так и область памяти уже существующего объекта.

При повторном обращении к этому адресу памяти выполняется считывание записи из БАТ. В случае присутствия соответствующей записи в

БАТ проход по таблицам не осуществляется. Благодаря этому экономится время доступа к памяти.

При выполнении демаскирующего события в случае присутствия ПОАВ БАТ очищается, а при отсутствии ПОАВ состояние БАТ не изменяется.

При повторном измерении длительности обращения к ранее посещённому адресу возможны две ситуации – БАТ очищен или нет. В первом случае обращение к памяти занимает больше времени, чем во втором.

Путём сравнения длительности обращения к памяти до и после выполнения демаскирующей операции принимается решение о наличии ПОАВ: если в пределах погрешности значения совпадут, то ПОАВ отсутствует в ЭВМ, в противном случае ПОАВ присутствует.

Данный способ прост в реализации, а принятие решения о наличие ПОАВ занимает сравнительно мало времени.

К недостаткам способа следует отнести его неработоспособность на процессорах AMD и новейших процессорах Intel, которые содержат в БАТ поля «ASID» и «PCID», обеспечивающие целостность буфера при передаче управления ПОАВ. Недостатком является также сложность с переносимостью данного способа на различные процессоры, обусловленная наличием уровней и множественной ассоциативностью кеш-памяти.

Способ обнаружения на основе буфера стековых возвратов

Буфер стековых возвратов (Return Stack Buffer, RSB, стековый буфер возвратов, буфер адресов возврата, буфер стековых возвратов, далее «БСВ») предназначен для повышения производительности ЭВМ под управлением процессоров Intel путём использования сохранённых адресов возвратов из функций [50], [89].

БСВ является внутрипроцессорным стеком и, согласно работе А. Фога (A. Fog) [97], состоит из 16 записей. При вызове очередной функции её адрес возврата сохраняется в этом буфере.

Возможность использования БСВ для выявления ПОАВ в процессорах Intel описана в работе Ю. Булыгина [72], Х. Фритша (H. Fritsch) [95] и М. Атрея (M. Athreya) [62].

Способ обнаружения ПОАВ на основе БСВ основан на измерении времени возврата управления из 16-ти вложенных функций (рис. 4).

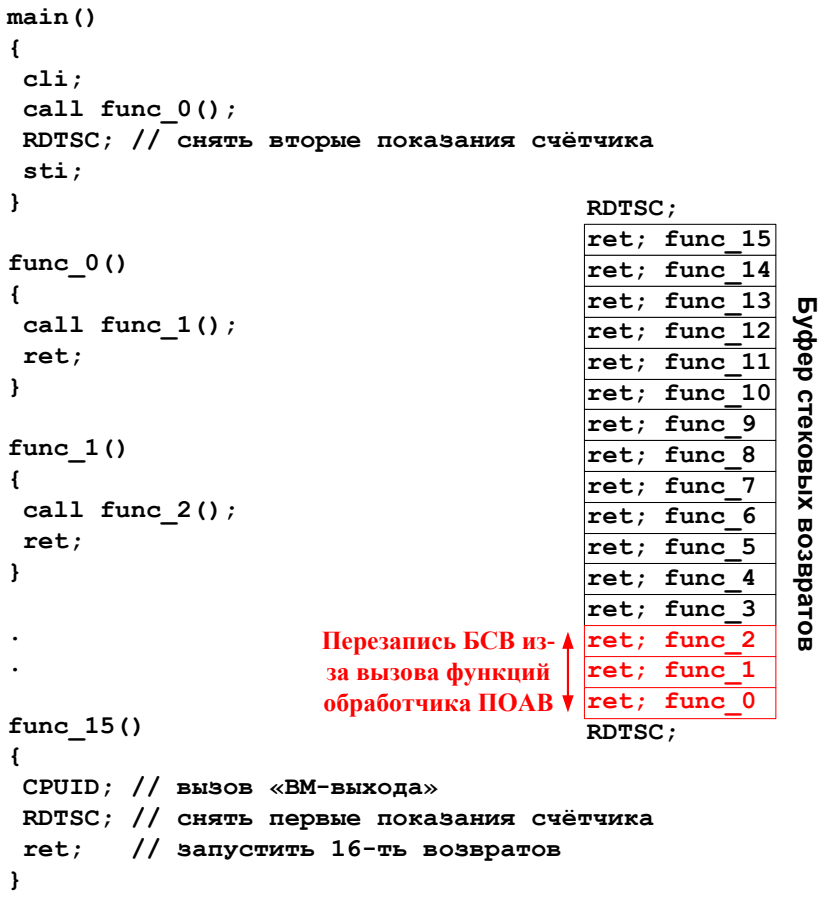


Рисунок 4 — Способ обнаружения ПОАВ на основе БСВ [62]

В работе М. Атрея (M. Athreya) предлагается следующая конструкция – организация 16-ти вложенных функций, из которых первые 15 функций вызывают только одну последующую. Последняя 16-я функция производит вызов демаскирующего события и регистрирует показания процессорного счётчика тактов TSC. После возврата управления из первой функции повторно регистрируется значение счётчика тактов.

Для обнаружения ПОАВ измеряется длительность обработки указанных выше вложенных функций, после чего путём сравнения измеренной длительности с пороговой величиной принимается соответствующее решение.

Отметим, что при выполнении 16-ой функции буфер стековых возвратов полностью заполняется адресами возврата всех вызванных функций. После вызова демаскирующего события и регистрации показаний счётчика тактов управление последовательно возвращается в каждую функцию.

При выполнении демаскирующего события в зависимости от отсутствия и присутствия ПОАВ возможны две ситуации. В первом случае БСВ не перезаписывается и содержит необходимые адреса возврата функций. Во втором случае в БСВ добавляются адреса обработчика ПОАВ и вызванных вложенных функций ПОАВ, в результате чего БСВ частично перезаписывается. В результате регистрируемая длительность выполнения функций в случае присутствия ПОАВ существенно больше, чем в случае его отсутствия.

Путём сравнения полученной длительности выполнения функций с эталонным значением принимается решение о наличии ПОАВ: если в пределах погрешности эти длительности совпадут, то ПОАВ отсутствует в ЭВМ, а в противном случае ПОАВ присутствует. Эталонное значение длительности может быть определено заранее на доверенной ЭВМ без ПОАВ.

Данный способ прост в реализации, а процедура принятия решения требует сравнительно малого времени. Однако он не защищён от противодействия. В работе [62] показано, что если обработчик ПОАВ не содержит подфункций или содержит минимально возможное их число, то обнаружить ПОАВ затруднительно.

Способ обнаружения на основе списка демаскирующих событий

Данный способ обнаружения ПОАВ рассматривался в работах [13], [62], [63], [79], [82], [93], [94], [95], [98], [101], [115], [116], [120], [122], [131], [149], [150], [152], [160], [173].

Рассмотрим в качестве манипуляций в ОС, совершаемых для обнаружения ПОАВ, выполнение демаскирующих событий. При этом необходимо выполнить следующие шаги:

– снять показание счётчика длительности;

– многократно выполнить демаскирующие события;

– повторно снять показание счётчика времени;
– подсчитать длительность выполнения демаскирующих событий;
– путём сравнения полученной величины длительности выполнения событий с эталонным значением принять решение о наличии ПОАВ. Если в пределах погрешности значения совпадут, то ПОАВ отсутствует в ЭВМ, в противном случае, оно присутствует.

М. Майерсом (M. Myers) и С. Йондтом (S. Youndt) [131] установлено, что длительность выполнения 10000 повторов инструкции CPUID в случае отсутствия ПОАВ составляет 200 тактов, а в случае присутствия ПОАВ – на два порядка больше.

Для процессоров Intel можно из множества демаскирующих событий выделить подмножество безусловно перехватываемых ПОАВ инструкций (БП-инструкций), выполнение которых всегда приводит к передаче управления ПОАВ [108]. Для процессоров AMD такие БП-инструкции отсутствуют. Однако в этом случае обнаружить ПОАВ можно путём чтения бита SVME регистра EFER. Для противодействия такому обнаружению образец ПОАВ должен перехватывать, в частности, инструкцию RDMSR, которую, фактически, можно занести в список БП-инструкций для процессоров AMD [62].

Эталонная длительность может быть заранее определена на доверенной ЭВМ без ПОАВ, либо в качестве неё может быть выбрана длительность выполнения эквивалентной инструкции или их набора, если известно, что их выполнение не перехватывается ПОАВ [82].

Способ на основе демаскирующих событий имеет следующие достоинства: он прост в реализации и переносим на различные ПОАВ и процессоры. Однако, если предприняты меры противодействия обнаружению, например, целенаправленная компрометация счётчика тактов или временная выгрузка ПОАВ из памяти, то обнаруживать ПОАВ с использованием данного способа без разработки специальных методов невозможно [23], [24].

Решению этой проблемы и посвящена данная работа.

Способ обнаружения на основе обращения к памяти

Идея этого способа предложена А. Ю. Тихоновым, ассистентом кафедры «Криптология и дискретная математика» НИЯУ МИФИ. В её основу положено различие длительности обращения к памяти в случаях присутствия и отсутствия ПОАВ.

При работе ПОАВ в памяти находится обработчик ПОАВ и связанные с ним структуры, поэтому на основе анализа копии памяти можно принять решение о наличии ПОАВ. Способ обнаружения ПОАВ на основе соответствующих сигнатур описан в подразделе 1.2.5.

ПОАВ может противодействовать получению корректной копии памяти, перехватывая соответствующие обращения к ней различными способами [93], [95] в результате обработка обращений к памяти в случае присутствия ПОАВ требует большего времени, чем в случае его отсутствия.

Для обнаружения ПОАВ необходимо выполнить следующие шаги:

– осуществить последовательные обращения по выбранным адресам памяти, измерить соответствующее время доступа и сформировать массив времени откликов для каждого адреса памяти;
– путём анализа полученного массива можно принять решение о наличии ПОАВ.

Адреса для опроса выбираются аналитиком.

Способ позволяет обнаружить не только факт присутствия ПОАВ, но и диапазон адресов, в которых находится его обработчик, поскольку при измерении времени обращения к памяти в системе будет выявлен диапазон адресов с аномально большим временем отклика. Заметим, что ПОАВ может компрометировать показания счётчика, препятствуя своему обнаружению.

Из изложенного следует, что в случаях целенаправленного искажения показаний счётчика тактов или временной выгрузки ПОАВ из памяти все рассмотренные временны́е способы не позволяют выявить ПОАВ.

Далее будет изложено, как можно преодолеть эту проблему.

1.2.3 Поведенческие способы обнаружения

В отличие от временны́х поведенческие способы обнаружения не используют измерение длительности выполнения манипуляций, а основаны на различиях в поведении ЭВМ в случаях отсутствия и присутствия ПОАВ. Можно выделить поведенческие способы обнаружения на основе буфера ассоциативной трансляции и с использованием ошибок в процессорах и ПОАВ.

Способ обнаружения на основе буфера ассоциативной трансляции

Как уже упоминалось, в процессорах Intel и AMD для ускорения преобразования виртуальных адресов в физические используется буфер ассоциативной трансляции (БАТ), в котором содержатся сведения о недавно посещённых адресах памяти.

В работах учёных [58], [63], [91], [95], [131], [152] описывается поведенческий способ обнаружения ПОАВ с использованием буфера БАТ. В его основу положено различие в адресации памяти.

Рассмотрим этапы этого способа. На первом этапе выделяется и заполняется определёнными значениями байтов некоторая область памяти, в результате в БАТ добавляются соответствующие записи об адресах памяти (рис. 5).

На втором этапе определённым образом изменяются записи в каталоге страниц и таблиц. В результате этого при трансляции виртуального адреса в физический с использованием каталогов таблиц и страниц обращение будет выполняться к новым адресам, а не к ранее выделенной области памяти.

Рисунок 5 — Заполнение БАТ при обращении к памяти

На третьем этапе выполняется демаскирующее событие, приводящее к вызову ПОАВ.

На четвёртом этапе происходит чтение области памяти, выделенной на первом этапе.

Если прочитанные значения совпадают с раннее записанными данными, то принимается решение об отсутствии ПОАВ, в противном случае – о его присутствии.

Проанализируем возможные результаты чтения памяти после выполнения демаскирующего события.

В случае присутствия ПОАВ процесс обращения к памяти сопровождается очисткой буфера памяти БАТ (рис. 6).

Рисунок 6 — Обращение к памяти в случае присутствия ПОАВ

Вследствие этого трансляция виртуального адреса в физический осуществляется с использованием каталога страниц в соответствии с изменёнными записями (надпись «правка»). В итоге обращение к ранее выделенной памяти не происходит.

В случае отсутствия ПОАВ (рис. 7) БАТ не очищается и трансляция адресов осуществляется с использованием данных в буфере БАТ, поэтому обращение происходит к ранее выделенной области памяти.

Рисунок 7 — Обращение к памяти в случае отсутствия ПОАВ

Данный способ прост в реализации, но ему присущи те же недостатки, что и аналогичному способу, описанному в подразделе 1.2.2. Поэтому он может быть применён только на сравнительно старых моделях процессоров Intel для выявления ПОАВ, не использующих сокрытие.

Способ обнаружения на основе ошибок в ПОАВ и процессорах

Этот способ основан на ошибках или неточностях на стадии разработки процессоров и ПОАВ, проявляющихся при выполнении определённых инструкций в ОС [63], [82], [131], [149], [151].

Так, в официальном отчёте компании AMD [61] указана ошибка аппаратного обеспечения «Erratum 140», заключающаяся в некорректной работе инструкции RDMSR 10h. В случае присутствия ПОАВ при чтении значения процессорного счётчика тактов TSC с помощью инструкции RDTSC учитывается значение поля TSC_OFFSET структуры УСВМ, в то время как инструкция RDMSR 10h возвращает истинное значение счётчика тактов без учёта этого поля. В случае отсутствия ПОАВ прочитанные значения с помощью описанных двух инструкций совпадают. На основе этого принимается решение о наличии ПОАВ в системе. Данная ошибка присуща некоторым устаревшим моделям процессоров AMD.

Другую ошибку аппаратного обеспечения «VMSAVE 0x67» обнаружил Э. Барбоса (E. Barbosa) [63]. Она заключается в том, что в случае присутствия ПОАВ выполнение инструкции VMSAVE с префиксом 0x67 приводит к сбою в работе ЭВМ, а в случае отсутствия ПОАВ нарушение работы не происходит.

Поскольку предложенный способ нарушает работу пользователя, использовать его в целях обнаружения ПОАВ можно только в лабораторных исследованиях.

Отдельную группу составляют способы, основанные на учёте ошибок и неточностей в обработке ПОАВ определённых инструкций ОС.

Так, Х. Фритш (H. Fritch) предложил способ обнаружения присутствия ПОАВ «Blue Pill» на основе так называемых «гипер-вызовов» («Bpknock») [95]. Было обнаружено, что если в случае присутствия этого ПОАВ в регистр EAX записать значение «0xbabecafe» и выполнить инструкцию CPUID, то в регистр EAX запишется значение «0x69696969». При выполнении описанной процедуры на ЭВМ без ПОАВ регистр EAX будет содержать нулевое значение. Путём сравнения значений регистра EAX с величиной 0x69696969 можно принять решение о наличии ПОАВ «Blue Pill» в ЭВМ. Программная реализация способа обнаружения ПОАВ, основанная на использовании этого различия, доступна на сайте [67].

В ряде случаев обнаружить присутствия ПОАВ можно путём анализа отладочных сообщений, которые может выводить ПОАВ, используя, например, функцию DbgPrint [129].

Указанные «ошибки» при разработке ПОАВ могли быть допущены случайно на стадии разработки ПО и АО, либо внесены преднамеренно.

Способы обнаружения ПОАВ на основе ошибок аппаратного и программного обеспечения применимы на некоторых устаревших моделях процессоров и версиях ПОАВ. Их адаптация к современным условиям требует глубоких и нетривиальных исследований и поиска уязвимостей.

1.2.4 Способ обнаружения на основе доверенного монитора виртуальных машин

Данный способ обнаружения используется в продуктах «HyperSafe» [166], «Hypersight Rootkit Detector» [135] и «Красная пилюля» [30].

Доверенный МВМ после своей загрузки может контролировать работу ЭВМ: перехватывать обращения к памяти и выполнение процессорных

инструкций, а также управлять запуском образцов ПОАВ, уведомляя об этом пользователя и предоставляя ему возможность выбора либо разрешить, либо запретить запуск образца ПОАВ.

Реализованные компаниями Intel и AMD технологии Trusted Execution Technology (TXT) и Secure Extension Mode (SEM) соответственно позволяют запускать доверенный ПОАВ с помощью специального аппаратного модуля платформы – Trusted Platform Module (TPM) [74], [77], [118], [134], [163]. Однако эти технологии также могут быть компрометированы: атака на Intel TXT описана в работах [42], [169], [170], идеи и примеры виртуального TPM модуля представлены в работах [105]. На кафедре «Криптология и дискретная математика» НИЯУ МИФИ Н. В. Ивановым в рамках дипломного проекта разработан также прототип виртуального модуля TPM для процессоров AMD.

К недостаткам способа доверенного МВМ следует отнести его уязвимость к атаке «человек посередине» («Man-In-The-Middle»), поскольку нелегальный ПОАВ может получить управление на более раннем этапе загрузки системы и компрометировать легитимный ПОАВ, загруженный позже [53], [68]. Применение аппаратных модулей доверенной загрузки [33] позволяет предотвратить эту атаку, однако применение данных модулей в ряде случаев может быть затруднительным или невозможным [31].

Способ на основе доверенного МВМ может быть устойчив к указанной выше атаке в случае, если при загрузке ЭВМ образец ПОАВ будет получать управление из BIOS. Однако данный способ пригоден исключительно в лабораторных условиях, поскольку его реализация не переносима на различные материнские платы, а адаптация требует проведения дополнительных исследований, что связано с трудностями.

1.2.5 Сигнатурные способы обнаружения

Под сигнатурой понимается участок памяти, характерный для случая присутствия ПОАВ: фрагменты обработчика ПОАВ и связанные с ПОАВ структуры. Сигнатурные способы основаны на поиске в дампе памяти сигнатур

ПОАВ. Этому способу посвящены публикации [62], [73], [80], [82], [95], [113], [122], [123], [131], [137], [147], [160], [168], [171].

При работе ПОАВ в физической памяти находится его обработчик и связанные с ним структуры. С помощью анализа копии памяти можно принять решение о наличии ПОАВ [122]. Например, ПОАВ «Blue Pill» при работе выделяет области памяти со следующими названиями «BLPB», «BLUE» и «BLUP». Используя эти названия и размер соответствующих областей памяти можно обнаружить присутствие ПОАВ «Blue Pill» [95].

Сигнатурные способы обнаружения можно разделить по принципу получения дампа памяти на программные и аппаратные. К программным будем относить способы, использующие штатное аппаратное обеспечение. К аппаратным – способы, использующие возможности дополнительного подключаемого аппаратного обеспечения.

Рассмотрим программные способы, использующие проход по каталогам страниц и таблиц, отображение физической памяти, анализ таблиц преобразования графических адресов и перепрограммирование контроллера памяти.

Путём прохода по каталогу таблиц и страниц можно получить копию адресного пространства [44]. Например, в случае реализации ПОАВ в виде драйвера ОС с помощью данного способа можно получить дамп адресного пространства, в системной части которого будет содержаться бинарный код драйвера образца ПОАВ вместе с его обработчиком [82]. Аналогичный дамп можно получить и на основе модификации записей таблиц страниц [62].

Другой способ получения копии образца ПОАВ состоит в последовательном проходе и отображении физической памяти, например, с использованием функции ZwMapViewOfSection [130].

Получить дамп памяти можно также, используя шины FireWire и PCI устройств [64], [69], [79], [95].

Возможность получения дампа памяти с использованием таблицы преобразования графических адресов (Graphics Address Remapping Table,

GART) описана в работе Н. Лоусона (N. Lawson), Д. Голдсмита (D. Goldsmith) и Т. Пташека (T. Ptacek) [116].

Метод «DeepWatch» для обнаружения ПОАВ с использованием встроенного контроллера северного моста разработан Ю. Булыгиным из компании Intel [62], [73].

Следует отметить, что имеется множество средств по противодействию программным способам обнаружения ПОАВ. Так, технология External Access Protection фирмы AMD позволяет скрывать области памяти ПОАВ при обращении периферийных устройств [60]. А технологии Shadow Page Tables (AMD) и Extended Page Tables (Intel), основанные на скрытых таблицах страниц, позволяют противодействовать сигнатурному обнаружению ПОАВ [152].

Способ противодействия аппаратному считыванию памяти путём изменения адреса таблицы диспетчеризации ввода/вывода памяти северного моста разработан Дж. Рутковской (J. Rutkowska) [147].

Ф. Дэвис (F. Davies) в своей работе [79] отмечает, что с появлением технологий IOMMU (input/output memory management unit) фирмы AMD и VT-d (Virtualization Technology for Directed Input/Output) фирмы Intel программные методы чтения памяти потеряют свою работоспособность.

К аппаратным относятся способы, использующие аппаратные средства для получения дампа памяти: Tribble, Copilot, RAM Capture Tool [71].

Tribble представляет собой аппаратную плату, подключаемую к ЭВМ и предназначенную для проведения криминалистических экспертиз [69], [70], [100].

Аппаратное средство Copilot, разработанное для платформы IA-32 фирмой Komoku, так же как и Tribble, основано на PCI-карте, которая может использоваться для контроля состояния памяти и файловой системы ЭВМ [68], [80]. В марте 2008 г. компания Microsoft купила фирму Komoku, и с тех пор информация о средстве Copilot не публикуется.

В 2005 г. компания BBN Technologies разработала аппаратное средство RAM Capture Tool, подключаемое к ЭВМ и позволяющее получать копию содержимого памяти [80].

Важно отметить, что существуют программные подходы, которые позволяют противодействовать получению копии памяти даже с помощью аппаратных средств [132].

Несмотря на стойкость к противодействию со стороны ПОАВ, аппаратные средства пригодны исключительно в исследовательских целях, поскольку неудобны в использовании и тиражировании.

1.2.6 Анализ возможностей компьютерных счётчиков для измерения длительности выполнения процессорных инструкций

Поскольку длительность выполнения набора процессорных инструкций являлась предметом исследования данной работы, выполнены классификация и анализ применяемых в ЭВМ счётчиков.

В зависимости от используемых ресурсов все счётчики можно разделить на две группы (рис. 8): локальные, которые встроены в ЭВМ, и сетевые – работающие на удалённых ЭВМ [11], [46].

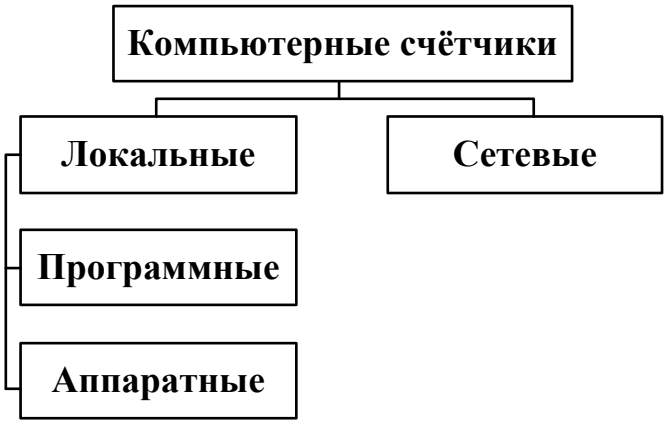


Рисунок 8 — Классификация компьютерных счётчиков

Локальные счётчики могут быть программными и аппаратными. К программным относятся счётчики, показания которых формируются в результате работы программного кода, запущенного на исследуемой ЭВМ.

Показания аппаратных счётчиков являются результатом опроса устройств, встроенных или подключённых к ЭВМ.

Значения сетевых счётчиков формируются с помощью локальных счётчиков, работающих на удалённой ЭВМ.

Идея программного счётчика предложена Э. Барбосой (E. Borbosa) [63], а исследования выполнены М. Майерсом (M. Myers), С. Йондтом (S. Youndt), А. Десносом (A. Desnos) и Э. Филиолом (E. Filiol) [82], [131].

Функционирование этого счётчика базируется на одновременном использовании двух ядер процессора ЭВМ: на первом ядре в цикле инкрементируется значение переменной-счётчика, на втором – выполняются демаскирующие события. Длительность выполнения демаскирующих событий определяется с помощью регистрации и вычитания значений переменной-счётчика до и после их выполнения.

Путём сравнения регистрируемой длительности с эталонным значением принимается решение о наличие ПОАВ.

Достоинства программного счётчика – высокая стойкость к компрометации и высокая разрешающая способность и разрядность счётчика. (Под разрешающей способностью счётчика в тактах или в долях секунды понимается минимально допустимый период следования входных сигналов, при котором счетчик работает без сбоев. Разрядность счётчика определяется максимальным числом, до которого счётчик осуществляет инкремент).

Программные счётчики применимы к процессорам с двумя и более ядрами. Отметим, что в открытых источниках пока отсутствует информация о возможности компрометации регистрируемых значений программного счётчика. Однако известно, что этот счётчик уязвим при использовании технологии временной выгрузки ПОАВ из памяти. К тому же, по мнению Х. Фритша (H. Fritsch) [95], применение описанного счётчика на практике затрудняется обеспечением синхронизации потоков.

В отличие от программных в аппаратных счётчиках используются ресурсы процессоров и периферийных устройств ЭВМ. Учёные К. Касперски,

Д. Бове (D. Bovet), исследователи компании VMware выделяют следующие аппаратные счётчики [14], [76], [165]:

– счётчик тактов центрального процессора (счётчик TSC);
– часы реального времени (счётчик RTC);
– ACPI-таймер;
– APIC-таймер;
– HPET- таймер;
– программируемый таймер интервалов (PIT-таймер);
– иные аппаратные счетчики.

Счётчик TSC (счётчик тактов центрального процессора, Time Stamp Counter) присутствует во всех процессорах с архитектурой x86. В каждом ядре процессора имеется отдельный счётчик TSC. Разрешающая способность счётчика TSC составляет 1 такт. Кроме того, счётчик TSC имеет высокую разрядность равную 2^{64}, что обеспечивает счёт тактов без переполнения в течение длительного времени [10]. Согласно работе Егорова В. Б. [11], для процессора с тактовой частотой 2 ГГц погрешность процессорного счетчика тактов составляет 0,5 нс.

Учитывая эти особенности, а также то, что процессорные инструкции RDTSC и RDMSR 10h позволяют снимать показания счётчика TSC на повышенном 31-м уровне IRQL [35], именно этот счётчик использовался в данных исследованиях.

Счётчик RTC (часы реального времени, Real Time Clock) хранит текущие хронометрические данные: время системы, день недели и т.п.) [40]. С помощью программирования этого счётчика можно генерировать периодические прерывания IRQ8 с заданной частотой [144]. Однако, К. Касперски отмечает нестабильную работу счётчика RTC, проявляющуюся в значительных биениях на временных интервалах порядка десятых долей секунды [14]. Эта нестабильность, а также зависимость точности показаний счётчика от состояния питающей батареи не позволяют использовать его для высокоточных измерений длительности выполнения процессорных инструкций.

Разработанный компаниями Hewlett-Packard, Intel, Microsoft, Phoenix, Toshiba открытый промышленный стандарт ACPI (англ. Advanced Configuration and Power Interface) определяет работу ACPI-таймера. В соответствии со спецификацией [56] ACPI-таймер включает в себя регистр Power Management Timer (PM_TMR), содержащий 32-битные значения счётчика. В статье [14] К. Касперски отмечает недостаточную точность этого счётчика (порядка 0.3 мс), а обновление счётчика происходит «рывками». Это не позволяет рассматривать ACPI-таймера как надёжное средство для измерения длительности выполнения процессорных инструкций.

Счётчик APIC-таймер имеет разрешающую способность всего 1 мс [15]. Поскольку длительность выполнения набора из 10-ти процессорных инструкций может составлять 1мкс, этот счётчик не пригоден для выполняемых исследований в данной работе.

HPET-таймер (высокочастотный таймер, High Precision Event Timer) разработан совместно компаниями Intel и Microsoft. Инициализация таймера HPET происходит при загрузке ЭВМ. Значения HPET-таймера можно получить путём обращения к определённым адресам памяти [106]. Автор статьи [167] указывает, что наличие HPET-таймера в ЭВМ является опциональным. Как отмечает К. Касперски в своей статье «Разгон и торможение Windows NT» [14], HPET-таймер пока ещё мало распространён в ЭВМ, что не позволяет выбрать этот таймер в качестве средства измерения длительности выполнения трассы.

Программируемый таймер интервалов (PIT-таймер) используется в ЭВМ уже более 30 лет [15]. Считывание его показаний осуществляется посредством взаимодействия с портами ввода\вывода, на что тратится дополнительное время. К тому же, согласно К. Касперски [14], показания счётчика нестабильны. По этим причинам PIT-счётчик не пригоден для использования в данных исследованиях.

Наряду с рассмотренными на практике используются счетчики устройств, подключаемых к ЭВМ. Эти счётчики могут либо непрерывно

инкрементировать своё значение и по запросу выдавать текущее значение, либо поддерживать команды запуска и останова счёта.

Однако счётчики этого класса сложны в тиражировании и использовании, а также требуют учёта временной задержки при чтении их показаний.

Используются также сетевые счётчики, у которых показания передаются с удалённого узла, в связи с чем существенно увеличивается время, необходимое для получения показаний счётчика. Для преодоления этого недостатка разработаны протоколы синхронизации времени NTP (Network Time Protocol) [126], Precision Time Protocol (PTP), DAYTIME и TIME, Simple Network Time Protocol (SNTP) [127] и др. Однако их разрешающая способность для измерения длительности выполнения процессорных инструкций недостаточна.

Выполненный анализ показывает, что для измерения длительности выполнения процессорных инструкций подходит процессорный счётчик тактов TSC. Он достаточно точен, имеет подходящие разрядность и разрешающую способность, прост в использовании и позволяет снимать показания из кода режима ядра, работающего на повышенном 31-м уровне IRQL [44]. Не менее важное преимущество данного счётчика в его распространённости: он присутствует на всех ЭВМ под управлением процессоров Intel и AMD.

1.3 Выводы

1. Процессоры Intel и AMD с поддержкой технологии аппаратной виртуализации получают всё большее распространение. Эти технологии поддерживают работу ПОАВ.
2. В настоящее время в открытых источниках содержатся сведения об экспериментальных образцах ПОАВ, разработанных для процессоров Intel и AMD.
3. Внедрение нелегитимного ПОАВ создаёт угрозы для информационной безопасности – представляется возможным реализация вредоносного ПО для активного добывания информации и деструктивного информационного

воздействия на ЭВМ различного назначения, в том числе на информационно-телекоммуникационные системы критически важных объектов. Ввиду этого обнаружение ПОАВ является приоритетной задачей.

4. Штатные средства обнаружения ПОАВ отсутствуют, а опубликованные обладают рядом недостатков:
   - временны́е способы подвержены противодействию путём достаточно точной компрометации счётчика длительности, а также путём временной выгрузки ПОАВ;
   - поведенческие способы на новых процессорах не работоспособны, а их адаптация требует глубоких исследований;
   - способы на основе доверенного ПОАВ уязвимы к атаке «человек посередине» («Man-In-The-Middle»);
   - аппаратные средства обнаружения неудобны в использовании и тиражировании.
5. Сравнительная оценка существующих средств для обнаружения ПОАВ приведена в табл. 1.
6. Анализ показал, что подходящими возможностями для измерения длительности выполнения процессорных инструкций обладает счётчик тактов TSC. Его разрешающая способность, широкая распространённость и возможность работать на разных приоритетах, в том числе и на повышенном 31-м уровне IRQL, превосходят характеристики всех остальных счётчиков.
7. В настоящее время отсутствуют программные средства обнаружения ПОАВ в ситуациях, когда нарушителем предпринимаются меры по противодействию обнаружению нелегитимных, в том числе и вложенных образцов ПОАВ, путём компрометации счётчика времени или временной выгрузки ПОАВ из памяти.

Таблица 1 – Сравнение существующих средств обнаружения ПОАВ

Наименование средства	**Способ обнаружения ПОАВ**	**Возможность обнаружения ПОАВ**		**Удобство использования и тиражирования**	**Обнаружение нескольких образцов ПОАВ**
		не скрытого	**скрытого**		
Hypersight Rootkit Detector (North Security Labs, 2007–2011)	На основе доверенного МВМ	+	—	+	—
«Красная пилюля» (Луценко А. 2010 г.)		+	—	+	—
DeepWatch (Булыгин Ю. 2008 г.)	Сигнатурный на основе аппаратных средств	+	+	—	—
Copilot (Komoku, 2008 г.)		+	+	—	—
Экспериментальные образцы ПО	Временные и поведенческие способы на основе буфера ассоциативной трансляции и др.	+	—	+	—

Под не скрытым образцом ПОАВ подразумевается отсутствие в этом образце компонента, обеспечивающего противодействие обнаружению. Под скрытым образцом ПОАВ подразумевается наличие в этом образце такого компонента. В таблице знаками «+» и «—» показано наличие (отсутствие) указанной характеристики. Под удобством тиражирования понимается отсутствие в средстве обнаружения внешнего аппаратного компонента, необходимого на протяжении всего времени работы.

2 Теоретические предпосылки обнаружения программного обеспечения, использующего технологию аппаратной виртуализации

В данной главе представляются теоретические предпосылки обнаружения ПОАВ, строится модель нарушителя и анализируются возможные угрозы обрабатываемой в ЭВМ информации. Рассматривается процедура определения длительности выполнения трассы. Строятся модели выполнения трассы в терминах теории графов для случаев отсутствия (присутствия) одного образца (нескольких вложенных образцов) ПОАВ. Проверяется адекватность моделей экспериментальным данным и из их анализа выявляются особенности длительности выполнения процессорных инструкций в случаях отсутствия и присутствия ПОАВ. Вводится понятие критерия различимости распределений и доказывается его существование для теоретических распределений в случае отсутствия и присутствия ПОАВ.

2.1 Модель нарушителя

В данном разделе приводится модель нарушителя, анализируются технические аспекты способов, используемых нарушителем для сокрытия ПОАВ.

2.1.1 Общие сведения

Рассматривается нарушитель, обладающий возможностью несанкционированно внедрять ПОАВ с помощью:

– установки драйвера операционной системы;
– модификации главной загрузочной записи жёсткого диска;
– внесения изменений в микропрограмму BIOS.

При этом учитывается, что реализованный нарушителем образец ПОАВ может противодействовать обнаружению посредством компрометации процессорного счётчика тактов, временной деинсталляции из памяти, а также предотвращать получение копии дампа памяти, содержащей ПОАВ.

Нарушителями могут быть производители программного и аппаратного обеспечения, соответствующие 3 и 4 уровню предоставляемых возможностей, по РД ФСТЭК «Концепция защиты средств вычислительной техники и автоматизированных систем» [37] (рис. 9).

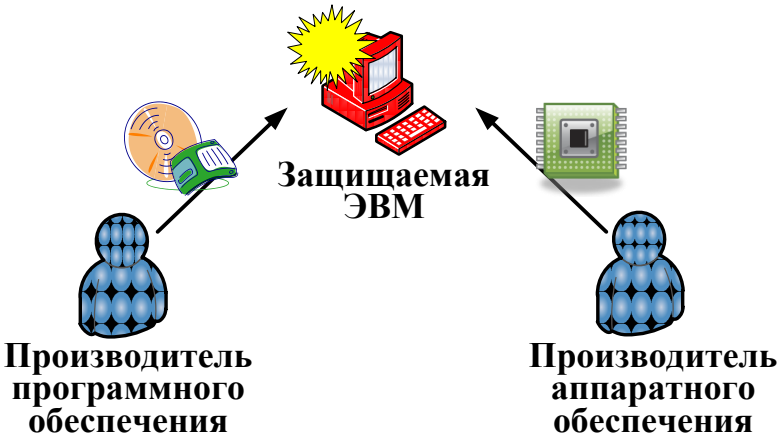


Рисунок 9 — Схема возможных нарушителей

Злонамеренный производитель аппаратного обеспечения может разрабатывать оборудование, содержащее ПОАВ. В нашем случае под аппаратным обеспечением подразумеваются материнские платы с микропрограммой BIOS, которая при загрузке системы передаёт управление ПОАВ. Злонамеренный производитель ПО может разрабатывать приложения, заведомо содержащие ПОАВ.

В результате от нарушителей исходит целый ряд различных угроз информационной безопасности: нарушения конфиденциальности, целостности и доступности информации.

Угроза нарушения конфиденциальности в ряде случаев может быть снята, если ЭВМ находится в контуре, не связанном какими-либо телекоммуникационными соединениями с сетью Интернет. При этом, однако, утечка информации возможна посредством внешних носителей [140].

Угрозы нарушения целостности и доступности информации заключаются в том, что ПОАВ может осуществлять деструктивное информационное воздействие как на данные, обрабатываемые ЭВМ, так и на ПО и АО, в том числе и на АО, подключаемое к ЭВМ, а также нарушать работу информационно-телекоммуникационных сетей и технологических процессов.

Поэтому для предотвращения угроз безопасности информации необходимо периодически проверять ЭВМ на отсутствие образцов ПОАВ, а при обнаружении подсчитывать их число.

2.1.2 Возможные способы компрометации счётчика тактов

Поскольку в данной работе разрабатывается методика обнаружения ПОАВ на основе временны́х статистик с использованием списка демаскирующих событий в условиях целенаправленной компрометации нарушителем счётчика тактов, необходимо проанализировать возможные способы такой компрометации.

Компрометация счётчика тактов может осуществляться:

– при перехвате БП-инструкций;
– при перехвате инструкций чтения счётчика тактов;
– с использованием поля TSC_OFFSET структуры УСВМ.

Альтернативным способом сокрытия образца ПОАВ является временная выгрузка его из памяти. Она описана в подразделе 1.2.2.

Рассмотрим возможности каждого из перечисленных способов.

Компрометация при перехвате БП-инструкции

Согласно изложенному в подразделе 1.2.2, для реализации данного способа каждый образец ПОАВ должен поддерживать обработку БП-инструкций.

На рис. 10 схематично представлена последовательность процедур измерения длительности выполнения трассы с учётом перехвата инструкций ПОАВ. Для простоты рассматривается трасса, состоящая из одной инструкции CPUID. Слева на рисунке 10 условно изображена ось времени, соответствующая показаниям процессорного счётчика тактов TSC. Левый прямоугольник соответствует защищённому режиму работы процессора (R-режим), в котором осуществляется измерение длительности выполнения трассы. Правый прямоугольник соответствует режиму монитора виртуальных машин (V-режим), в котором находится обработчик ПОАВ,

компрометирующий показания процессорного счётчика тактов. Стрелками показаны возможные переключения между R- и V-режимами.

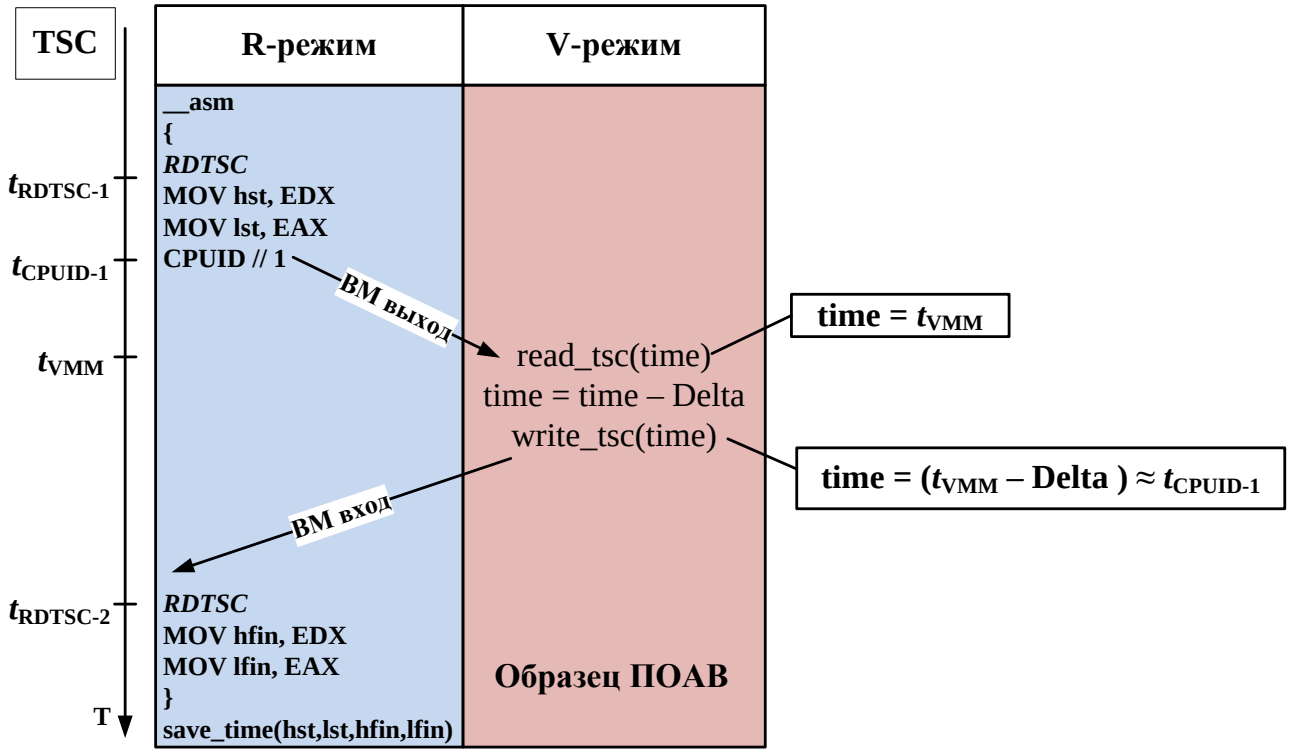


Рисунок 10 — Компрометация показаний счётчика тактов при перехвате БП-инструкций

Рассмотрим процесс компрометации счётчика тактов TSC в случае перехвата БП-инструкций.

При первом чтении показания счётчика регистрируется значение $t_{RDTSC-1}$. По окончании выполнения инструкции CPUID счётчик будет иметь значение $t_{CPUID-1}$.

В случае присутствия ПОАВ его образец получит управление. При этом в момент начала выполнения инструкций образца ПОАВ счётчик тактов будет иметь значение t_{VMM}. Образец ПОАВ записывает показания счётчика TSC во временную переменную «time». На рисунке это условно показано вызовом функции «read_tsc(time)». Далее из значения переменной «time» вычитается заданная величина компрометации «Delta» и записывается результат в переменную «time». Новое значение переменной «time» оператором «write_tsc(time)» записывается в счётчик тактов. После чего управление

возвращается в R-режим. При втором чтении счётчика TSC будет зарегистрировано значение $t_{RDTSC-2}$.

В случае отсутствия ПОАВ величина $t_{RDTSC-2}$ будет близка к величине $t_{CPUID-1}$.

В случае присутствия ПОАВ можно экспериментально подобрать величину компрометации «Delta» так, чтобы длительности выполнения трассы в случаях отсутствия и присутствия ПОАВ совпадали $t_{RDTSC-2} \approx t_{CPUID-1}$.

Для достижения точной компрометации счётчика TSC для ПОАВ с множеством условных переходов предлагается использовать трассировку инструкций [152]. Суть её состоит в том, чтобы вычитать не постоянное значение, а рассчитывать его с учётом длительности исполнения каждой отдельной инструкции, что позволит добиться точного совпадения средней длительности выполнения трассы в случаях отсутствия и присутствия ПОАВ.

Длительность выполнения инструкций t определяется как

$$t = t'' - t', \quad (1)$$

где и t' и t'' – регистрируемые показания счётчика тактов в начале и в конце выполнения инструкции.

Если в случае отсутствия ПОАВ измерение длительности выполнения инструкции t_{OT} выполняется многократно (в цикле), то схематично это можно представить, как показано на рис. 11 *а*.

Особенностью счётчика является недетерминированность его показаний (подробно об этом далее), из-за чего длительность выполнения инструкции представляется случайной величиной, многократно возрастающей в случае работы ЭВМ с образцом ПОАВ (рис. 11, *б*). Из этого следует, что если не будут предприняты надлежащие меры по искажению показаний счётчика с целью уменьшения времени $t_{ПР}$ до размера t_{OT}, обнаружить ПОАВ не составит труда [55].

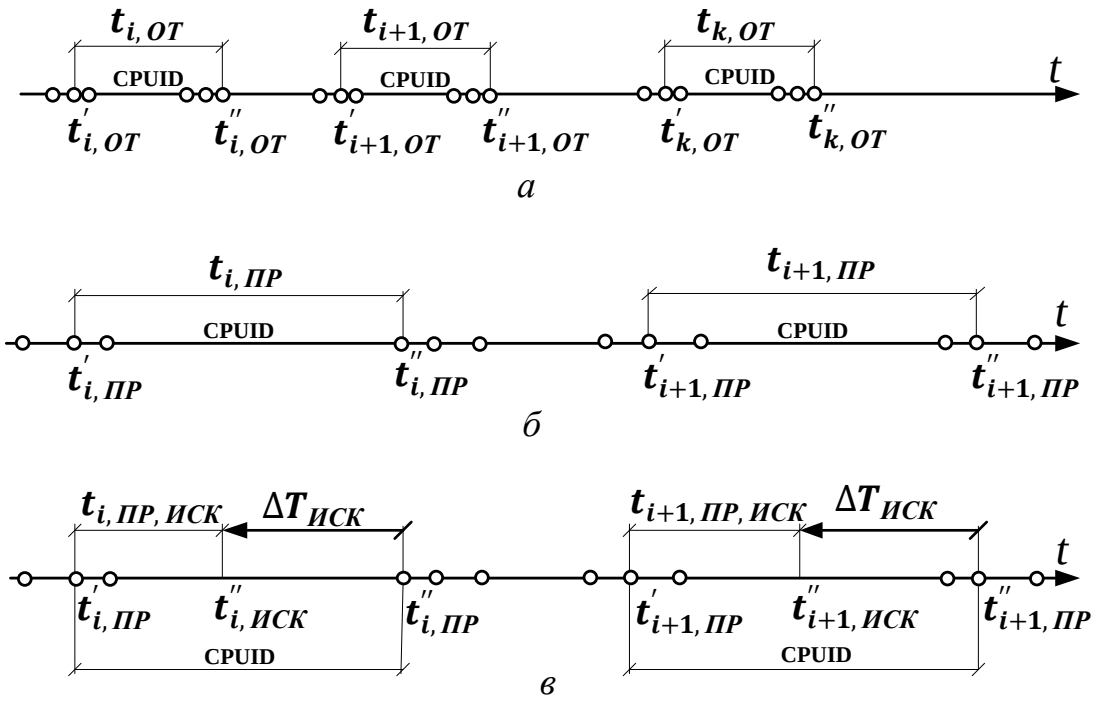


Рисунок 11 — Схема измерений длительности t выполнения в цикле трассы из одной инструкции CPUID: без ПОАВ (*а*), с ПОАВ без искажения показаний (*б*) и с ПОАВ с искажения второго показания счётчика (*в*)

Поэтому в целях сокрытия образцов ПОАВ показания счётчика должны искажаться так, чтобы регистрируемое время $t_{ПР}$ незначительно отличалось от аналогичного времени $t_{ОТ}$ (рис. 11, *в*). Этого можно достичь разработкой программы уменьшения показаний счётчика $t''_{ПР}$ на такую величину $\Delta T_{ИСК}$, которая делала бы незначительным отличие $t_{ПР,ИСК}$ от $t_{ОТ}$

$$t_{ПР,ИСК} = (t''_{ПР} - \Delta T_{\text{ИСК}}) - t'_{ПР} \approx t_{ОТ}. \qquad (2)$$

Описанный способ компрометации счётчика тактов не зависит от числа инструкций в трассе.

Компрометация при перехвате инструкций чтения счётчика

Наряду с рассмотренным способом, возможна компрометация счётчика тактов также при перехвате инструкций чтения счётчика. В этом случае, помимо БП-инструкций, образец ПОАВ должен обрабатывать и инструкции чтения счётчика.

Без ограничения общности в качестве инструкции чтения счётчика рассмотрим процессорную инструкцию RDTSC [136].

Представленная на рис. 12 схема измерения длительности выполнения трассы с учётом перехвата ПОАВ инструкций RDTSC аналогична схеме рис. 11, но с добавлением переключения между R- и V-режимами процессора при выполнении инструкции RDTSC.

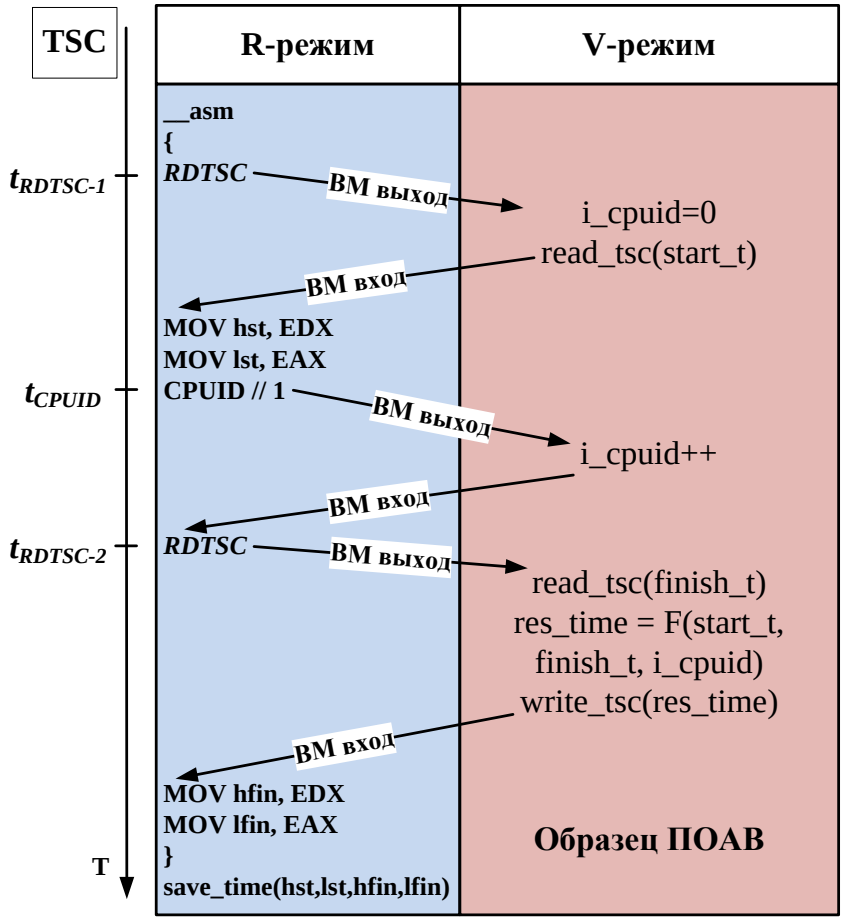


Рисунок 12 — Компрометация показаний счётчика тактов при перехвате инструкции чтения счётчика тактов RDTSC

При выполнении первой инструкции RDTSC обработчик обнуляет переменную-счётчик числа инструкций чтения счётчика и записывает показания счётчика во временную переменную «start_t». Далее, при выполнении каждой инструкции из трассы обработчик ПОАВ получает управление и инкрементирует переменную-счётчик (на схеме для этого приведён оператор «i_cpuid++». При выполнении второй инструкции RDTSC в обработчике выполняются следующие шаги:

– записываются показания процессорного счётчика тактов во временную переменную «finish_t»;

– рассчитывается значение переменной «res_time» с учётом значений счётчика до и после выполнения трассы, а также числа инструкций в трассе;
– полученное значение «res_time» записывается в счётчик TSC.

Описанный способ сложен в реализации, однако позволяет задавать точное значение счётчика, которое будет прочитано измеряющей программой при втором чтении.

Компрометация с использованием поля TSC_OFFSET структуры УСВМ

Технология аппаратной виртуализации предоставляет штатные средства для автоматической компрометации счётчика тактов. В случае технологии VT-x компрометация счётчика TSC достигается выставлением флага «use TSC offsetting» и указанием величины компрометации в MSR регистре «IA32_TIME_STAMP_COUNTER» [108]. В этих случаях при выполнении инструкции RDTSC происходит чтение MSR регистра IA32_TIME_STAMP_COUNTER, полученное значение суммируется со значением счётчика TSC, полученная сумма записывается в соответствующие регистры, подменяя при этом возвращаемое значение счётчика.

Данный подход реализован в ПОАВ «Blue Pill» [150]. Однако, согласно работе [152], при использовании этого подхода возникают трудности, обусловленные сложностью точного подбора величины компрометации.

2.2 Построение моделей выполнения трассы

В данном разделе анализируются схемы переключения между режимами работы процессора в случаях отсутствия (присутствия) одного образца (нескольких образцов) ПОАВ. Рассматриваются модели выполнения трассы в терминах теории графов и проверяется их адекватность опытным данным. На основе анализа моделей выявляются характерные особенности длительности выполнения трассы в случаях отсутствия (присутствия) ПОАВ и определяется критерий для обнаружения ПОАВ.

2.2.1 Сравнительный анализ схем переключения между режимами работы процессора

Для разработки подходов к обнаружению ПОАВ проводится анализ длительности выполнения трассы в случаях отсутствия (присутствия) одного образца (нескольких образцов) ПОАВ в ЭВМ.

Согласно работе Л. Дюфло (L. Duflot) [85] в случае отсутствия ПОАВ процессор может находиться в одном из двух режимов: либо в защищённом режиме (R-режиме), в котором выполняются пользовательские инструкции и трасса, либо в режиме системного управления (S-режиме). Режим системного управления документирован лишь частично и описан в работах К. Жмудзински (K. Zmudzinski) [172], Ш. Эмблетона (S. Embleton) [86].

Если в качестве основного принять первый режим, то смену указанных режимов в случае отсутствия ПОАВ можно представить схемой, приведённой на рис. 13.

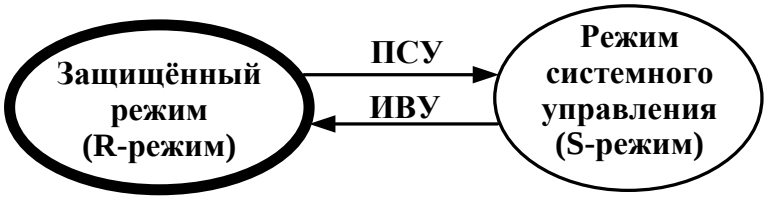


Рисунок 13 — Схема переключения между режимами работы процессора при выполнении набора инструкций в случае отсутствия ПОАВ в ЭВМ

При получении прерывания системного управления (ПСУ, англ. System Management Interrupt, SMI) процессор переходит из R- в S-режим, а возврат происходит при выполнении инструкции возврата управления (ИВУ, англ. RSM). Поскольку появление прерывания ПСУ имеет случайный характер, то передача управления в S-режим происходит с некоторой вероятностью, которая очень важна, и мы к этому вопросу ещё вернёмся. При отсутствии прерывания ПСУ перехода из R- в S-режим не происходит. Таким образом, процессор в случае отсутствия ПОАВ представляет собой систему со случайными переходами из одного состояния в другое.

В случае присутствия ПОАВ процессор может находиться не в двух, а в трёх режимах: сохраняются R- и S-режимы, добавляется режим монитора виртуальных машин (VMX root mode, V-режим). В этом случае R-режим называется режимом гостевой машины (VMX non root mode). В работе Ш. Эмблетона (S. Embleton) и Ш. Спаркс (S. Sparks) описано сравнение режима системного управления и режима монитора виртуальных машин [87].

На рис. 14 показана смена трёх режимов, из которых R-режим принят в качестве основного, режим S условно разнесли на S и S`. При выполнении БП-инструкции в R-режиме управление всегда передаётся в V-режим, из которого управление возвращается обратно (на рисунке это условно обозначено стрелками «ВМ выход» и «ВМ вход»).

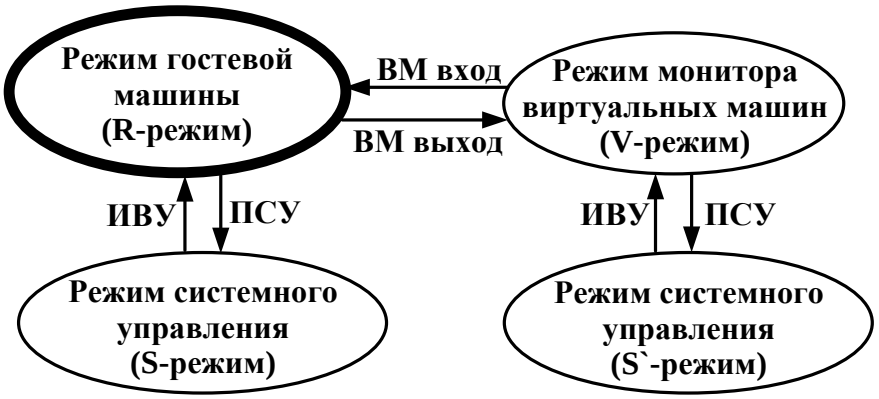


Рисунок 14 — Схема переключения между режимами процессора при выполнении набора инструкций в случае присутствия ПОАВ

При получении процессором ПСУ происходит переключение в S-режим, после чего управление возвращается в режим, в котором произошло прерывание. Такое переключение может произойти либо из R- либо из V-режима. Поскольку появление ПСУ случайно, то передача управления в S-режим происходит с некоторой вероятностью.

Как и в первом случае, работа процессора представляет собой вероятностную систему, но появление V-режима усиливает хаотичность смены режимов и, следовательно, увеличивается вариабельность статистик длительности выполнения трассы.

Из рассмотренного также следует, что в случае присутствия ПОАВ трасса выполняется дольше, следовательно, частота переключений в S-режим будет больше. Это значит, что длительность выполнения трассы и её показатели вариабельности существенно возрастают.

Эти выводы очень важны, и они будут учтены при разработке средств обнаружения ПОАВ в ЭВМ.

Рассмотренную схему переключения режимов процессора можно распространить и на случай, когда в ЭВМ не один, а несколько вложенных образцов ПОАВ. В этом случае процессор также может находиться в указанных выше трёх режимах, но ситуация будет отличаться наличием нескольких обработчиков ПОАВ (рис. 15). Каждый из них может быть также прерван переключением в S-режим, который на схеме разнесли на S, S` и S``.

Рисунок 15 — Схема переключения между режимами работы процессора при выполнении набора инструкций в случае присутствия вложенных образцов ПОАВ

Указанные обработчики ПОАВ работают в привилегированном V-режиме, который на схеме обозначен заштрихованным прямоугольником.

При выполнении каждой инструкции трассы в R-режиме происходит переключение в V-режим и управление, как по цепочке, передаётся последовательно от первого к последнему обработчику ПОАВ, после чего управление возвращается в R-режим (на схеме это обозначено стрелками «ВМ вход» и «ВМ выход») [25].

При получении процессором прерывания ПСУ происходит переключение в S-режим, после чего управление возвращается в режим, в котором произошло

прерывание. Такое переключение может произойти либо из R- либо из V-режима. Прерывание системного управления может прервать работу любого из обработчиков ПОАВ (на схеме это обозначено стрелками «ИВУ» и «ПСУ»).

Как и в предыдущем случае, функционирование процессора представляет собой вероятностную систему с той разницей, что в случае нескольких вложенных образцов ПОАВ длительность выполнения пользовательских функций и хаотичность смены режимов ещё более возрастает.

Анализ работы процессора позволяет перейти к построению моделей выполнения трассы, о чём будет сказано далее.

2.2.2 Особенности выполнения набора процессорных инструкций

Как отмечалось выше, выполнение процессором пользовательских инструкций сопровождается прерываниями и переходами из режима в режим, из-за чего регистрируемая длительность выполнения этих инструкций случайна и существенно больше фактической длительности.

Чтобы выяснить, каким образом оказывают влияние прерывания в работе процессора и режимы ЭВМ на длительность выполнения пользовательских инструкций, начнём с рассмотрения выполнения фрагмента программы для определения длительности выполнения процессором трассы в некотором гипотетическом режиме ЭВМ без ПОАВ и без прерываний (рис. 16).

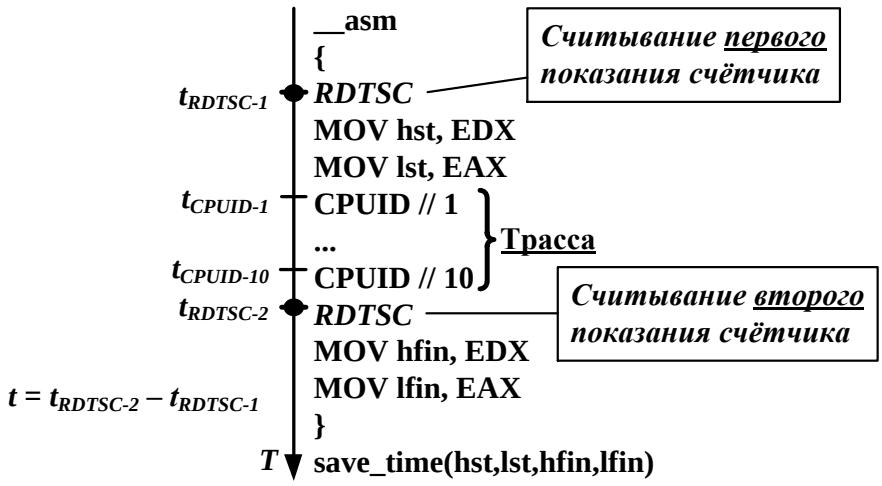


Рисунок 16 — Фрагмент программы измерения длительности трассы

Считывание счётчика тактов осуществляется до и после выполнения трассы с использованием инструкции RDTSC. Слева от фрагмента исходного кода программы приведена ось времени, на которой отмечены моменты выполнения инструкций считывания счётчиков тактов ($t_{RDTSC-1}$ и $t_{RDTSC-2}$) и инструкций из трассы $t_{CPUID-1}$ и $t_{CPUID-10}$.

Регистрируемая длительность выполнения трассы t определяется по показаниям счётчика TSC как

$$t = t_{RDTSC-2} - t_{RDTSC-1}. \quad (2)$$

Если бы процессор выполнял только указанные в программе инструкции, т.е. без прерываний и смены режимов, то регистрируемая длительность выполнения трассы t являлась бы фактической длительностью. Однако в реальных условиях из-за прерываний в работе процессора регистрируемая длительность превышает «чистую» длительность, определить которую, как вытекает из изложенного, не представляется возможным.

На рис. 17 представлена схема для случая отсутствия ПОАВ, но с учётом прерываний и смены режимов процессора.

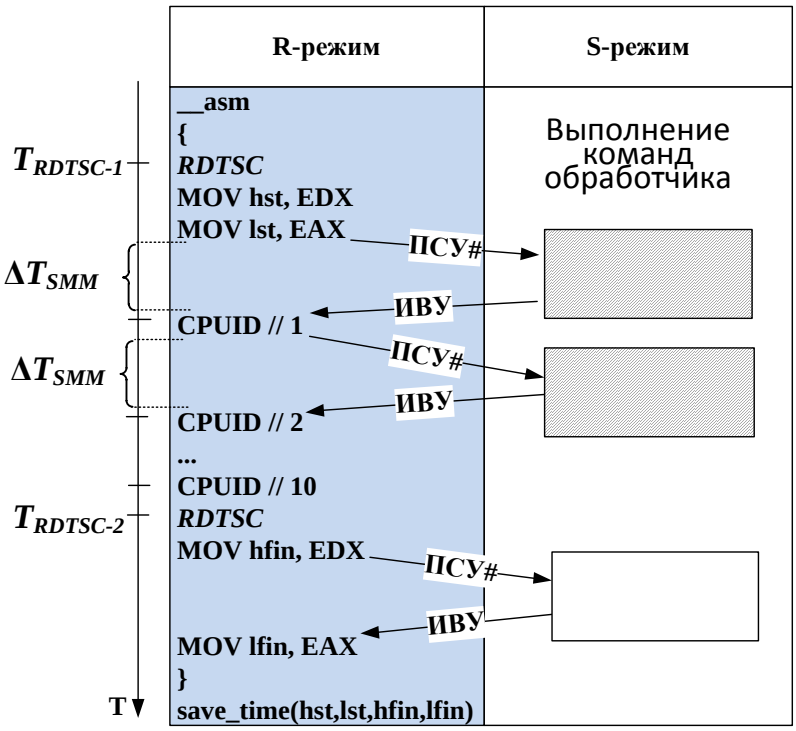


Рисунок 17 — Смена режимов процессора при выполнении трассы в случае отсутствия ПОАВ

Левый прямоугольник соответствует защищённому режиму работы процессора (R-режим), в котором осуществляется измерение длительности выполнения трассы. Правый – соответствует режиму системного управления (S-режим), в котором находится его обработчик. Стрелками условно обозначены переключения между режимами.

Как уже отмечалось, смена режимов работы процессора по ходу выполнения инструкций из трассы может происходить многократно. Вследствие этого регистрируемая по показаниям процессорного счётчика тактов длительность выполнения трассы, во-первых, оказывается больше длительности её фактического выполнения, поскольку будет складываться из длительности выполнения трассы и длительности переключения и работы в S-режиме; во-вторых, эта величина случайная, поскольку указать количество прерываний не представляется возможным.

При этом, поскольку кванты времени для случаев отсутствия и присутствия ПОАВ постоянные, а множества состояний процессора счётные, следует ожидать, что при многократном выполнении пользовательской инструкции или их набора (трассы) измеряемая длительность выполнения трассы будет распределяться, хотя и случайным образом, но по некоторым фиксированным уровням. Если экспериментально это подтвердится и окажется, что эти уровни зависят от наличия ПОАВ, то выявленную особенность выполнения трассы, наряду с другими, можно будет также использовать для обнаружения ПОАВ.

Схема определения длительности выполнения трассы и переключений процессора в случае присутствия одного образца ПОАВ представлена в трёхмерных координатах рис. 18. По сравнению с предыдущей схемой добавился ещё V-режим процессора.

Переключение между режимами показано линиями. При выполнении каждой инструкции трассы управление передаётся из R- в V- режим, после чего возвращается в R-режим. При этом возможны переключения в S-режим как из R-, так и из V-режима.

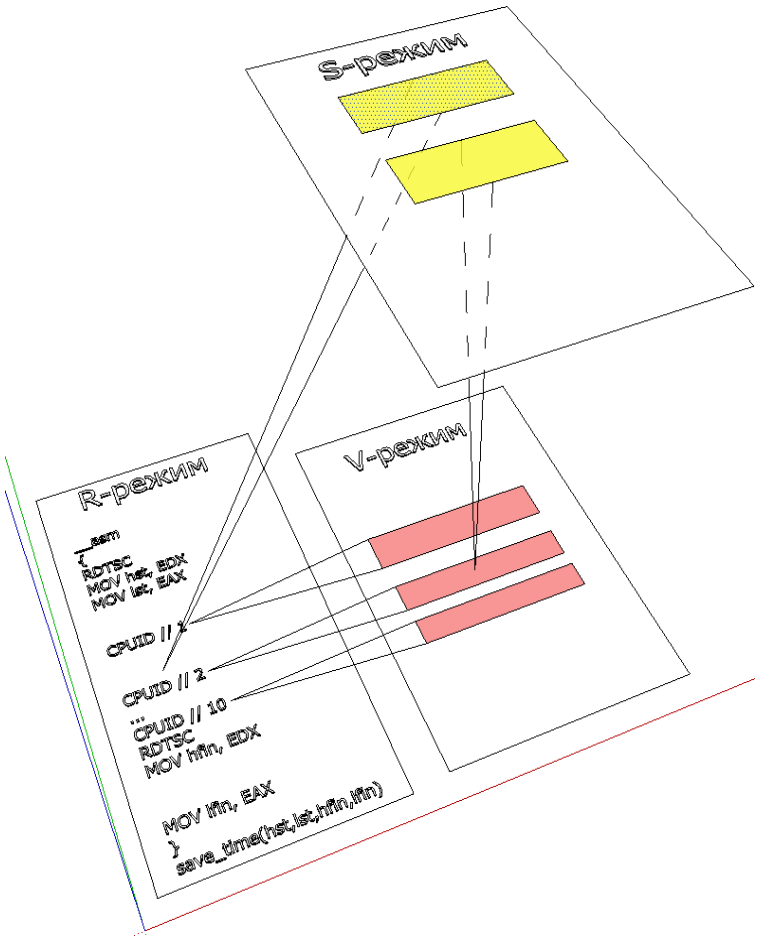


Рисунок 18 — Смена режимов процессора при выполнении трассы в случае присутствия одного образца ПОАВ

Вследствие этого значительно возрастут и величина, и вариабельность длительности выполнения трассы, что станет отличительной особенностью случая, когда в ЭВМ содержится образец ПОАВ.

Вместе с тем ПОАВ при перехвате инструкций трассы может компрометировать значения счётчиков тактов, так что средняя длительность выполнения трассы будет мало отличаться от длительности в случае отсутствия ПОАВ.

При выполнении трассы в случае присутствия нескольких образцов ПОАВ, каждый из них имеет возможность обрабатывать запросы предыдущего, поэтому длительность выполнения трассы ещё более возрастёт как по величине, так и по варьированию, что может быть использовано для обнаружения ПОАВ и подсчёта количества образцов ПОАВ в ЭВМ.

Выявленные особенности выполнения трассы в случаях отсутствия и присутствия ПОАВ позволяют указать некоторые признаки для обнаружения ПОАВ, а также построить модельные зависимости длительности выполнения трассы от характеризующих работу процессора параметров.

2.2.3 Построение моделей выполнения трассы

Представляемые далее модели основываются на следующих допущениях:

– все инструкции процессора выполняются последовательно на одном вычислительном устройстве;
– длительность выполнения каждой инструкции одинакова и составляет k единиц (тактов);
– при получении управления ПОАВ компрометирует счётчик тактов на некоторую постоянную величину;
– случайное число переключений процессора из R- в S-режим при выполнении инструкции подчиняется биномиальному распределению.

Модель выполнения трассы в случае отсутствия ПОАВ

В случае отсутствия ПОАВ выполнение трассы можно представить в виде ориентированного помеченного мультиграфа (рис. 19) [51].

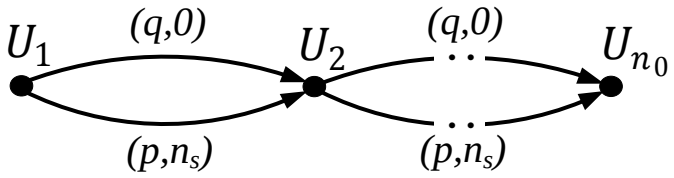


Рисунок 19 — Модель выполнения трассы в случае отсутствия ПОАВ

Вершинами графа $U_1, U_2, \dots, U_{n_0}$ являются инструкции трассы, n_0 – число инструкций в трассе, n_S – число инструкций в обработчике S-режима, дуги соответствуют возможным переключениям между режимами процессора. При выполнении каждой инструкции трассы процессор может перейти с некоторой вероятностью p из R- в S-режим, на схеме это показано нижней дугой. Как альтернативный вариант, инструкция выполняется в R-режиме без перехода в

S-режим, на схеме это отмечено верхней дугой. Вероятность такого события $q = 1 - p$ [8].

Длительность выполнения трассы t_{OT} из n_0 инструкций в случае отсутствия ПОАВ можно выразить формулой (3), где случайная величина m_{OT}, соответствующая числу переключений в S-режим, подчиняется закону биномиального распределения с параметрами n_0 и p [12]

$$t_{OT} = (n_0 + m_{OT} * n_S) * k. \quad (3)$$

В случае присутствия одного образца ПОАВ модель выполнения трассы представляется также ориентированным мультиграфом с помеченными дугами, который представлен на рис. 20, с тем отличием, что при выполнении каждой инструкции трассы U_i управление передаётся в ПОАВ, инструкции которого на рисунке помечены символами V_i, $i = 1, .., n_V$, где n_V – число инструкций в этом образце.

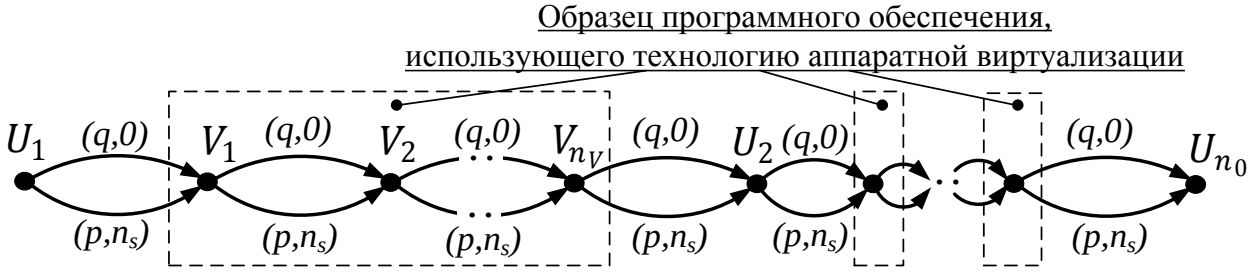


Рисунок 20 — Модель выполнения трассы в случае присутствия одного образца ПОАВ

При выполнении любых инструкций как трассы, так и ПОАВ, возможен переход в S-режим (на рисунке это обозначено нижними дугами). Ситуации, когда таких переходов нет, обозначены верхними дугами.

Длительность выполнения трассы $t'_{ПР}$ из n_0 инструкций в случае присутствия одного ПОАВ без компрометации счётчика тактов можно представить формулой

$$t'_{ПР} = (n_0 + n_0 * n_V + m_{ПР} * n_S) * k, \quad (4)$$

где $m_{ПР}$, соответствующая числу переключений в S-режим, подчиняется закону биномиального распределения с параметрами $(n_0 + n_0 * n_V)$ и p, слагаемое

$n_0 * n_V$ соответствует тому, что при выполнении каждой из n_0 инструкций трассы происходит переключение в V-режим (передача управления образцу ПОАВ), на что тратится дополнительное время.

Если при обработке каждой из n_0 инструкций трассы образец ПОАВ осуществляет компрометацию счётчика путём прибавления к его значению некоторой величины $\delta < 0$, то длительность выполнения трассы $t_{ПР}$ из n_0 инструкций выразится как

$$t_{ПР} = (n_0 + n_0 * n_V + m_{ПР} * n_S + n_0 * \delta) * k. \quad (5)$$

В случае присутствия нескольких вложенных образцов ПОАВ модель выполнения трассы представляется ориентированным мультиграфом с помеченными дугами (рис. 21).

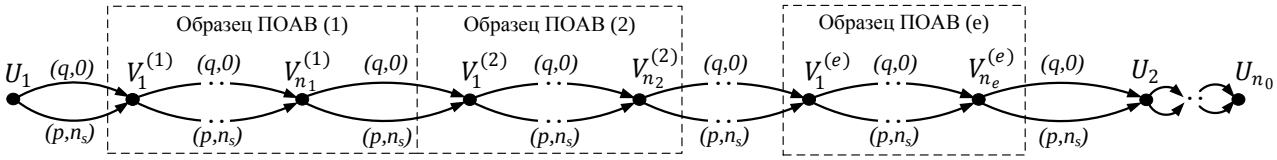


Рисунок 21 — Модель выполнения трассы в случае присутствия e-вложенных образцов ПОАВ

Данный граф аналогичен предыдущему графу за исключением того, что управление передаётся последовательно каждому образцу ПОАВ. Вершина $V_i^{(j)}$, где $i = 1, \dots, n_j,\ j = 1, \dots, e$ интерпретируется как i-я инструкция j-го образца ПОАВ; n_j – число инструкций в обработчике j-го ПОАВ; e – число вложенных образцов ПОАВ.

Тогда длительность выполнения трассы определяется как

$$t_{ВЛ} = \left(n_0 + n_0 * \sum_{j=1}^{e} n_j + m_{ВЛ} * n_S + n_0 * \sum_{j=1}^{e} \delta_j \right) * k, \quad (6)$$

где $m_{ВЛ}$ – случайная величина, соответствующая числу переключений в S-режим и подчиняющаяся закону биномиального распределения с параметрами $(n_0 + n_0 * \sum_{j=1}^{e} n_j)$ и p. При этом вероятность переключения в S-режим p осталась неизменной, а число инструкций, при выполнении которых возможен переход в S-режим (количество независимых испытаний в терминах

схемы Бернулли), увеличилось по сравнению со случаем отсутствия ПОАВ на величину $n_0 * \sum_{j=1}^{e} n_j$, свидетельствующую о возрастании числа переключений в S-режим. Сумма $\sum_{j=1}^{e} n_j$ обусловлена тем, что выполнение очередной инструкции трассы прерывается и обрабатывается поочерёдно каждым вложенным образцом ПОАВ. Слагаемое $n_0 * \sum_{j=1}^{e} \delta_j$ означает величину длительности компрометации счётчика при обработке инструкций каждым образцом ПОАВ. При передаче управления j-му образцу ПОАВ он прибавляет к значению счётчика тактов величину δ_j, $\delta_j < 0, j = 1, \dots, e$.

2.2.4 Проверка адекватности построенных моделей опытным данным

Проверка состояла в определении возможности получения по формулам (3) – (6) значений длительностей выполнения трассы, близких к реально наблюдаемым в случаях отсутствия и присутствия ПОАВ с искажением показаний счётчика тактов, в также в оценке критерия согласия распределений расчётной и наблюдаемой длительности выполнения трассы.

Было установлено, что путём подбора приемлемых значений, входящих в (3) – (6) параметров, можно получить совпадение расчётных величин длительности выполнения трассы с опытными данными, из чего вытекает, что модельные формулы не противоречат данным из опыта.

Для оценки согласия модельных и экспериментальных распределений сначала были выполнены опыты, состоящие в получении массивов длительности выполнения трассы путём выполнения в непрерывном цикле 1000 измерений длительности выполнения трассы из 10-ти инструкций CPUID на различных ЭВМ с таким искажением показаний счётчика тактов, при котором выборочные средние длительности выполнения трассы в случаях отсутствия и присутствия ПОАВ практически не различались.

Характерной особенностью получаемых гистограмм и полигонов распределения было наличие длинного правого «хвоста», свидетельствующего

о наличии относительно большого количества низкочастотных вариант в вариационных рядах длительности выполнения трассы. В таких случаях, как известно [49], дисперсия и моменты 4-го порядка оказываются очень большими. Настолько, что различия между указанными статистиками в случаях отсутствия и присутствия ПОАВ получить удаётся далеко не во всех опытах. Для преодоления этой проблемы получаемые из опыта массивы длительности выполнения трассы подвергались низкочастотной фильтрации.

В результате даже при малом уровне *фильтрации* (всего 0,02) длина вариационных рядов сокращалась с 10-15 до стабильных 2-4 вариант (рис. 22, 23). Иными словами, рассматривалась только та часть выборок, значения которой определяли характерные особенности выборок.

Благодаря таким мерам, обеспечивалась приемлемая различимость статистик, используемых для обнаружения ПОАВ. В разделе 3.2 показано, как это достигалось. Другими характерными особенностями опытных данных были недостаточные сходимость и воспроизводимость результатов измерений [9], состоящие, как это видно на приведенных рисунков, в смещённости одних полигонов относительно других при неизменных условиях повторных опытов. В разделе 3.2 показано, как преодолевалась и эта проблема.

Генерация массивов длительности выполнения трассы с использованием модельных формул (3) и (5), подчиняющихся биномиальному распределению, производилась в среде Matlab с использованием функции binornd [119]. Для полученных массивов строили вариационные ряды, представленные модельными полигонами на рис. 22 и 23.

Установлено, что в случаях отсутствия ПОАВ модельные полигоны полностью совпадают с экспериментальными (на рис. 22 для наглядности сравниваемые полигоны искусственно разнесены по горизонтали на 1 такт). В случае присутствия ПОАВ согласие распределений оценивалось по критерию Колмогорова [6], [7], [16], [59], [90] с использованием Matlab-функции kstest2. Проверка показала, что гипотеза о согласии модельного и экспериментального распределений на 5%-ом уровне значимости не отвергается. Правомерность

применения критерия согласия Колмогорова для дискретных распределений показана в [7], [59], [90].

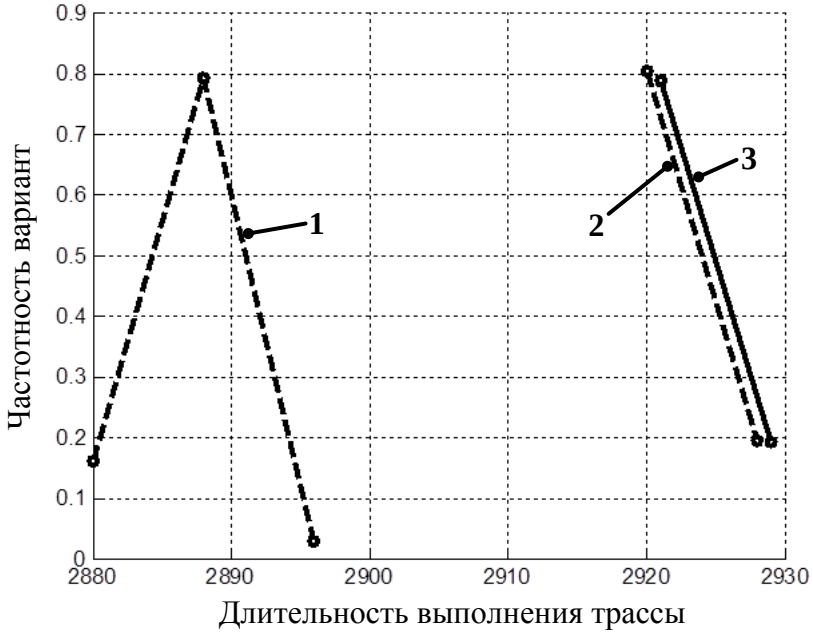


Рисунок 22 — Экспериментальные (1 и 2) для двух повторных опытов и модельный (3) полигоны распределений длительности выполнения трассы в случае отсутствия ПОАВ для ЭВМ с процессором Intel Core 2 Duo E8200 под управлением ОС Windows 7

Аналогичные результаты получены и для других опытных данных. Поэтому было принято решение, что гипотеза об адекватности предложенных моделей выполнения трассы опытным данным не отвергается и что эти модели можно использовать в аналитических целях.

Из рис. 22 видно, что полученные по двум повторным опытам полигоны в случае отсутствия ПОАВ имеют различные значения средней арифметической длительности выполнения трассы, поскольку интервалы варьирования в первом опыте были 2880-2890 тактов, а во втором – 2920-2930 тактов.

Подобно предыдущему случаю, полигоны в случае присутствия ПОАВ также имеют различные значения средней арифметической и их интервалы варьирования: 2875–2900 тактов, 2908–2935 тактов.

Аналогичные закономерности наблюдаются и на других ЭВМ в случаях отсутствия и присутствия ПОАВ.

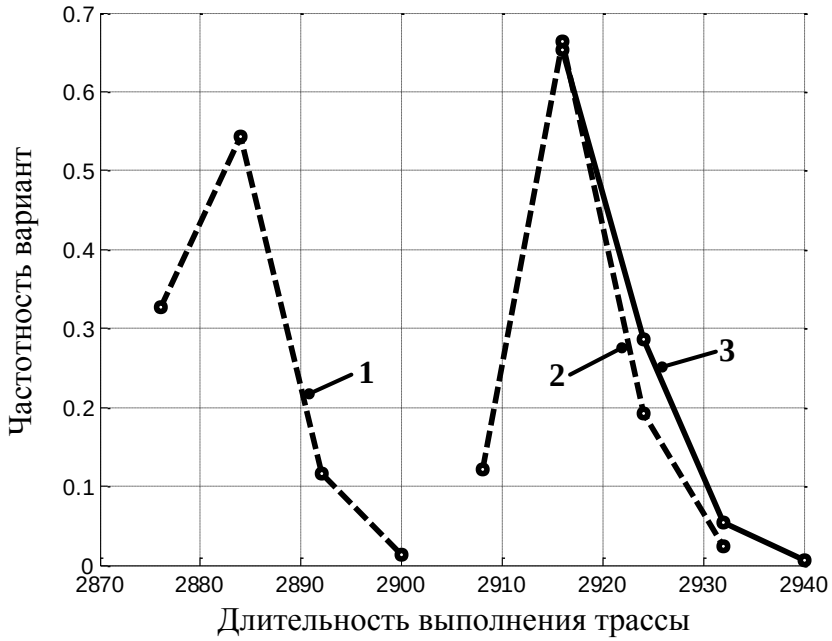


Рисунок 23 — Экспериментальные (1 и 2) и модельный (3) полигоны распределения длительности выполнения трассы в случае присутствия ПОАВ с искажением счётчика тактов в ЭВМ с процессором Intel Core 2 Duo E8200 и ОС Windows 7

Приведенные опытные данные свидетельствуют об их недостаточной сходимости, что затрудняет полноценную проверку на адекватность полученных модельных выражений. Представляется возможным для каждого экспериментального полигона отдельно подобрать модельное распределение, как это сделано для правого опытного полигона. Однако при проверке критерия согласия выбранного модельного и различных экспериментальных распределений, полученных на той же ЭВМ, было обнаружено, что не все опытные данные согласуются с модельными. Модельное распределение с постоянными параметрами не может одновременно совпадать со всеми данными повторных опытов из-за недостаточной их сходимости (повторяемости), поэтому решение задачи классическими методами математической статистики путём проверки гипотез согласия не представляется возможным.

Описанную особенность распределений длительности выполнения трассы, выражающуюся различием значений среднего арифметического повторных опытов, будем называть *дрейфом данных*.

Было принято решение об использовании альтернативного критерия присутствия ПОАВ, который описан в 2.2.5.

Представляется возможным использование и других закономерностей длительности выполнения трассы для обнаружения ПОАВ. Так из выражения (5) следует, что если число переключений $m_{ПР}$ в S-режим окажется достаточно малым,

$$m_{ПР} < (1 + n_V + \delta)\frac{n_0}{n_S}, \qquad (7)$$

где $\delta < 0$, то измеряемая длительность выполнения трассы $t_{ПР}$ в ЭВМ с ПОАВ приобретёт отрицательное значение. А если такие события будут происходить достаточно часто, то и среднее выборочное длительности выполнения трассы $t_{ПР}$ тоже может стать отрицательным.

Следовательно, если при обнаружении ПОАВ в получаемых массивах длительности выполнения трассы обнаружатся отрицательные значения, то это будет указывать на то, что в ЭВМ имеется ПОАВ и что искажение показаний счётчика тактов с целью сокрытия ПОАВ выполняется некачественно.

Выражения (3) – (6) позволяют также выявить, насколько надёжно и точно можно осуществлять искажение показаний счётчика тактов.

Поскольку в этих выражениях переменные $m_{ОТ}$, $m_{ПР}$ и $m_{ВЛ}$ являются случайными величинами, то длительности выполнения трассы $t_{ОТ}$, $t'_{ПР}$, $t_{ПР}$ и $t_{ВЛ}$ могут рассматриваться как линейные функции случайного аргумента. Из этого следует, что, согласно [8], законы распределения длительности выполнения трассы и числа переключений в S-режим совпадают и можно определить их числовые характеристики через законы распределения аргументов.

При биномиальном распределении математическое ожидание длительности выполнения трассы $E[t_{OT}]$ с учётом [3] в случае отсутствия ПОАВ выразится как

$$E[t_{OT}] = n_0 * k * (p * n_S + 1). \quad (8)$$

В случае присутствия ПОАВ с компрометацией счётчика тактов на величину δ математическое ожидание длительности выполнения трассы $E[t_{ПР}]$ определится как

$$E[t_{ПР}] = n_0 * k * (\delta + p * n_S(n_V + 1) + n_V + 1). \quad (9)$$

Приравняв правые части (8) и (9), после преобразований получим выражение для величины искажения счётчика тактов $[\delta]$, обеспечивающей неразличимость математических ожиданий длительности выполнения трассы $\overline{t}_{OT}$ и $\overline{t}_{ПР}$

$$[\delta] = -(p * n_S + 1) * n_V. \quad (10)$$

Из (10) следует, что теоретически имеется возможность такого искажения показаний счётчика, при котором по выборочному среднему длительности выполнения трассы $\overline{t}_{тест}$ выявить наличие ПОАВ в тестируемом ЭВМ будет невозможно. Однако, поскольку параметры выражения (10) неизвестны, определить значение $[\delta]$ очень сложно. Поэтому для сокрытия ПОАВ нарушителю остаётся только экспериментальный подбор такой величины искажения показаний счётчика $[\delta]$, чтобы наблюдаемые средние арифметические длительности выполнения трассы $\overline{t}_{OT}$ и $\overline{t}_{ПР}$ совпадали бы с той или иной точностью.

Однако этому препятствуют сложности, связанные с непостоянством (случайностью) выборочных средних при повторных опытах. Это значит, что при постоянном выбранном $[\delta]$ в силу случайности величин переключений процессора в S-режим m_{OT}, $m_{ПР}$ и $m_{ВЛ}$ выборочные средние $\overline{t}_{ПР}$ даже при самом тщательном выборе величины искажения счётчика тактов $[\delta]$ могут значимо отличаться от выборочных средних $\overline{t}_{OT}$: выборочные средние $\overline{t}_{ПР}$

могут быть как существенно бóльшими, так и меньшими выборочных средних $\overline{t}_{OT}$.

Из этого следует, что нарушитель для сокрытия ПОАВ не располагает 100%-ой вероятностью для достаточно точного искажения показаний счётчика. Поэтому тестирование ЭВМ на наличие ПОАВ следует начинать с проверки тестовой выборки $T_{тест}$ длительности выполнения трассы на наличие в ней отрицательных значений и с сопоставления выборочной средней $\overline{t}_{тест}$ с нормативной $[\overline{t}_{OT}]$, значение которой можно установить до начала тестирования ЭВМ.

Анализ модельных формул позволяет выявить также ещё один признак для обнаружения ПОАВ.

Поскольку переменные в формулах (3) – (6) принимают только положительные целые значения из некоторых фиксированных множеств, то значения длительности выполнения трассы в случаях отсутствия и присутствия ПОАВ в повторных опытах, хотя и будут представляться случайными величинами, однако распределяться они могут и по некоторым *постоянным* уровням. Это подтверждается полученными из опытов полигонами рисунков 22, 23.

По этой же причине точечные графики длительности выполнения трассы должны иметь уровневый (слоистый) характер. При этом следует ожидать, что в случае отсутствия ПОАВ в ЭВМ число таких уровней и их значения будут постоянными, в то время как в других случаях они будут отличаться. Эти отличия также можно использовать в целях обнаружения ПОАВ.

Таким образом, выполненный анализ предложенных моделей работы процессора позволил наметить показатели, по которым можно выявлять наличие ПОАВ в ЭВМ. В главе 3 эти показатели будут подвергнуты экспериментальной проверке.

Другие идентификационные показатели для обнаружения ПОАВ представлены в следующем подразделе.

2.2.5 Критерий присутствия программного обеспечения, использующего технологию аппаратной виртуализации

Предлагается критерий присутствия образца ПОАВ, позволяющий различать массивы (выборки) длительности выполнения трассы в случаях отсутствия и присутствия ПОАВ [21].

Пусть параметры $n_0, n_S, n_V, \delta, k \in \mathbb{Z}$; $p \in \mathbb{R}$ – некоторые фиксированные значения. Пусть X_{OT} – множество значений, полученных при реализации t_{OT}, $X_{ПР}$ – множество значений, полученных при реализации $t_{ПР}$. Пусть генеральная совокупность X: $X = X_{OT} \cup X_{ПР}$. Пусть $\vec{x}_n = (x_1, .., x_n)$ – выборка объёма n из генеральной совокупности X.

Критерий присутствия образца ПОАВ на основе дисперсии.

Критическое множество (образец ПОАВ присутствует): $W = \{\vec{x}_n : \hat{\sigma}^2(\vec{x}_n) \geq d\}$, где $\hat{\sigma}^2$ – выборочная дисперсия, $d \in \mathbb{Z}$ – определяется экспериментально.

Принятие решения: если $\vec{x}_n \in W$, то образец ПОАВ присутствует, если $\vec{x}_n \notin W$, то образец ПОАВ отсутствует.

Критерий присутствия образца ПОАВ на основе момента 4–го порядка.

Критическое множество (образец ПОАВ присутствует): $W = \{\vec{x}_n : \hat{\upsilon}_2(\vec{x}_n) \geq \mu\}$, где $\hat{\upsilon}_2$ – выборочный центральный момент 4–го порядка, $\mu \in \mathbb{Z}$ – определяется экспериментально.

Принятие решения: если $\vec{x}_n \in W$, то образец ПОАВ присутствует, если $\vec{x}_n \notin W$, то образец ПОАВ отсутствует.

Критерий присутствия образца ПОАВ на основе длины вариационного ряда.

Критическое множество (образец ПОАВ присутствует): $W = \{\vec{x}_n : \hat{l}(\vec{x}_n) \geq e\}$, где $\hat{l}$ – длина вариационного ряда, построенного по выборке, $e \in \mathbb{Z}$ – определяется экспериментально.

Принятие решения: если $\vec{x}_n \in W$, то образец ПОАВ присутствует, если $\vec{x}_n \notin W$, то образец ПОАВ отсутствует.

Пороговые значения d, μ, e, значения вероятностей ошибок первого и второго рода α и β, а также объём выборки n определяются экспериментально.

Предложенные критерии присутствия образца ПОАВ проверяются и уточняются в главе 3.

Утверждение 1. Пусть t – выборка объёма n, $f(t) = D_в[t]$ – выборочная дисперсия выборки t, тогда

$$D_в[t_{ПР}] = D_в[t_{ОТ}] * (n_V + 1) \quad (11)$$

Доказательство.

Приведём значения дисперсий для случайных величин $t_{ОТ}$ и $t_{ПР}$ [3].

Выполняются равенства:

$$D_в[t_{ОТ}] = n_0 * (1 - p) * p * k^2 * n_S^2 \quad (12)$$

$$D_в[t_{ПР}] = n_0 * (1 - p) * p * k^2 * n_S^2 * (n_V + 1) \quad (13)$$

Из равенств (12) и (13) получаем, что:

$$\frac{D_в[t_{ПР}]}{D_в[t_{ОТ}]} = \frac{n_0 * (1 - p) * p * k^2 * n_S^2 * (n_V + 1)}{n_0 * (1 - p) * p * k^2 * n_S^2} = n_V + 1 \quad (14)$$

Значит, справедливо (11). Утверждение доказано.

Из формулы (11) следует, что значение выборочной дисперсии в случае присутствия ПОАВ превышает в $(n_V + 1)$ раз значение выборочной дисперсии в случае отсутствия ПОАВ.

Утверждение 2. Пусть t – выборка объёма n, $f(t) = \mu_в[t]$ – выборочный центральный момент 4-го порядка выборки t, тогда $\mu_в[t_{ПР}] > \mu_в[t_{ОТ}]$.

Доказательство.

Приведём значения центральных моментов 4-го порядка для случайных величин $t_{ОТ}$ и $t_{ПР}$ [3].

Выполняются равенства:

$$\mu_в[t_{ОТ}] = n_0 * p * q * (3 * (n_0 - 2) * p * q + 1) \quad (15)$$

$$\mu_в[\mathrm{t}_{ПР}] = (n_0 + n_0 * n_V) * p * q * (3 * (n_0 + n_0 * n_V - 2) * p * q + 1) \quad (16)$$

Из равенств (15) и (16) получаем, что:

$$\frac{\mu_в[\mathrm{t}_{ПР}]}{\mu_в[\mathrm{t}_{ОТ}]} = \frac{(n_0 + n_0 * n_V) * p * q * (3 * (n_0 + n_0 * n_V - 2) * p * q + 1)}{n_0 * p * q * (3 * (n_0 - 2) * p * q + 1)} \quad (17)$$

Приведём подобные члены, получим,

$$\frac{\mu_в[\mathrm{t}_{ПР}]}{\mu_в[\mathrm{t}_{ОТ}]} = 1 + 6 * \frac{n_0 * n_V}{3 * (n_0 - 2) + \frac{1}{p * q}} > 1 \quad (18)$$

Значит, при $n_0 > 2$

$$\mu_в[t_{ПР}] > \mu_в[t_{ОТ}] \quad (19)$$

Из формулы (19) следует, что значение моментов 4-го порядка в случае присутствия ПОАВ превышает значение моментов 4-го порядка в случае отсутствия ПОАВ. Утверждение доказано.

Гипотеза. Пусть t – выборка объёма n, $f(t) = v_в[t]$ – значение длины вариационного ряда выборки t, тогда $v_в[t_{ПР}] > v_в[t_{\mathrm{ОТ}}]$.

Экспериментальная проверка. Графики полигонов модельных распределений в случае отсутствия и присутствия ПОАВ приведены на рис. 24. Из приведённых графиков видно, что число точек (уровней) в случае присутствия ПОАВ намного больше, чем в случае его отсутствия. Эта отличительная особенность подтверждается многочисленными опытами на разных ЭВМ.

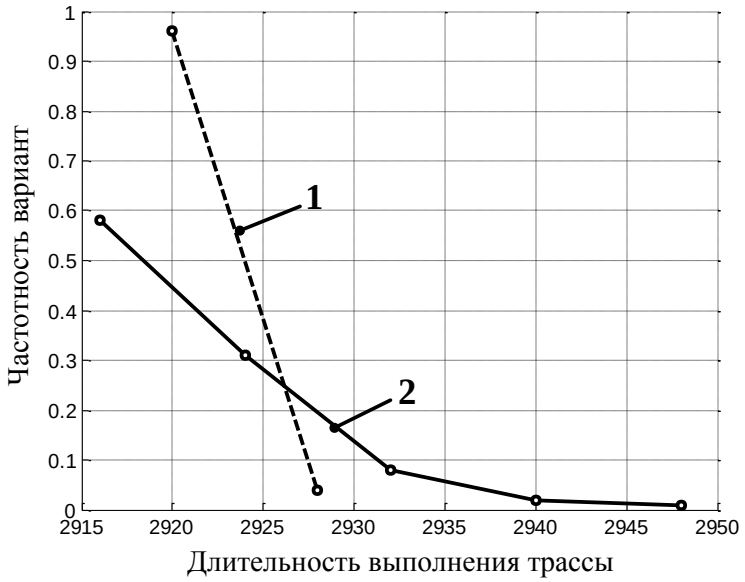


Рисунок 24 — Графики полигонов двух модельных распределений в случае отсутствия ПОАВ (1) и присутствия ПОАВ (2)

2.3 Выводы

1. На основе анализа схем переключения между режимами работы процессора получен ряд моделей выполнения трассы, устанавливающих зависимость длительности выполнения трассы от параметров, характеризующих условия функционирования процессора в случаях присутствия и отсутствия ПОАВ.
2. Выполненный анализ полученных моделей выявил показатели распределения длительности выполнения трассы для обнаружения ПОАВ в ЭВМ: таких как выборочные средние длительности выполнения трассы, выборочные дисперсии и моменты 4-го порядка, показатели вариационных рядов и другие статистики.
3. Показано, что для обеспечения адекватности предложенных моделей опытным данным и для обеспечения различимости статистик длительности выполнения трассы в случаях отсутствия и присутствия ПОАВ получаемые из опытов массивы длительности выполнения трассы необходимо отфильтровывать от низкочастотных значений.
4. Анализ моделей длительности выполнения трассы также выявил ограниченные возможности у нарушителя для сокрытия ПОАВ в системах. Установлено, что из-за случайного характера регистрируемых значений длительности выполнения трассы вполне возможны ситуации, когда предпринимаемые нарушителем действия окажутся недостаточными для сокрытия ПОАВ и их можно будет просто обнаружить. С учётом этого предложенные рекомендации по организации тестирования систем на наличие в них ПОАВ после экспериментальной проверки следует использовать в разрабатываемых алгоритмах обнаружения ПОАВ в ЭВМ.
5. Изложенные в данной главе результаты теоретического анализа предопределили задачи и программу экспериментальных исследований, описываемых в следующей главе.

3 Исследование статистических характеристик длительности выполнения набора процессорных инструкций и разработка методики обнаружения нелегитимного программного обеспечения, использующего технологию аппаратной виртуализации

Доступных публикаций по данной теме крайне мало, а содержащиеся в них сведения носят ограниченный и противоречивый характер. Так, в работе [95] приводятся точечные графики реализаций случайного процесса измерений длительности выполнения трассы, однако анализа, выводов и предложений по обнаружению ПОАВ в системах не сделано.

В другой работе [82], наоборот, предлагается подход к проблеме обнаружения ПОАВ, основанный на предположении, что распределение длительности выполнения трассы подчиняется нормальному закону. При этом опытных подтверждений этому не приводится, а также не учитываются возможные противодействия нарушителя попыткам обнаружить ПОАВ. Кроме того, вопреки требованиям международного метрологического стандарта ISO 5725 [9], не приводится никаких сведений о прецизионности (сходимости и воспроизводимости) опытных данных.

Некоторые из этих вопросов затрагивались в подразделе 2.2.4, но поскольку приведённые выше замечания имеют принципиальный характер, они нуждаются в дополнительном рассмотрении.

Выполненные нами многочисленные опыты на различных ЭВМ свидетельствуют, что во всех без исключения опытах длительности выполнения трассы распределяются не по нормальному закону. Для иллюстрации этого на рис. 25 приведены типичные гистограммы, состоящие чаще всего из одного или двух высоких столбцов и легковесного правого «хвоста». Другая особенность – неудовлетворительная сходимость опытных данных: как видно из рисунка, гистограммы двух повторных опытов с малым промежутком времени на ПК с ПОАВ статистически неоднородны.

Аналогичные расхождения между гистограммами повторных опытов наблюдаются и в случаях отсутствия ПОАВ.

Ещё бóльшая неоднородность выполняемых в разные дни параллельных опытов свидетельствует о недостаточной воспроизводимости получаемых результатов.

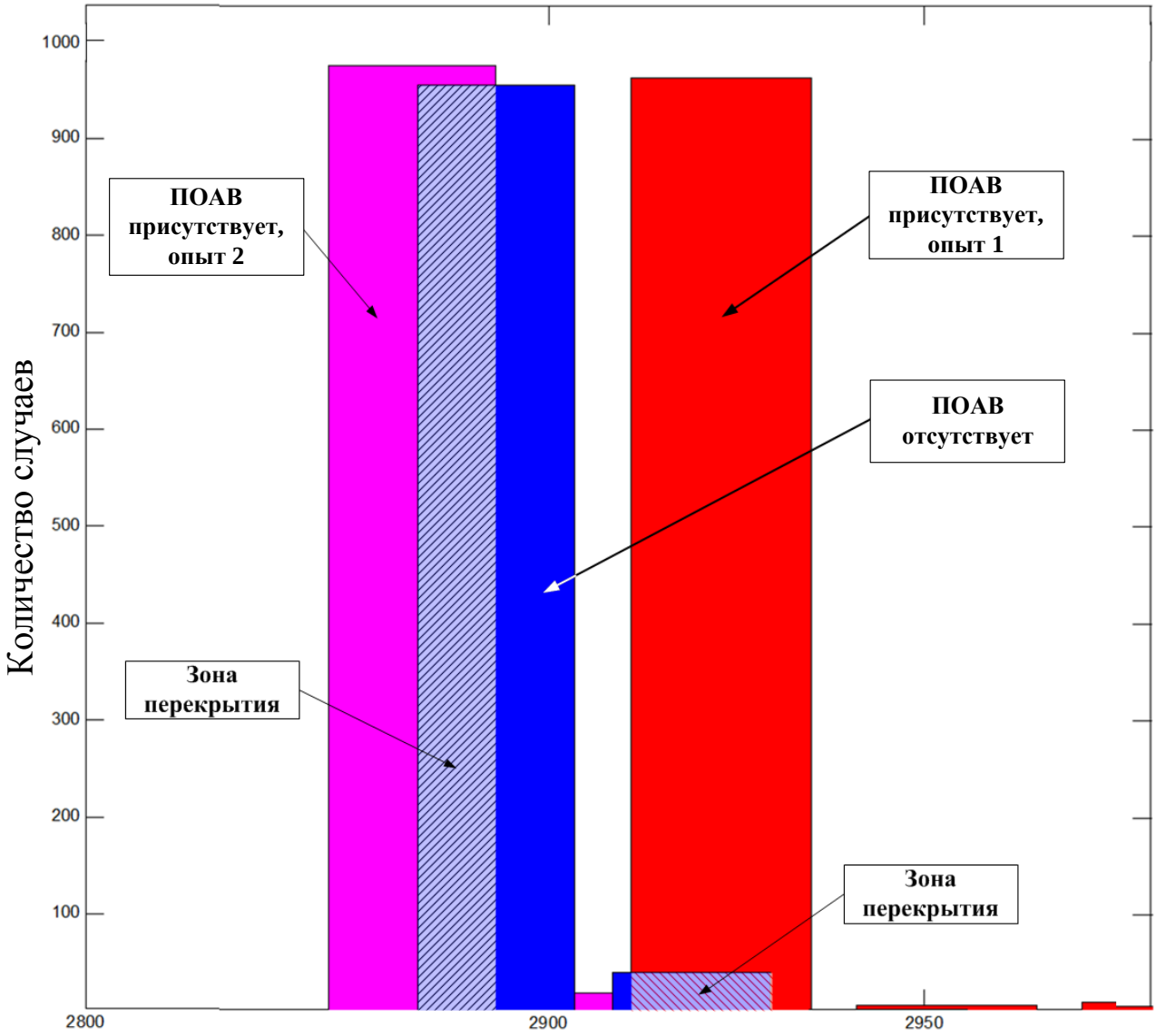


Рисунок 25 — Гистограммы длительности выполнения трассы по данным опытов Ит10R и Иг10V на ПК без ПОАВ и с искажением показаний счётчика тактов в случае присутствия ПОАВ

Поскольку неудовлетворительная прецизионность имеет постоянный и неустранимый характер, получаемые из опытов выборки длительности

выполнения трассы нельзя считать статистически представительными (репрезентативными) по ГОСТ Р ИСО 5725.

Из приведенного на рисунке также следует, что компрометация счётчика тактов с целью сокрытия ПОАВ путём уменьшения его показаний на некоторую постоянную величину однозначно не ведёт к точному совпадению выборочных средних длительности выполнения трассы в случаях отсутствия и присутствия ПОАВ: гистограммы в случае присутствия ПОАВ повторных опытов могут находиться как справа, так и слева на некотором удалении от гистограмм в случае отсутствия ПОАВ. Из этого следует, что в силу случайного характера длительности выполнения трассы у нарушителя нет гарантированной возможности для достаточно точного искажения показаний счётчика тактов.

Однако для сокрытия ПОАВ по признаку совпадения выборочных средних в случаях присутствия и отсутствия ПОАВ этого и не требуется, поскольку интервалы варьирования этих статистик перекрывают друг друга.

Аналогично и гистограммы в случаях отсутствия и присутствия ПОАВ могут существенно перекрывать друг друга даже при не очень тщательном искажении показаний счётчика тактов, что в большинстве случаев также препятствует использованию стандартных статистических методов для выявления ПОАВ в системах.

Из изложенного следует, что

- получаемые из опытов данные не обладают достаточной прецизионностью, вследствие чего выборки длительности выполнения трассы в статистическом смысле нельзя считать репрезентативными;
- распределение длительности выполнения трассы не подчиняется нормальному закону;
- гистограммы длительности выполнения трассы в случае отсутствия и присутствия ПОАВ с целенаправленным искажением показаний счётчика могут существенным образом перекрывать друг друга.

В этих условиях применение традиционных выборочных методов математической статистики (параметрической или непараметрической) было бы неправомерным.

Из изложенного следует, что при разработке алгоритмов обнаружения ПОАВ в системах следует учитывать, что выявление ПОАВ по выборочным средним длительности выполнения трассы всё же возможно, но только в тех случаях, когда искажение нарушителем показаний счётчика тактов выполняется недостаточно тщательно.

В общем же случае для обнаружения ПОАВ необходимы такие статистики, которые позволяли бы выявлять ПОАВ вне зависимости от степени искажения показаний счётчика. В связи с этим в данной главе исследовались возможности альтернативных статистик, характеризующих вариабельность длительности выполнения трассы, таких как дисперсия и её интервалы варьирования, вариационные ряды, полигоны и интервалы варьирования их длины, а также зависимости распределения длительности выполнения трассы по фиксированным уровням от случая присутствия и отсутствия ПОАВ и др.

Для преодоления проблемы недостаточной прецизионности результатов измерений опыты соответствующим образом организовывались и использовались специально разработанные приёмы обработки статистического материала.

Экспериментальные исследования выполнялись в 2 этапа. На первом этапе осуществлялась экспериментальная проверка выводов из анализа полученных моделей функционирования процессора (частично об этом сказано в главе 2). Выявлялись особенности длительности выполнения трассы в случаях отсутствия и присутствия ПОАВ, с учётом которых затем определялись подходы к обработке экспериментальных данных.

На втором этапе осуществлялся поиск статистических показателей длительности выполнения трассы, пригодных в качестве индикационных признаков для обнаружения ПОАВ.

Эксперименты осуществлялись по схеме однофакторных опытов с учётом разрабатываемой методики обнаружения нелегитимного ПОАВ. Варьируемым фактором была операционная среда ЭВМ с 2-мя качественными уровнями: отсутствие и присутствие ПОАВ. Результирующими признаками были различные статистики длительности выполнения трассы.

На основе выполненных исследований предложена методика обнаружения одного и нескольких образцов ПОАВ.

3.1 Организация проведения опытов по измерению длительности выполнения набора процессорных инструкций

В соответствии с результатами предварительных опытов разработка методов обнаружения ПОАВ базировалась на использовании различия статистических свойств длительности выполнения трассы – набора из 10-ти БП-инструкций CPUID при повышенном уровне приоритета в случае отсутствия (присутствия) ПОАВ [108]. Методика определения длительности выполнения трассы в случаях отсутствия t_{OT} и присутствия ПОАВ $t_{ПР}$ с компрометацией счётчика тактов изложена в подразделе 2.1.2.

Для минимизации внесения хаотичности в характеристики распределения длительности $t_{ПР}$ использовался авторский ПОАВ, разработанный на базе программного образца [88], содержащий минимальный набор инструкций и обеспечивающий компрометацию счётчиков тактов. Это наиболее сложный из возможных на практике случай для обнаружения ПОАВ.

Компрометация процессорного счётчика тактов осуществлялась образцом ПОАВ так, чтобы средние многократных измерений длительности выполнения трассы $\bar{t}_{OT}$ $и$ $\bar{t}_{ПР}$ в случае отсутствия (присутствия) ПОАВ были практически не различимыми.

Следует при этом заметить, что, как показывает практика, регистрируемые результаты измерения длительности выполнения трассы не всегда стабильны. Так, опыты подтвердили вывод из анализа моделей (3) – (6) о нестабильности длительности выполнения трассы: при неизменных условиях

опытов и величины компрометации счётчика δ длительность выполнения трассы $t_{ПР}$ в разные дни может очень сильно различаться и даже иметь отрицательные значения (наблюдается «дрейф данных»).

Тем самым подтверждена содержащаяся в главе 2 рекомендация, что тестирование ЭМВ на наличие ПОАВ следует начинать с проверки тестовых массивов длительности выполнения трассы $T_{тест}$ на наличие в них отрицательных значений и определения, насколько средняя длительность выполнения трассы $\bar{t}_{тест}$ отличается от нормативного $[t_{ОТ}]$ для случая отсутствия ПОАВ.

Длительность выполнения трассы измерялась с использованием процессорных инструкций RDTSC, считывающих показания процессорного счётчика тактов TSC до и после выполнения трассы (обоснование выбора данного счётчика и фрагменты программы приведены в подразделах 1.2.6 и 2.1.2). При этом во избежание возможных искажений по системным причинам регистрация показаний счётчика тактов выполнялась с учётом механизма сериализации [5], [96], [109].

Выполнение трассы и измерения длительности $t_{ОТ}$ и $t_{ПР}$ осуществлялись многократно с 2-х секундной паузой между вложенными циклами. Результатом опытов являлись матрицы $T_{ОТ}$ и $T_{ПР}$ размером 1000x10, столбцами которых были данные измерений длительности выполнения трассы, полученные при выполнении вложенных циклов.

С учётом метрологического ГОСТ Р ИСО 5725 [9] для оценки сходимости опыты выполнялись сериями из 5-ти повторных опытов с небольшим интервалом между ними. Оценка воспроизводимости производилась по серии повторных опытов, выполняемых в течение 10-ти дней. Опыты показали, что такого периода достаточно для стабилизации наблюдаемых границ интервалов варьирования определяемых статистик.

Всего было обследовано шесть ЭВМ с разными процессорами. Их модели, версии ОС, а также коды опытов приведены в табл. 2.

В первых пяти ЭВМ использовался разработанный автором образец ПОАВ, реализованный в виде драйвера ОС, в шестом ЭВМ – специализированный образец ПОАВ, получающий управление при загрузке ЭВМ из BIOS.

Таблица 2 – Модели процессоров и ОС на ЭВМ, использованных в опытах

№ ЭВМ	**Модель процессора и установленная ОС**	**Код серии опытов**
1	Intel Core 2 Duo E6300 с Windows 7	Ал
2	Intel Core 2 Duo E8200 с Windows 7	Иг
3	Intel Core 2 Duo E8600 с Windows Live CD XP	Нии
4	Intel Core i7 950 с Windows XP	НН
5	Intel Xeon X5600 с Windows 7	HPW
6	AMD Phenom X4 945 с Windows Live CD XP	AMD

3.2 Особенности процесса выполнения набора процессорных инструкций и обоснование подходов к обработке экспериментальных данных

Исследования подтвердили, что измеряемая с помощью процессорного счётчика тактов TSC длительность выполнения различного набора процессорных инструкций является случайной величиной, статистические свойства которой зависят как от модели процессора и версии ОС, так и от состояния операционной среды ЭВМ – в случае присутствия ПОАВ величина и вариабельность длительности выполнения трассы $t_{ПР}$ существенно больше, чем в случае отсутствия ПОАВ $t_{ОТ}$. Причём различие статистик $t_{ОТ}$ и $t_{ПР}$ возрастает, если в ЭВМ присутствует не один, а два и более *вложенных* образцов ПОАВ.

Поэтому, если показания счётчика тактов целенаправленно не искажаются, определить по средним арифметическим $\bar{t}_{ОТ}$ и $\bar{t}_{ПР}$ массивов

длительности выполнения трассы $T_{ОТ}$ и $T_{ПР}$, присутствует ли ПОАВ в ЭВМ или нет, не составляет труда [55].

Однако технология аппаратной виртуализации процессоров Intel и AMD предоставляет штатные средства для искажения показаний процессорного счётчика тактов, так что разница между арифметическими средними получаемых массивов длительности выполнения трассы $T_{ОТ}$ и $T_{ПР}$ может стать практически неразличимой. В таких случаях для обнаружения присутствия ПОАВ требуются сведения о других статистиках длительности выполнения трассы в случаях отсутствия (присутствия) одного образца (нескольких образцов) ПОАВ.

Для их выявления наблюдения должны быть массовыми и их можно организовать по-разному.

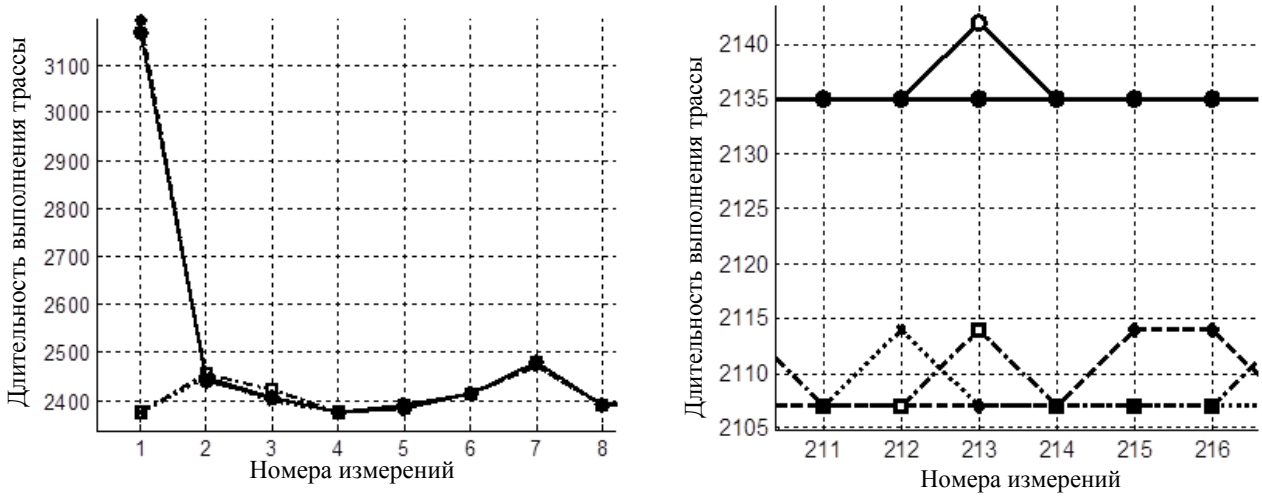
Если исходить из характера получаемых данных, наиболее естественной была бы такая постановка опытов, при которой выполнение трассы и измерение длительности $t_{ОТ}$ и $t_{ПР}$ рассматривались бы как *случайные процессы (случайные функции)* $T_{ОТ}(i)$ и $T_{ПР}(i)$, где i – номер измерения в последовательности повторных измерений, выполняемых во вложенном цикле.

Типичные реализации этих процессов представлены на рис. 26. Характерными особенностями являются выбросы при первых измерениях в начале каждого из вложенных циклов (рис. 26 *а*).

Причём, несмотря на компрометацию счётчика тактов в случае присутствия ПОАВ, такие выбросы чаще всего существенно больше, чем в случае его отсутствия. Далее процесс приобретает стабильный характер, но иногда переходный период занимает 10-12 последовательных измерений.

На стабилизированном участке (рис. 26 *б*), как и следует из анализа модельных формул главы 2, наблюдается случайное распределение точек длительности выполнения трассы $t_{ОТ}$ по нескольким фиксированным близко расположенным уровням, что придаёт процессу измерений длительности выполнения трассы *слоистый* (*уровневый*) характер.

Аналогичный характер имеют графики реализаций процесса измерений длительности выполнения трассы $t_{ПР}$ в случае присутствия ПОАВ с компрометацией процессорного счётчика тактов. Отличие в том, что количество уровней, по которым случайным образом распределяется длительность выполнения трассы $t_{ПР}$, может быть бóльшим. Другое отличие – уровни $t_{ПР}$ при тщательной компрометации счётчика тактов могут располагаться вблизи с уровнями $t_{ОТ}$, но, как правило, с ними не совпадать.



а *б*

Рисунок 26 — Графики семейства 5-ти реализаций случайного процесса измерения длительности выполнения трассы $t_{ОТ}$ в опыте Иг2 на ЭВМ под управлением процессора Intel Core 2 Duo E8200 с ОС Windows 7 в случае отсутствия ПОАВ: *а* – на начальном и *б* – на последующем этапах процесса измерений

Представленные результаты получены на всех 6-ти обследованных ЭВМ, а также в зарубежных опытах [95]. Заметим, что найденные особенности проявляются также при выполнении не только инструкций CPUID, но и других БП-инструкций.

Таким образом, эксперименты подтверждают основной вывод из анализа предложенной модели функционирования процессора при выполнении измерений длительности выполнения трассы: в случаях отсутствия и присутствия ПОАВ измерение длительности выполнения трассы представляет собой случайный процесс, отличающийся некоторой упорядоченностью:

измеряемая длительность выполнения трассы в цикле распределяется по фиксированным уровням случайным образом.

Подчеркнём, что обнаруженная упорядоченность проявляется в распределении значений длительности выполнения трассы $t_{ОТ}$ и $t_{ПР}$ по детерминированным уровням, а хаотичность – в том, что распределения точек на этих уровнях случайны: попадания результатов измерений $t_{ОТ}$ и $t_{ПР}$ на тот или иной уровень являются случайными событиями, распределение которых близко к биномиальному закону. Причём эта случайность от реализации к реализации процесса может проявляться по-разному, из-за чего математические ожидания случайных процессов $\bar{t}_{ОТ}(i)$ и $\bar{t}_{ПР}(i)$ *нестабильны*: от опыта к опыту они *дрейфуют*. Поэтому в случаях достаточно тщательного искажения показаний счётчика тактов сравнением $\bar{t}_{ОТ}(i)$ и $\bar{t}_{ПР}(i)$ выявить показатели присутствия ПОАВ не представилось возможным.

Кроме отмеченных неоднородностей, в получаемых рядах значений $t_{ОТ}$ и $t_{ПР}$ наблюдаются выбросы и разрывы (скачки), в результате чего структура процесса изменяется. Эти особенности рассмотрены далее.

Из изложенного следует, что выполнение трассы в цикле – это *нестационарный случайный процесс, часто с нестабильной структурой*, характер которой случайным образом меняется от реализации к реализации [8].

Предпринятые попытки свести наблюдаемые случайные процессы к стационарным путём разделения их траекторий на однородные участки и весового усреднения характеристик оказались безуспешными. Причина – сильная вариабельность и нестабильность дисперсии и корреляционной функции случайного процесса при повторных опытах.

В силу этого получить идентификационные показатели присутствия ПОАВ на базе характеристик случайного процесса выполнения трассы (математического ожидания, дисперсии и корреляционной функции) оказалось затруднительным. Поэтому при обработке экспериментальных данных использовались иные подходы.

Наиболее перспективными оказались подходы, учитывающие уровневый характер распределения длительности выполнения трассы. Самый простой из них основан на учёте того факта, что *уровни* длительности выполнения трассы t_{OT} в случае отсутствия ПОАВ в ЭВМ после частотной фильтрации с уровнем $f = 0{,}1$ представляют компактные детерминированные множества. Это позволяет принять их в качестве стандартных (или нормативных) $[U_{\text{ОТ}}]$, свидетельствующих об отсутствии ПОАВ, сравнивая с которыми тестовые множества уровней длительности выполнения трассы $U_{тест}$, можно однозначно устанавливать наличие или отсутствие ПОАВ в обследуемых ЭВМ.

Другой подход к обработке получаемого статистического материала состоит в использовании *методов вариационных рядов*. Однако, как показал опыт, традиционное их применение оказалось проблематичным не только из-за их нестабильности, но ещё и потому, что интервалы варьирования получаемых показателей вариационных рядов t_{OT} и $t_{ПР}$ перекрываются настолько, что учесть это при обнаружении ПОАВ очень сложно. Далее будет показано, как преодолевается эта проблема.

Также возможно использование *выборочных методов математической статистики* при условии, что сами опыты и обработка их результатов проводятся с учётом недостаточной прецизионности получаемых данных [9].

Как уже отмечалось, проблема состоит в том, что сходимость (повторяемость) результатов измерений, выполняемых сериями с небольшим интервалом между опытами, и, в особенности, воспроизводимость их по дням выполнения опытов недостаточны. Из-за дрейфа экспериментальных данных использовать традиционные оценочные методы математической статистики не представлялось возможным. Поэтому интервальная оценка статистик осуществлялась упрощённо – по методу Корнфельда [27], [39], согласно которому в качестве доверительных границ интервала варьирования получаемой из опыта статистики S принимались минимальное и максимальное

значения S_{min} и S_{max} из получаемого ряда статистики S при доверительной вероятности, вычисляемой по формуле

$$\alpha = 1 - \left(\frac{1}{2}\right)^{l-1}, \tag{20}$$

где l длина ряда получаемых значений статистики S.

Поскольку во всех опытах $l = 10$, то $\alpha = 0{,}998 \approx 1$. Это излишне высокая надежность. В подобных случаях в целях уменьшения погрешности Корнфельд М. И. рекомендует укорачивать обрабатываемый ряд путём отбрасывания его конечных значений, после чего доверительную вероятность пересчитывать [18]. Далее показано, как учитывались эти рекомендации.

Другая проблема – наличие выбросов и разрывов (скачков) в наблюдаемых рядах данных t_{OT} и $t_{ПР}$ (рис. 27), существенно влияющих на выборочные характеристики [49]. Поэтому, если не удалять выбросы и не учитывать разрывы, то устойчивого различия между статистиками в случаях отсутствия и присутствия ПОАВ получить не удаётся.

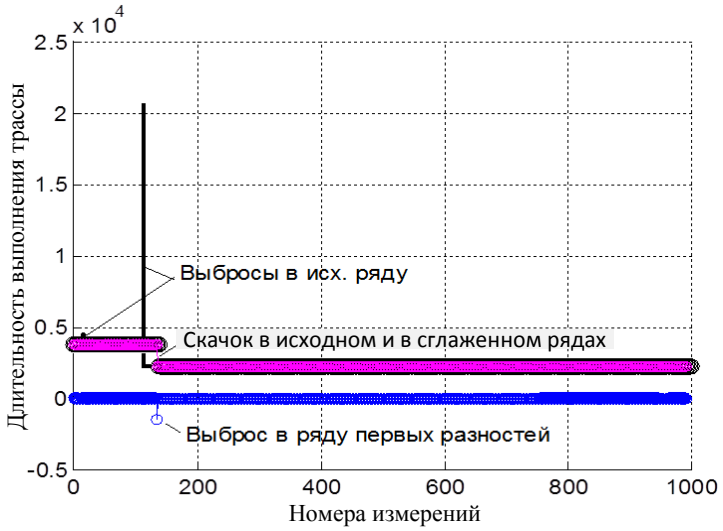


Рисунок 27 — Графики ряда длительности выполнения трассы t_{OT}: исходного, сглаженного с помощью медианного фильтра и первых разностей сглаженного ряда в опыте НИИ1 в случае отсутствия ПОАВ

Отрицательное влияние отмечаемых неоднородностей устранялось сглаживанием [36], [49] и (или) селекцией [54] путём фильтрации полученных данных.

Исследования показали, что проблемы неоднородности наблюдаемых данных решаются путём выполнения следующих условий:

- массивы экспериментальных данных длительности выполнения трассы $T_{ОТ}$ и $T_{ПР}$ получают во вложенном цикле в виде матриц, столбцы которых рассматриваются как статистические выборки. Эти матрицы должны быть достаточно большими, например, размером 1000x10, и их количество в сериях из повторных опытов должно быть не менее 5-ти;
- серии повторных опытов выполняются в течение длительного периода времени (порядка 10-ти дней);
- для устранения влияния выбросов и разрывов в наблюдаемых рядах данные подвергаются предварительной обработке – фильтрации или сглаживанию с весовым усреднением статистик;
- разработка пороговых значений идентификационных признаков производится на базе таких статистик $t_{ОТ}$ и $t_{ПР}$, интервалы варьирования которых не перекрываются, или перекрываются так, что частота таких событий достаточно мала.

Преимущество выборочных методов состоит в возможности их адаптации к особенностям получаемых из опыта массивов длительности выполнения трассы $T_{ОТ}$ и $T_{ПР}$. Так, применением низкочастотных фильтров из этих массивов можно удалять выбросы, а влияние разрывов (скачков) – устранять определением статистик на участках между разрывами с последующим весовым усреднением для всего ряда наблюдений.

Особого подхода требует выбор типа статистик. Так, по причине пересечения множеств дисперсий длительности выполнения трассы $D_{ОТ}$ и $D_{ПР}$, получаемых в повторных опытах в случаях отсутствия и присутствия ПОАВ, традиционное использование дисперсии в качестве меры осцилляции случайной величины для большинства обследованных ЭВМ оказалось неподходящим.

В связи с этим показатели распределения определялись только после предварительной обработки первичных опытных данных, после чего получали ряды тех или иных статистик, по которым затем находили показатели распределения полученных статистик.

Иными словами, из первичных статистик S_{OT} и $S_{ПР}$ столбцов матриц T_{OT} и $T_{ПР}$ получали вторичные статистики − их «производные» S'_{OT} и $S'_{ПР}$. Практическая их значимость в том, что пересечения множеств S'_{OT} и $S'_{ПР}$ по сравнению с исходными сужаются до нескольких единиц, что упрощает оценку вероятности попадания индивидуальных значений определяемых статистик в эти пересечения, а, значит, это можно учесть при тестировании ЭВМ на наличие в них ПОАВ.

Исследования показали перспективность предложенных комбинированных показателей распределения случайных величин длительности выполнения трассы.

Отметим также, что в подразделе 2.2.1 при анализе работы модели процессора было установлено, что смена режимов работы процессора происходит случайным образом, вследствие чего в случае присутствия ПОАВ результаты измерения длительности выполнения трассы в условиях компрометации счётчика тактов могут быть нестабильными. Исследования подтвердили это: при неизменных входных параметрах ПОАВ с компрометацией счётчика тактов длительность выполнения трассы $t_{ПР}$ как по ходу опыта, так и от опыта к опыту может изменяться настолько, что оценки математического ожидания $\bar{t}_{ПР}$ будут существенно отличаться от оценок математического ожидания длительности выполнения трассы $\bar{t}_{OT}$ и даже приобретать отрицательные значения.

Отсюда следует, что тестирование ЭВМ на наличие в них ПОАВ следует начинать с сопоставления тестового среднего длительности выполнения трассы $\bar{t}_{тест}$ и порогового среднего для случая отсутствия ПОАВ $[\bar{t}_{OT}]$. Существенное их отличие, а также отрицательные значения в получаемых рядах измерений

длительности выполнения трассы $t_{тест}$ будут свидетельствовать о наличии ПОАВ в тестируемой ЭВМ.

Анализ показал, что выявить пороговые значения статистик длительности выполнения трассы в случаях *вложенных* ПОАВ можно путём поэтапного обследования ЭВМ: сначала определить, как описано выше, пороговые значения статистик длительности выполнения трассы в случае отсутствия ПОАВ – S_1, затем в ЭВМ с авторским ПОАВ установить систему Acronis [57] со встроенным ПОАВ или иной легитимный образец ПОАВ, после чего по аналогичной методике определить пороговые значения статистик S_2 для нового случая. Далее описанная процедура повторяется.

Для пояснения изложенного на рис. 28 показана числовая ось с указанием порогов идентификационной статистики $S_1, S_2, \dots S_n$ вместе с разделяемыми ими случаями отсутствия и присутствия ПОАВ.

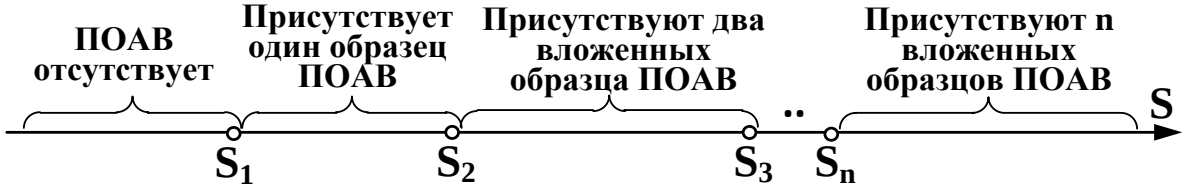


Рисунок 28 — Пороговые величины показателей присутствия одного, двух и более образцов ПОАВ

Из рисунка вытекает следующий порядок использования полученных пороговых значений. Если при тестировании ЭВМ окажется, что тестовая статистика $S_{тест} \le S_1$, то в ЭВМ отсутствует ПОАВ. При $S_1 < S_{тест} \le S_2$ в системе только один, ранее легитимно установленный образец ПОАВ. Если $S_{тест} > S_2$, то принимается решение, что в тестируемой ЭВМ присутствует два вложенных образца ПОАВ, из которых один нелегальный.

Указанный порядок обнаружения ПОАВ сохраняется и при $S_{тест} > S_i,\ i = 3, \dots, n$ при условии, что нелегальные образцы ПОАВ устанавливаются вне промежутков времени, когда выполняется определение пороговых значений статистик для легитимных образцов ПОАВ. Сведения о пороговых условиях и принимаемых решениях о наличии ПОАВ даны в табл. 3.

Альтернативным подходом к выявлению вложенных ПОАВ может быть одновременное использование двух методов обнаружения одного ПОАВ: на основе длины вариационных рядов и механизмов кэширования [17].

Таблица 3 – Пороговые условия для обнаружения ПОАВ

Пороговые условия	Принимаемое решение о наличии ПОАВ
$S \leq S_1$	ПОАВ отсутствует
$S \in (S_1, S_2]$	присутствует один ПОАВ из установленного ПО
$S \in (S_2, S_3]$	два вложенных ПОАВ из установленного ПО
...	…
$S \in (S_{n-1}, S_n]$	$(n-1)$ вложенных ПОАВ из установленного ПО
$S > S_n$	n-вложенных ПОАВ, в том числе $(n-1)$ – из установленного ПО и один нелегальный ПОАВ

Реализация предложенных подходов к решению поставленной задачи представляется далее.

3.3 Обработка опытных данных и выявление признаков для обнаружения программного обеспечения, использующего технологию аппаратной виртуализации

С учётом изложенного выше использовалась следующая методика предварительной обработки первичных данных длительности выполнения трассы.

Для удаления выбросов, а также в исследовательских целях получаемые из опытов матрицы T_{OT} и $T_{ПР}$ подвергались фильтрации по столбцам с 6-ю назначаемыми уровнями фильтрации относительных частот длительности выполнения трассы $f \in \{0;\ 0{,}02;\ 0{,}05;\ 0{,}1;\ 0{,}15;\ 0{,}2\}$. Полученные отфильтрованные массивы $T_{OT,f}$ и $T_{ПР,f}$ проверяли на наличие разрывов (скачков) по изложенной в [49] методике. При этом абсцисса скачка определялась как максимальное значение ряда первых разностей отфильтрованных значений длительности выполнения трассы $t_{OT,f}$ и $t_{ПР,f}$, а в качестве порогового значения ординаты скачка принималось 300 тактов.

Заметим, что эта величина может корректироваться с учётом особенностей экспериментальных данных.

Из всех отфильтрованных массивов $T_{OT,f}$ и $T_{ПР,f}$ для дальнейшей обработки отбирали те, у которых при наименьшем уровне фильтрации анализируемые статистики длительности выполнения трассы начинали приобретать стабильный характер. Выяснилось, что у всех 6-и обследованных ЭВМ это начинало происходить при $[f] = 0{,}1$.

Методика дальнейшей обработки опытных данных определялась 2-мя главными особенностями длительности выполнения трассы: детерминированностью уровней распределения значений длительности выполнения трассы $t_{OT,f}$ и $t_{ПР,f}$ и случайностью их попадания на эти уровни. В соответствии с этим использовались два метода выявления идентификационных признаков наличия ПОАВ – уровневый и метод выборочных статистик.

При использовании *уровневого метода* в столбцах отфильтрованных массивов $T_{OT,f=0,1}$ и $T_{ПР,f=0,1}$ определялись по всем сериям опытов значения уровней длительности выполнения трассы, из которых для каждого из обследованных ЭВМ составлялись простые статистические ряды значений вариант (уровней) U_{OT} и $U_{ПР}$.

Так, например, в сериях опытов Иг10R10–Иг11R11 получен обобщённый ряд U_{OT} = 2160, 2168, 2184, 2192 ,2200, 2478 2480, 2880, 2888, 2904, 2920, 2936 тактов, а в серии опытов Иг10V10 получен ряд $U_{ПР}$ = 2876, 2884, 2892, 2900, 2908, 2916, 2924 тактов. Из сравнения этих рядов следует, что среди их элементов нет ни одного совпадающего.

В других сериях повторных опытов Иг в случае присутствия ПОАВ, выполненных в разные дни, получены аналогичные результаты за тем исключением, что в рядах уровней длительности выполнения трассы изредка встречаются отрицательные значения.

Как показали опыты, на разных ЭВМ ряды U_{OT} и $U_{ПР}$ не пересекаются за исключением ЭВМ с процессором Intel Core 2 Duo E6300 с Windows 7, в рядах которого в случае присутствия ПОАВ встречались совпадающие с рядом U_{OT} значения. Однако частота таких событий крайне мала. Так что, если при тестировании данной ЭВМ выполнять не один, а несколько повторных опытов, то вероятность повторного появления данного события с учётом теоремы умножения вероятностей для повторных событий будет пренебрежимо малой.

Опыты свидетельствуют, что если обобщённые ряды уровней длительности выполнения трассы U_{OT} получены из достаточного объёма статистического материала и в разные дни, то, поскольку такие ряды практически детерминированные и не пересекаются или пересекаются очень редко, их можно принять в качестве нормативных $[U_{OT}]$, свидетельствующих об отсутствии ПОАВ.

Исследования показали, что уровневый метод обнаружения ПОАВ достаточно прост. Однако не исключено, что на некоторых ЭВМ этим методом не обнаружатся образцы ПОАВ и потребуются альтернативные методы, основанные на использовании различий статистических свойств длительности выполнения трассы t_{OT} и $t_{ПР}$.

Для разработки таких методов исследовались выборочные статистики массивов $T_{OT,f}$ и $T_{ПР,f}$: средних $\bar{t}_{OT,f=0}$ и $\bar{t}_{ПР,f=0}$, дисперсий $d_{OT,f}$ и $d_{ПР,f}$, моментов 4-го порядка $\mu_{OT,f}$ и $\mu_{ПР,f}$ и длины статистических (вариационных) рядов длительности выполнения трассы отфильтрованных столбцов $l_{OT,f}$ и $l_{ПР,f}$. В случаях отсутствия разрывов в столбцах массивов эти статистики определялись для каждого столбца непосредственно, при обнаружении разрывов статистики – для участков между точками разрывов, которые затем усреднялись с учётом весов, равных длине этих участков.

В приведенной в качестве иллюстрации табл. 4 в столбцах с 2-го по 11-ый помещены значения длины $l_{ПР,f}$ вариационных рядов, полученных по столбцам матрицы $T_{ПР,f}$ в опыте на ЭВМ под управлением процессора Intel Core 2 Duo

E8200 с ОС Windows 7 в случае присутствия ПОАВ. Усреднённые значения этих статистик находятся в 12-ом столбце.

Таблица 4 – Статистики длины $l_{ПР,f}$ вариационных рядов длительности выполнения трассы по столбцам матрицы $T_{ПР,f}$ и вектору $T_{в,ПР,f}$ из опыта Иг10V4 в случае присутствия ПОАВ

Уровень фильтрации f	**№ столбцов матрицы $T_{ПР,f}$**				**$\bar{l}_{ПР,f}$ по столбцам матрицы $T_{ПР,f}$**	**$l_{в,ПР,f}$ по вектору $T_{в,ПР,f}$**
	1	**2**	**…**	**10**		
1	2	3	…	11	12	13
0	28	29	…	10	12	53
0,02	4	3	…	3	4	6
0,05	3	3	…	3	3	6
$[f]$= 0,1	2	2	…	3	3	3
0,15	1	2	…	2	2	3
0,20	1	2	…	2	2	2

В последнем столбце помещены значения длины $l_{в,ПР,f}$ вариационных рядов векторов $T_{в,ПР,f}$, полученных путем векторизации данных столбцов массивов $T_{ПР,f}$.

Анализ табл. 4 показал, что, начиная с уровня фильтрации $[f]$= 0,1, значения статистик $\bar{l}_{ПР,f}$ и $l_{в,ПР,f}$ стабилизируются.

Для дальнейшего рассмотрения оставляли статистики, полученные для $f = 0{,}1$ или те, для которых сумма вероятностей ошибок I и II рода была минимальной (смысл этих ошибок рассматривается далее). Их значения для всех 10-ти опытов на данной ЭВМ сведены в сводную табл. 5.

Границы интервалов варьирования статистик $\bar{l}_{ОТ,f}$ и $\bar{l}_{ПР,f}$, $l_{в,ОТ,f}$ и $l_{в,ПР,f}$, помещённые в 4-ую снизу строку табл. 5, определялись по минимальному и максимальному значениям статистик соответствующих столбцов. Поскольку интервалы варьирования этих статистик в случаях отсутствия и присутствия ПОАВ могут перекрываться, сохраняется вероятность того, что при тестировании данной ЭВМ получаемые значения $\bar{l}_{тест}$ и $l_{в,тест}$ попадут в область перекрытия. В таких ситуациях однозначно обнаружить ПОАВ будет невозможно.

Таблица 5 – Статистики распределения длины вариационных рядов длительности выполнения трассы по данным опытов Иг10 – Иг19 при уровне фильтрации $f = 0{,}1$ (фрагмент таблицы)

Код серии опытов	**ПОАВ отсутствует**		**ПОАВ присутствует**	
	$\bar{l}_{OT,f}$ **по столбцам матрицы** $T_{OT,f}$	$l_{в,OT,f}$ **по вектору** $T_{в,OT,f}$	$\bar{l}_{ПР,f}$ **по столбцам матрицы** $T_{ПР,f}$	$l_{в,ПР,f}$ **по вектору** $T_{в,ПР,f}$
1	2	3	4	5
Иг10	5	23	11	47
	4	18	11	52
	4	15	10	34
	5	21	13	53
	4	15	14	68
…	…	…	…	…
Иг19	4	20	19	102
	6	32	15	77
	6	32	16	79
	6	32	20	88
	10	50	21	105
Интервал варьирования статистик	[4, 14]	[10, 110]	[8, 21]	[29,105]
Пороговое условие $[P_{\text{ОТ}}]$ и $[P_{\text{ПР}}]$	≤ 7	≤ 32	≥ 8	≤ 33
Вероятность ошибки I рода, α	0,04	0,12	–	–
Вероятность ошибки II рода, β	0	0,16	–	–

Во избежание этого подсчитывалась вероятность того, что статистики $\bar{l}_{OT,f}$ и $\bar{l}_{ПР,f}$, $l_{в,OT,f}$ и $l_{в,ПР,f}$ не попадут в зоны перекрытия. Для этого на предварительном этапе работы по столбцам 2 и 4, 3 и 5 табл. 5 определялись относительные частоты возможного попадания значений этих статистик в зону перекрытия, несколько расширенную за счёт искусственного сужения интервалов варьирования рассматриваемых статистик. По этим частотам оценивались вероятности статистических ошибок I и II α и β.

При этом под ошибкой I рода принималась ситуация, когда тест-статистика показывает на наличие ПОАВ, а в действительности его нет, а под

ошибкой II рода – ситуация, когда ПОАВ присутствует в ЭВМ, а тест-статистика показывает, что оно отсутствует. Оценить вероятности α и β, например, для статистики $\bar{l}_{OT,f}$ можно по отношению r/g, где r – число значений из столбца 2 табл. 5, которые выходят за пределы принятого порогового значения – новой допустимой верхней границы интервала варьирования $\bar{l}_{OT,f}$, $g = 50$ – общее число значений в столбце.

Пороговые значения $[\bar{l}_{OT,f}]$ и $[\bar{l}_{ПР,f}]$ выявляются интерактивно из условия, чтобы сумма вероятностей α и β ошибок I и II рода была бы минимальной. Значения $[\bar{l}_{OT,f}]$ и $[\bar{l}_{ПР,f}]$, α и β помещены в 3-х последних строках табл. 5. Аналогичным образом определялись пороговые значения и для других статистик. В качестве тест-статистик для обнаружения ПОАВ выбирались только те статистики, у которых сумма вероятностей ошибок I и II рода не превышает 0,2.

Описанная методика обнаружения ПОАВ использовалась при обследовании всех ЭВМ, указанных в табл. 2. Полученные для этих ЭВМ идентификационные признаки и их пороговые условия представлены в табл. 6.

Из табл. 6 следует, что у большинства обследованных ЭВМ вероятность ошибок I и II рода менее 5%. Для некоторых из них вероятности этих ошибок оказались большими. Однако, если при тестировании ЭВМ выполнять не один, а несколько повторных опытов, и по получаемым данным определять частоту выполнения тест-статистиками пороговых условий, то с учётом теоремы умножения вероятностей для повторных событий вероятности этих ошибок станут пренебрежимо малыми.

Таблица 6 – Пороговые условия тест-статистик длительности выполнения трассы для ЭВМ, указанных в табл. 2 ($\bar{T}_{OT,f}$ – средняя длительность выполнения трассы для всей матрицы $T_{OT,f}$; $\bar{L}_f$ – средняя длина вариационных рядов, $\bar{D}_f$ – средняя дисперсия и $\bar{M}_f$ – средний момент 4-го порядка столбцов матриц $T_{OT,f}$ и $T_{ПР,f}$; d_f и μ_f – дисперсия и момент 4-го порядка векторов $T_{в,OT,f}$ и $T_{в,ПР,f}$)

ЭВМ	Тест-статистика	Уровень фильтрации f	Пороговое значение		Вероятность ошибки	
			ПОАВ отсутствует	ПОАВ присутствует	I рода, α	II рода, β
1	$\bar{T}_{OT,f}$	0	≤ 2911	–	–	–
	$\bar{L}_f$	0	≤ 7	≥ 8	0.04	0
	$\bar{D}_f$	0	≤ 14	≥ 18	0.02	0
	$\bar{M}_f$	0.1	≤ 679	≥ 947	0.02	0
	μ_f	0.1	≤ 104161	≥ 111041	0.02	0.10
2	$\bar{T}_{OT,f}$	0	≤ 2492	–	–	–
	$\bar{L}_f$	0	≤ 11	≥ 12	0.1	0.06
	$\bar{D}_f$	0.2	≤ 100	≥ 101	0.08	0.1
	$\bar{M}_f$	0.2	≤ 168	≥ 13030	0.14	0.02
3	$\bar{T}_{OT,f}$	0	≤ 2431	–	–	–
	$\bar{L}_f$	0	≤ 6	≥ 8	0	0
	$\bar{D}_f$	0.1	≤ 15	≥ 41	0	0
	μ_f	0.1	≤ 609	≥ 3410	0	0
4	$\bar{T}_{OT,f}$	0	≤ 5018	–	–	–
	$\bar{L}_f$	0	≤ 22	≥ 26	0.02	0.02
	$\bar{D}_f$	0.1	≤ 177	≥ 181	0.1	0.1
5	$\bar{T}_{OT,f}$	0	≤ 2852	–	–	–
	$\bar{L}_f$	0	≤ 67	≥ 71	0.04	0
	$\bar{D}_f$	0	≤ 16416	≥ 48920	0	0
6	$\bar{T}_{OT,f}$	0	≤ 2126	–	–	–
	$\bar{L}_f$	0	≤ 34	≥ 241	0	0
	$\bar{l}_f$	0	≤ 134	≥ 593	0	0
	$\bar{D}_f$	0	≤ 216	≥ 5478	0	0
	d_f	0	≤ 345	≥ 5422	0	0
	$\bar{M}_f$	0,02	≤ 54	≥ 956	0	0

Поисковые эксперименты показали также, что наряду с предложенными статистиками для обнаружения ПОАВ можно использовать и другие показатели, такие как *координаты центра тяжести полигона* частот (ЦТП) –

вариант статистического (вариационного) ряда, а также длин вариационных рядов, строящихся по столбцам полученных из опытов массивов длительности выполнения трассы T_{OT} и $T_{ПР}$. Так, например, для массивов размером 1000x10 получается 10 таких вариационных рядов, из длин которых $l_{OT}(i)$ и $l_{ПР}(i)$, $i = 1, \dots, 10$ составляются векторы L_{OT} и $L_{ПР}$. По этим векторам строятся статистические ряды и полигоны частот, для которых, как для геометрических фигур, определяются полярные координаты ЦТП (ρ, φ), где ρ и φ – длина и угол радиуса [26].

Для иллюстрации на рис. 29 представлены примеры двух полигонов частот и их центры тяжести (ЦТ) длин L_{OT} и $L_{ПР}$ статистических рядов матриц T_{OT} и $T_{ПР}$, полученных на ЭВМ с процессором Intel Core 2 Duo E8600 с ОС Windows Live CD XP в случаях отсутствия и присутствия ПОАВ.

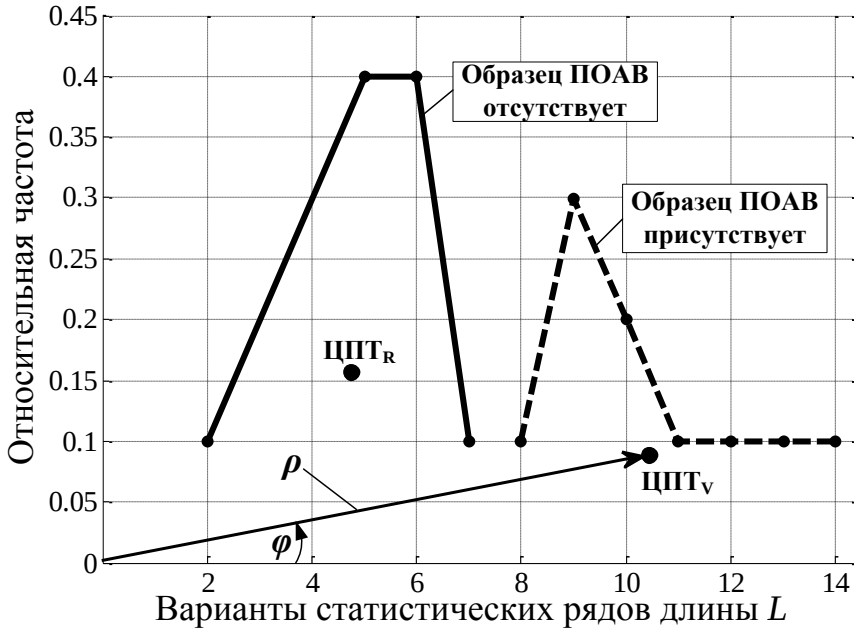


Рисунок 29 — Примеры полигонов частот длин L_{OT} и $L_{ПР}$ статистических рядов матриц T_{OT} и $T_{ПР}$ и их центры тяжести

На рисунке видно, что положение ЦТП существенно зависит от наличия или отсутствия ПОАВ, что позволяет принять координаты ЦТП ρ_{OT} и φ_{OT}, $\rho_{ПР}$ и $\varphi_{ПР}$ в качестве показателей для обнаружения ПОАВ. Их пороговые значения определяются по описанной выше методике за тем исключением, что

предварительную обработку получаемых из опытов матриц длительности выполнения трассы T_{OT} и $T_{ПР}$ можно не производить.

Исследования показали, что метод на базе ЦТП удовлетворительно работает на большинстве обследованных ЭВМ. Однако на ЭВМ с процессором Intel Core 2 Duo E6300 с ОС Windows 7 из-за большого объёма области пересечения множеств координат ЦТП в случаях отсутствия и присутствия ПОАВ этот метод оказался неработоспособным. Кроме того, при повторных опытах на некоторых ЭВМ могут выполняться альтернативные пороговые условия. Поэтому для обеспечения достаточной надёжности принимаемых решений при тестировании ЭВМ следует выполнять серию из нескольких повторных опытов и по полученным тест-матрицам $T_{тест}$ определять соотношение числа случаев выполнения альтернативных пороговых условий для координат ЦТП, которое затем сравнивать со своими пороговыми условиями.

Выполнением этой процедуры можно практически достоверно обнаруживать ПОАВ.

Для обнаружения ПОАВ пригодны также особенности неоднородностей в рядах наблюдаемых значений длительности выполнения трассы, такие как месторасположение разрывов в рядах измерений, величина выбросов, особенности начального участка ряда измерений и др. При этом следует учитывать, что эти особенности проявляются также случайным образом.

Экспериментальная проверка предложенных подходов к обнаружению *вложенных* ПОАВ осуществлялась по изложенной в разделе 3.2 методике. Полученные пороговые значения идентификационного показателя – средней длины вариационных рядов длительности выполнения трассы $\bar{L}_f$ для одной из обследованных ЭВМ представлены в табл. 7.

Таблица 7 – Пороговые условия средней длины вариационных рядов длительности выполнения трассы $\bar{L}_f$ на ЭВМ с процессором Intel Core i7 950 и с ОС Windows XP при уровне фильтрации f = 0

№	Пороговое значение	Решение о наличии ПОАВ
1	$\bar{L}_f \leq 31$	ПОАВ отсутствует
2	$32 \leq \bar{L}_f \leq 67$	Присутствует один ПОАВ Acronis
3	$\bar{L}_f \geq 86$	Присутствуют два вложенных ПОАВ: Acronis и нелегальный

Подсчёт экспериментальной вероятности показал, что если при тестировании данной ЭВМ статистика $\bar{L}_f \leq 31$, то вероятность ошибки о наличии одного ПОАВ $\alpha = 0{,}14$. А вероятность того, что будет ошибочно принято решение о наличии двух вложенных образцов ПОАВ, составляет $\beta = 0$. Если окажется, что $32 \leq \bar{L}_f \leq 67$, вероятность ложного принятия решения об отсутствии ПОАВ $\alpha \leq 0{,}06$. Вероятность того, что будет ложно принято решение о наличии двух вложенных ПОАВ, составляет $\beta = 0$. В случае $\bar{L}_f \geq 86$ вероятность того, что будет ложно принято решение об отсутствии или наличии одного образца ПОАВ, равна $\alpha = \beta = 0$.

3.4 Предлагаемая методика обнаружения нелегитимного программного обеспечения, использующего технологию аппаратной виртуализации

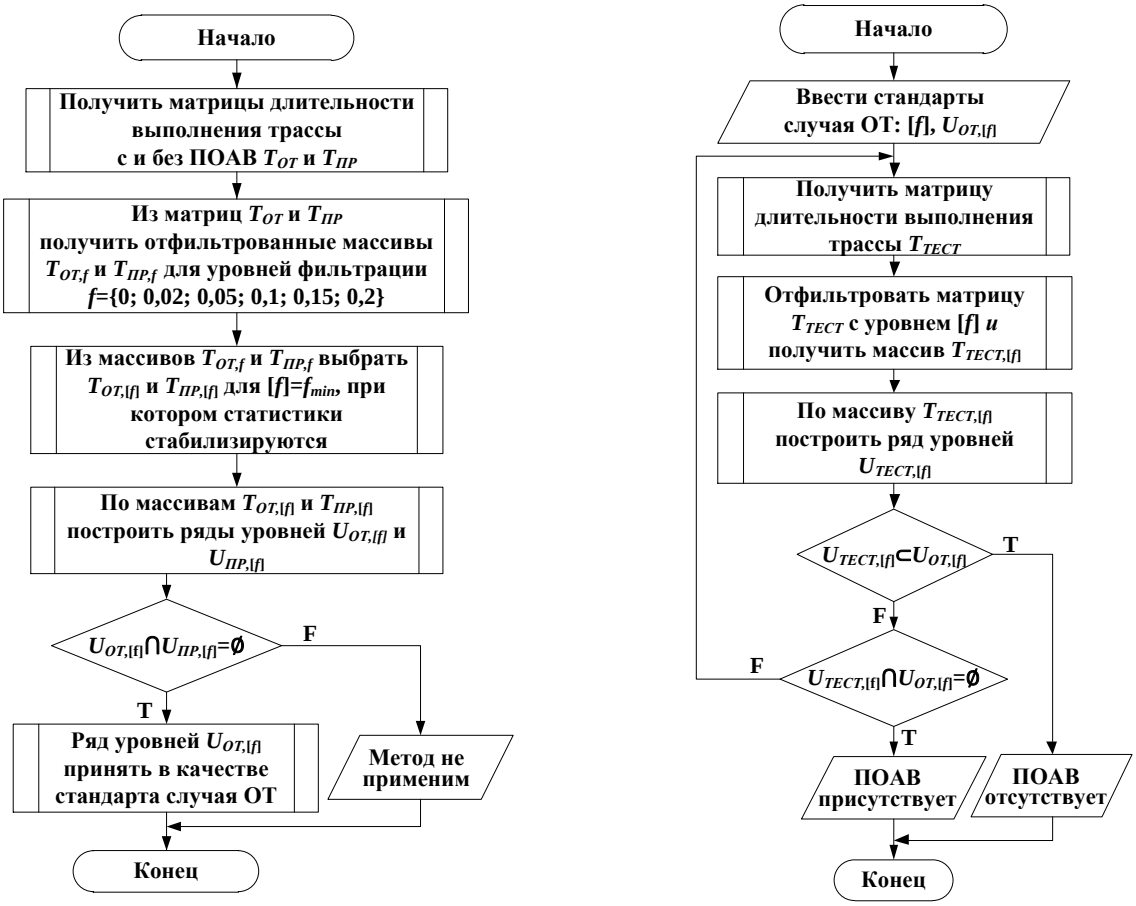
С учётом выполненных исследований обнаружение ПОАВ в ЭВМ должно выполняться в 2 этапа: предварительного и оперативного.

На предварительном этапе по полученным матрицам длительности выполнения трассы T_{OT} и $T_{ПР}$ выявляются нормативные показатели для случая отсутствия ПОАВ [S]: выборочные средние [$\bar{T}_{OT,f}$] для уровня фильтрации $f=0$, ряд уровней длительности выполнения трассы [U_{OT}] и пороговые значения статистик, указанные в табл. 6 для уровня фильтрации [f].

На оперативном этапе получают набор тестовых матриц $T_{тест}$, по ним вычисляют статистики $S_{тест}$ и по результатам сравнения их с нормативными

значениями $[S]$ принимают решение о наличии или отсутствии ПОАВ в тестируемой ЭВМ.

На рис. 30 и 31 приведены блок-схемы определения получения пороговых значений и обнаружения ПОАВ уровневым и комбинированным методами.



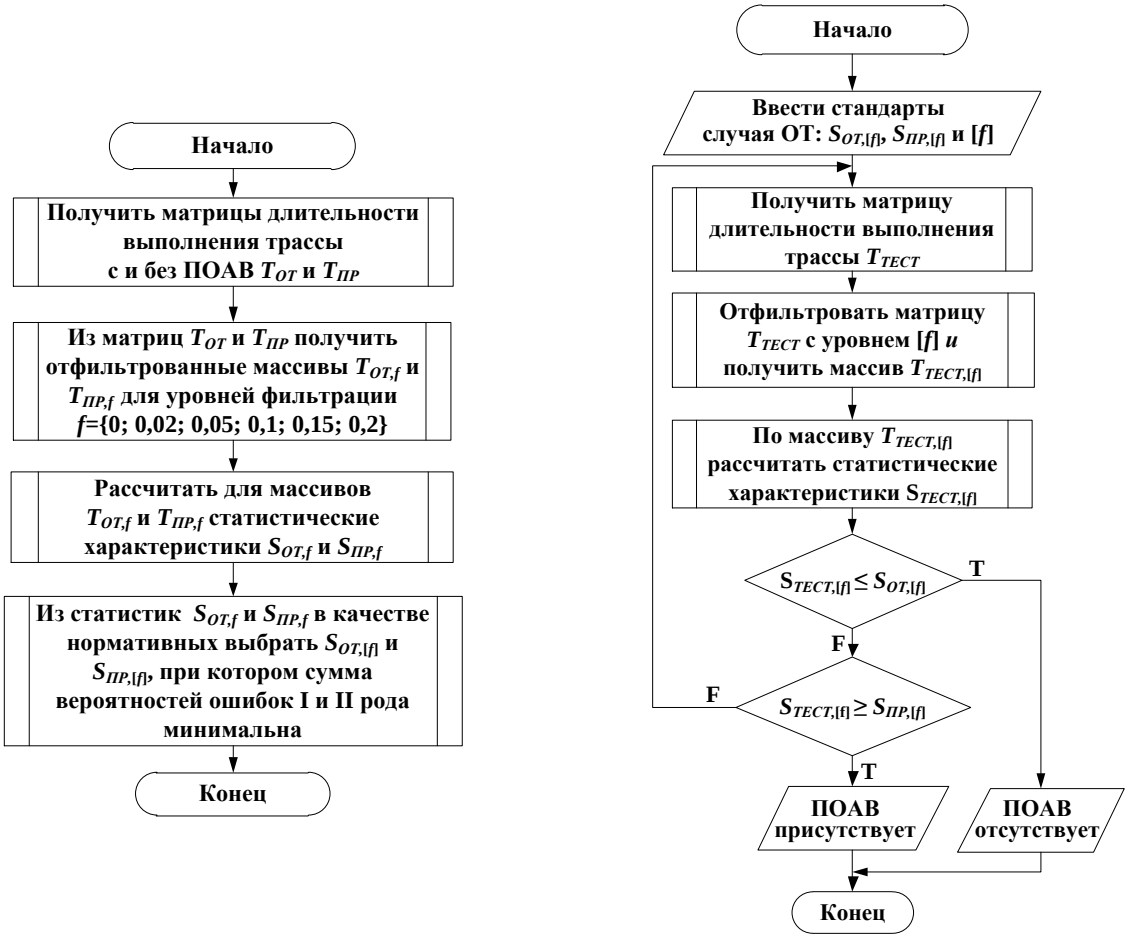
а *б*

Рисунок 30 — Алгоритмы: *а* – для выявления нормативных уровней длительности выполнения трассы и *б* – для обнаружения ПОАВ

В разделе 3.3 отмечалось, что результаты функционирования программного обеспечения для искажения показаний счётчика не всегда стабильны: в матрицах $T_{ПР}$ могут встречаться и аномальные (в том числе и отрицательные) значения, а средняя длительность выполнения трассы $\bar{t}_{ПР}$ – существенно отличаться от нормативной $[\bar{t}_{ПР}]$ для случая, когда ПОАВ отсутствует. Поскольку на практике такие ситуации встречаются довольно часто, их целесообразно использовать в идентификационных целях.

С учётом этого предлагается на оперативном этапе следующий порядок тестирования ЭВМ. Рекомендуется начинать с выявления в тестовых матрицах

$T_{тест}$ отрицательных значений. Их присутствие будет свидетельствовать, что в ЭВМ имеется ПОАВ, а компрометация счётчика тактов выполнена неточно.



а *б*

Рисунок 31 — Алгоритмы, используемые комбинированным методом: *а* – для выявления нормативных статистик длительности выполнения трассы и *б* – для обнаружения ПОАВ

Если выяснится, что отрицательных значений в матрицах $T_{тест}$ нет, следует сопоставить среднюю длительность выполнения трассы $\bar{t}_{тест}$ с нормативной $[\bar{t}_{ОТ}]$. Если окажется, что $\bar{t}_{тест} \gg [\bar{t}_{ОТ}]$, принимается решение о присутствии ПОАВ в тестируемой ЭВМ, а показания счётчика тактов либо вообще не искажены, либо искажаются недостаточно тщательно.

Если окажется, что отрицательных значений в матрицах $T_{тест}$ нет, а выборочные средние неразличимы ($\bar{t}_{тест} \approx [\bar{t}_{ОТ}]$), следует переходить к использованию уровневого и/или комбинированного методов обнаружения, алгоритмы которых представлены выше.

Методика выполнения процедур тестирования аналогична описанной в разделе 3.3.

Алгоритм выявления вложенных образцов ПОАВ подобен представленному на рис. 31. Особенности его использования изложены в разделах 3.2 и 3.3.

Пошаговая методика обнаружения нелегитимного ПОАВ приведена в табл. 8.

Таблица 8 – Пошаговая методика обнаружения нелегитимного ПОАВ

Название этапа	Содержание шагов каждого этапа
Предварительный	1. Аппаратным образом записать доверенную микропрограмму в BIOS. 2. Установить ОС. 3. Получить пороговые значения для обнаружения ПОАВ с помощью соответствующего алгоритма.
Оперативный	1. Начать проверку ЭВМ на отсутствие ПОАВ с помощью алгоритма обнаружения. 2. Последовательно установить дополнительное ПО (MS Office и т.п.). 3. Следить за сообщениями об обнаружении ПОАВ. 4. Для адаптации средства обнаружения к легитимному образцу ПОАВ выполнить шаг 3.

Кратко опишем каждый этап методики обнаружения ПОАВ.

Важно отметить, что выработка пороговых значений для обнаружения ПОАВ может выполняться только в системе с гарантированным отсутствием ПОАВ как в BIOS, так и в ОС.

На первом шаге предварительного этапа с помощью подключаемого аппаратного программатора осуществляется запись доверенной микропрограммы BIOS. Образ микропрограммы BIOS можно загрузить с официального сайта производителя материнской платы, после чего дополнительно осуществить контроль отсутствия ПОАВ в загруженном файле образа путём проведения его реверс-инжиниринга. Для обеспечения гарантированной записи микропрограммы BIOS необходимо использовать аппаратный программатор, поскольку вредоносное ПОАВ может нарушить работу программного средства обновления и даже внедриться в обновлённую

микропрограмму. Аппаратный способ записи микропрограммы BIOS позволяет этого избежать. В качестве аппаратного программатора можно выбрать, например, универсальный программатор «ТРИТОН» [48], а также различные экспериментальные схемы, имеющиеся в открытом доступе.

На втором шаге происходит установка ОС, в которой гарантированно отсутствует нелегитимное ПОАВ. С этой целью установку ОС следует производить с официального дистрибутива, поставляемого на лицензионном компакт-диске, либо загружать с сайта производителя. Гарантия отсутствия ПОАВ в установленной ОС и обновлённой микропрограмме BIOS в этом случае обеспечивается разработчиками, а также проведением мероприятий по реверс-инжинирингу ОС, рассмотрение которых выходит за рамки представляемой работы.

На третьем шаге выполняется приведенный выше алгоритм выработки пороговых значений. Для этого в ЭВМ устанавливается легитимный образец ПОАВ, содержащий минимальный набор инструкций и компрометирующий показания счётчика тактов. Это наиболее тяжёлый случай для обнаружения, который может встретиться на практике.

Целью оперативного этапа, выполняемого на стадии эксплуатации ЭВМ, является установление наличия или отсутствия ПОАВ в тестируемой ЭВМ.

Выполнение четвёртого шага осуществляется на стадии эксплуатации ЭВМ. Для обнаружения ПОАВ определяют показатели распределения, получаемые по тестовым выборкам длительности выполнения трассы, и сравнивают их с полученными на предыдущем этапе пороговыми значениями. Для этого используются алгоритмы обнаружения ПОАВ. Проверка системы на отсутствие ПОАВ осуществляется в непрерывном режиме на всех последующих шагах.

На пятом шаге устанавливается по отдельности необходимое пользователю программное обеспечение.

Если после установки очередного программного обеспечения средство обнаружения сообщило о наличии ПОАВ, необходимо определить его

легитимность. Методы решения этой задачи выходят за рамки представляемой работы. Можно отметить только, что узнать о наличии ПОАВ в приобретённом программном средстве можно путём обращения в соответствующую службу технической поддержки.

Если принято решение о наличии легитимного ПОАВ, необходимо адаптировать средство обнаружения к изменившемся условиям, для чего повторяется шаг 3. Таким образом, с использованием поэтапного адаптивного подхода создаются пороговые значения для каждого вложенного ПОАВ.

Полученные пороговые значения для очередного ПОАВ передаются в средство обнаружения.

3.5 Выводы

В этой главе были получены следующие результаты.

1. Экспериментальные исследования подтвердили изложенные в главе 2 выводы из анализа функционирования моделей процессора: длительность выполнения БП-инструкций, в частности CPUID, есть случайная величина, распределяющаяся по фиксированным уровням. При этом установлено, что
   – не менее 90% измерений значений длительности распределяются по стабильным уровням, зависящим от присутствия или отсутствия ПОАВ;
   – показатели вариации длительности выполнения трассы существенно выше в случае присутствия ПОАВ, чем в случае его отсутствия. Однако их статистическая различимость проявляется только при использовании не традиционных, а предложенных в данной главе статистических показателей.
2. Выявленная зависимость показателей распределения длительности выполнения трассы от отсутствия и присутствия ПОАВ позволила разработать методы обнаружения ПОАВ, нечувствительные к противодействию со стороны нарушителя.
3. Проблема сходимости и воспроизводимости экспериментальных данных преодолевается путём выполнения достаточного количества повторных

опытов в разные дни до наступления стабилизации статистических показателей длительности выполнения трассы и применения специальных статистик, разработанных с учётом наблюдаемых перекрытий их интервалов варьирования.

4. Предложенные методы позволяют выявлять как одиночные, так и вложенные образцы ПОАВ в различных условиях, в том числе и при целенаправленном противодействии со стороны нарушителя.
5. Разработанная методика обнаружения нелегитимного ПОАВ может быть использована специалистами по информационной безопасности в целях повышения защищённости ЭВМ в классе систем с поддержкой технологии аппаратной виртуализации.

4 Архитектура и реализация средства обнаружения программного обеспечения, использующего технологию аппаратной виртуализации

В данной главе предлагается архитектура и описывается реализация средства обнаружения ПОАВ. Описывается порядок настройки и эксплуатации средства обнаружения.

4.1 Состав и архитектура средства обнаружения

Основные требования к средству обнаружения ПОАВ следующие. Оно должно:

- обеспечивать получение пороговых значений статистик длительности выполнения трассы, используемых для обнаружения ПОАВ;
- обнаруживать как один, так и несколько вложенных образцов ПОАВ;
- обеспечивать в случаях необходимости также возможность ручной обработки регистрируемых данных.

Первые два требования выполняются путём реализации алгоритмов получения пороговых значений и обнаружения из раздела 3.4. Переносимость средства на различные ЭВМ обеспечивается выполнением первого требования. Последнее требование позволяет уточнять пороговые значения и выполняется путём программной реализации средства обнаружения.

Для удовлетворения этим требованиям средство обнаружения должно состоять из следующих подсистем (рис. 32):

- выработки пороговых значений;
- имитации действий нарушителя;
- выявления образцов ПОАВ.

Подсистема выработки пороговых значений обеспечивает сбор регистрируемых данных для случаев отсутствия и присутствия ПОАВ, их обработку и получение пороговых значений. При этом подсистема имитации действий нарушителя в процессе сбора данных на предварительном этапе

компрометирует счётчик тактов так, чтобы длительности выполнения трассы в случаях присутствия и отсутствия ПОАВ не отличались.

Подсистема выявления образцов ПОАВ с учётом наработок предыдущих подсистем обеспечивает обнаружение ПОАВ в непрерывном режиме.

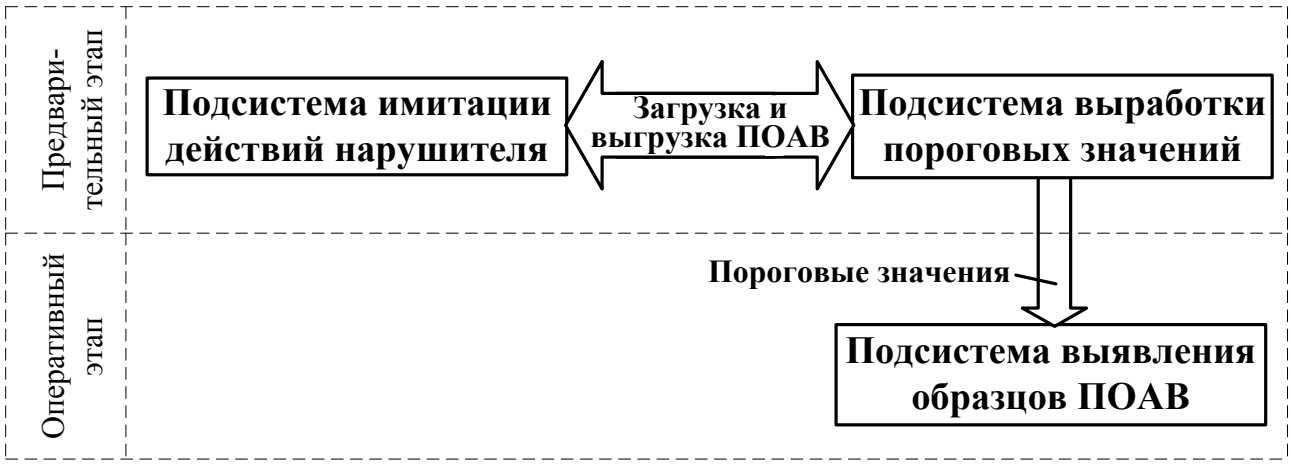


Рисунок 32 — Архитектура средства обнаружения ПОАВ

Согласно методике обнаружения нелегитимного ПОАВ (раздел 3.4), получение пороговых значений осуществляется на предварительном этапе путём совместной работы подсистем выработки пороговых значений и имитации действий нарушителя.

На оперативном этапе осуществляется работа подсистемы выявления образцов ПОАВ.

4.2 Форматы передаваемых данных

Для представления модулям заданий, регистрируемых данных и наборов пороговых величин были разработаны и реализованы специальные структуры данных.

Для этого информация о показаниях счётчика тактов сохраняется в виде контейнера из структур vector [45], содержащих четыре 32-битных целых числа для хранения младших и старших частей 64-разрядного значения счётчика, соответствующих его показаниям до и после измерения (рис. 33).

Данная структура заполняется драйвером при каждом отдельном измерении длительности выполнения трассы. Раздельное хранение младших и

старших разрядов позволяет упростить реализацию драйвера и повысить скорость его работы.

Старшие разряды значения счётчика до измерения	**Младшие разряды значения счётчика до измерения**
Старшие разряды значения счётчика после измерения	**Младшие разряды значения счётчика после измерения**

Рисунок 33 — Формат данных для хранения измерений

Для представления заданий были разработаны специальные структуры данных, приведённых на рис. 34.

Управляю-щий код	**Маска ядра**	**Длина трассы**	**Число измерений, строк в матрице**	**Число опытов, матриц**
Число повторов, столбцов в матрице		**Задержка между повторами измерений**		**Задержка между опытами**

Рисунок 34 — Фрагмент формата задания

Кратко опишем назначение каждого из полей структуры.

Управляющий код. Данное поле определяет код управления вводом/выводом IOCTL для указания способа измерения длительности выполнения трассы драйвером [44]. Это поле, не изменяя уже разработанные механизмы, позволяет расширять функциональные возможности средства, например, указывать, что необходимо регистрировать длительность выполнения трассы с предварительной очисткой кеш-памяти.

Маска ядра. Данное поле определяет маску процессорного ядра, проверяемого на отсутствие образца ПОАВ. На предварительном этапе опытный образец ПОАВ и подсистема выработки пороговых значений работают на первом ядре процессора с маской «0x1», на оперативном этапе проверяется отсутствие ПОАВ по очереди на каждом ядре процессора [121].

Длина трассы. Данное поле содержит заданное число БП-инструкций в трассе, по умолчанию значение равно 10.

Число измерений (число строк в матрице). Данное поле содержит число измерений длительности выполнения трассы, которые регистрируются непрерывно в цикле, по умолчанию оно равно 1000.

Число повторов (число столбцов в матрице). Данное поле определяет число повторных измерений длительности выполнения трассы. После каждого повтора осуществляется задержка. Число повторов определяет число столбцов в матрице, по умолчанию оно равно 10.

Задержка между повторами измерений. После каждого повтора осуществляется останов управляющей программы на заданное время. Значение данного поля задаёт задержку в миллисекундах между повторами измерений. По умолчанию значение равно 2000 мс.

Число опытов (число матриц). Данное поле определяет число повторных опытов. По умолчанию значение равно 10, а для обеспечения непрерывной работы средства обнаружения ПОАВ задаётся нулевым значением.

Задержка между опытами. Данное поле определяет задержку в миллисекундах после каждого опыта. По умолчанию оно равно 60000 мс.

Для хранения и обработки измерений использовались различные контейнеры стандартной библиотеки шаблонов «map» и «vector» из специально разработанных структур.

4.3 Подсистема выработки пороговых значений

Данная подсистема предназначена для «адаптации» разработанного средства обнаружения к особенностям обследуемой ЭВМ, которая состоит в получении данных о длительности выполнения трассы для случаев отсутствия и присутствия ПОАВ в обследуемой ЭВМ и в их обработке с целью получения пороговых значений статистических идентификаторов для обнаружения ПОАВ.

Подсистема состоит из четырёх основных модулей (рис. 35): управления, измерения длительности, обработки матрицы, расчёта статистических характеристик и выработки пороговых значений.

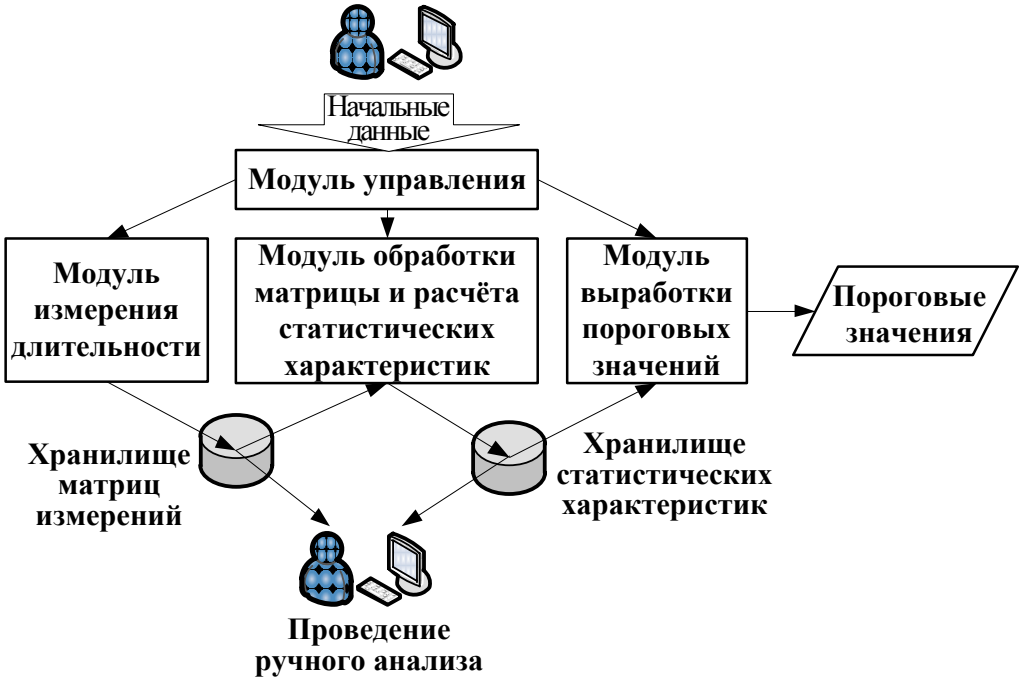


Рисунок 35 — Архитектура подсистемы выработки пороговых значений

После запуска этой подсистемы модуль управления создаёт для каждого из последующих модулей отдельные задания с учётом начальных данных, введенных оператором. Модуль управления реализован на языке программирования C++.

Составленное задание передаётся в модуль измерения, предназначенный для получения матриц измерений длительности выполнения трассы для случаев отсутствия и присутствия ПОАВ и последующей передачи их в хранилище. Модуль измерения длительности реализован на языке программирования C++. Он состоит из прикладной программы и драйвера (рис. 36).

Прикладная программа в соответствии с полученным заданием выделяет в памяти одномерный массив структур (столбец измерений), который передаётся драйверу вместе с заданием. Драйвер регистрирует длительность выполнения трассы, заполняет входной массив полученными значениями и возвращает управление прикладной программе. Прикладная программа, получив управление, останавливается на заданное число миллисекунд и в соответствии с заданием выделяет новый массив и передаёт его драйверу для заполнения. Операция повторяется до тех пор, пока матрица не заполнится целиком.

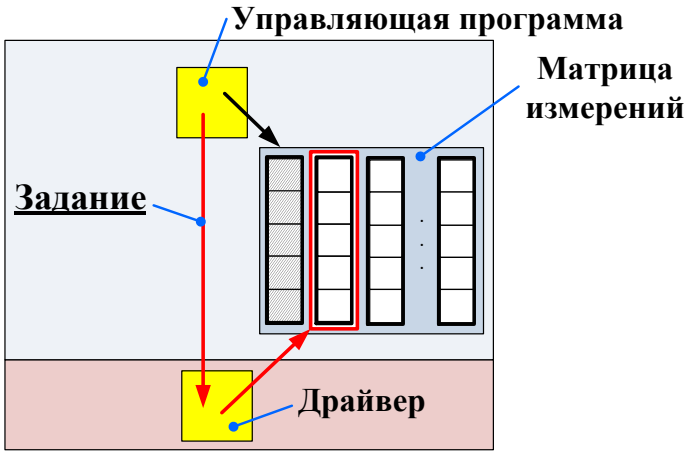


Рисунок 36 — Схема работы модуля измерения длительности выполнения трассы

Фрагмент исходного кода драйвера для измерения длительности приведён на рис. 37.

```
 1: KeSetSystemAffinityThread(affinity)
 2: KfRaiseIrql(HIGH_LEVEL)
 3: for (...)
 4: {
 5: __asm
 6:     {
 7:         RDTSC
 8:         MOV hst, EDX
 9:         MOV lst, EAX
10:         CPUID // 1
11:         ...
12:         CPUID // 10
13:         RDTSC
14:         MOV hfin, EDX
15:         MOV lfin, EAX
16:     }
17:     save_time(hst, lst, hfin, lfin)
18: }
```

Рисунок 37 — Фрагмент исходного кода драйвера для измерения длительности выполнения трассы на языке программирования C++

В строке 1 функция «KeSetSystemAffinityThread» осуществляет привязку программного кода к заданной маске ядра процессора. В строке 2 осуществляется повышение уровня запросов прерывания (IRQL) до наивысшего значения 31 (HIGH_LEVEL). В строке 3 организуется цикл, в строках 7-9 с помощью инструкции RDTSC осуществляется чтение счётчика тактов и запись его показаний в переменные «hst» и «lst». В строках 10-12 показана трасса из 10-ти инструкций CPUID. В строках 13-15 осуществляется

повторное чтение счётчика тактов и запись его показаний в переменные «hfin» и «lfin». В 17-й помещена функция сохранения регистрируемых показаний счётчика тактов.

Описание процедуры искажения длительности выполнения трассы приведено в разделе 4.4.

Составленное модулем управления задание передаётся в модуль обработки матрицы и расчёта статистических характеристик (модуль OMPCX), который извлекает очередную матрицу из хранилища и после её обработки по изложенной в разделе 3.3 методике рассчитывает статистические характеристики длительности выполнения трассы, которые помещаются в соответствующее хранилище.

Модуль OMPCX последовательно выполняет преобразование полученных данных, предварительную обработку и расчёт статистических характеристик данных измерений.

Преобразование полученных данных состоит в переводе показателей счётчика тактов до и после выполнения трассы в длительность выполнения трассы ΔT_{TP}, по формуле

$$\Delta T_{TP} = ((Finish_{HI} \ll 32)|Finish_{LO}) - ((Start_{HI} \ll 32)|Start_{LO}), \qquad (21)$$

где $Start_{HI}$ и $Start_{LO}$, $Finish_{HI}$ и $Finish_{LO}$ – старшие и младшие разряды значений счётчика до и после измерений соответственно; символ $\ll$ – побитовый сдвиг влево на заданное количество разрядов.

Полученные значения длительности выполнения трассы сохраняются в файл формата «csv» [158] на случай, если потребуется их ручная обработка аналитиком (например, с использованием пакета программ Matlab совместно с Excel из пакета программ Microsoft Office [120]).

Модуль OMPCX реализован на языке программирования Matlab, в качестве примера, фрагменты его исходного кода приведены в приложении Б. Взаимодействие Matlab-программы и программы на языке C++ описано в книге И. Бейя [1].

Получив от модуля управления задание, модуль выработки пороговых значений выполняет чтение из хранилища статистических характеристик, отбирает из них пригодные в качестве идентификационных показателей для обнаружения ПОАВ, формирует их пороговые значения и передаёт на вход подсистемы выявления МВМ.

В тех случаях, когда модуль выработки пороговых значений по каким-то причинам в автоматическом режиме не справляется с заданием, потребуется анализ ситуации аналитиком. Для этого выполняется обращение к хранилищам матриц длительности выполнения трассы и их статистик, выполняется анализ реализаций процесса измерения длительности выполнения трассы, вариационных рядов и полигонов распределения длительности выполнения трассы и по полученным результатам уточняются показатели, используемые при обработке полученных данных.

4.4 Подсистема имитации действий нарушителя

Данная подсистема необходима для имитации ситуации, наихудшей из возможных на практике для обнаружения ПОАВ. Она должна обеспечивать загрузку и выгрузку ПОАВ и для противодействия его обнаружению искажать показания счётчика так, чтобы средние длительности выполнения трассы в случаях отсутствия и присутствия ПОАВ практически не отличались бы.

С учётом этих требований архитектура данной подсистемы должна состоять из модулей управления и образца ПОАВ (рис. 38). Модуль управления получает управление после запуска подсистемы. С его помощью оператор ЭВМ выдаёт модулю команды на загрузку и выгрузку ПОАВ, а также задаёт величину искажения показания счётчика тактов.

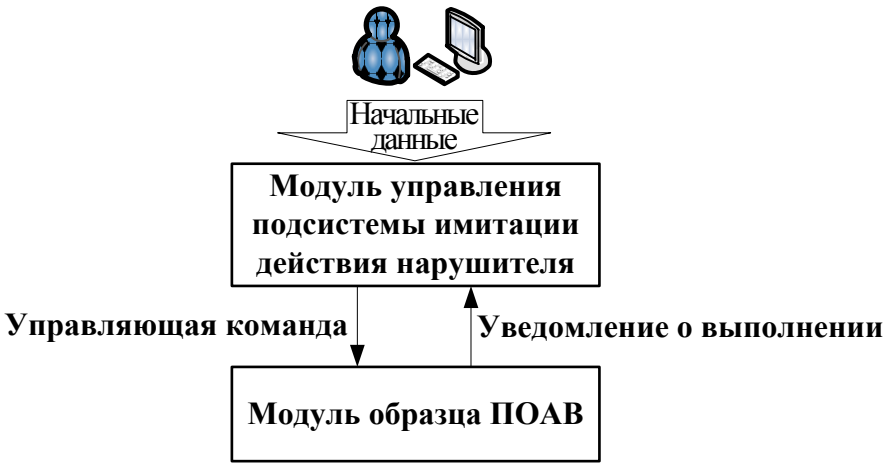


Рисунок 38 — Архитектура подсистемы имитации действий нарушителя

Модуль образца ПОАВ поддерживает выполнение команд модуля управления. В качестве его прототипа был использован разработанный Ш. Эмблетоном (S. Embleton) [88] программный образец, в который были добавлены функции компрометации счётчиков тактов при перехвате БП-инструкций (рис. 39 и табл. 9).

Таблица 9 – Функции, используемые для искажения показаний процессорного счётчика тактов TSC

Название функции	Назначение функции
set_delta	Позволяет задавать величину компрометации «g_Delta»
read_tsc	Записывает значение счётчика тактов в глобальную переменную «g_TscNext»
patch_tsc	Вычитает из значения переменной «g_TscNext» величину «g_Delta» и записывает уменьшенное значение «g_TscNext» в счётчик тактов TSC

После получения управления обработчик ПОАВ вызывает функцию «read_tsc», а перед возвратом управления – функцию «patch_tsc». Расчёт среднего значения длительности выполнения трассы осуществляется с использованием модуля измерения длительности и модуля OMPCX подсистемы выработки пороговых значений.

```
__int64 g_TscNext = 0;
__int64 g_Delta = 1000;

void set_delta(__int64 delta)
{
    g_Delta = delta;
}

void read_tsc()
{
    ULONG cycles_high = 0, cycles_low = 0;
    __asm
    {
        pushad
        RDTSC
        mov cycles_high, edx
        mov cycles_low, eax
        popad
    }
    g_TscNext = (((unsigned __int64)cycles_high << 32) | cycles_low);
}

void patch_tsc()
{
    g_TscNext -= g_Delta;
    ULONG cycles_high = (ULONG)(g_TscNext>>32);
    ULONG cycles_low = (ULONG)((g_TscNext<<32)>>32);
    __asm
    {
            MOV ECX, 10h        /*IA32_TIME_STAMP_COUNTER*/
            mov edx,cycles_high
            mov eax,cycles_low
            WRMSR
    }
}
```

Рисунок 39 — Фрагмент модуля ПОАВ для искажения длительности выполнения инструкции

Величина компрометации показаний счётчика тактов δ определяется с использованием модуля измерения длительности и модуля ОМРСХ подсистемы выработки пороговых значений по формуле

$$\delta = \frac{\left(\overline{t}_{ПР} - \overline{t}_{ОТ}\right)}{10}, \tag{22}$$

где $\overline{t}_{ОТ}$ и $\overline{t}_{ПР}$ – средняя длительность выполнения трассы из 10-и инструкций CPUID в случаях отсутствия и присутствия ПОАВ.

Опыт показывает, что, используя выражение (22) и интерактивную коррекцию δ, можно добиться совпадения средних значений длительности выполнения трассы $\overline{t}_{ОТ}$ и $\overline{t}_{ПР}$ с точностью до 10 тактов. Однако, как отмечалось

в разделах 2.2 и 3.2, даже при самом тщательном выборе величины δ в последующих повторных опытах из-за случайного характера смены режимов работы процессора выборочные средние $\bar{t}_{ПР}$ могут значительно отличаться от выборочных средних $\bar{t}_{ОТ}$.

4.5 Подсистема выявления образцов ПОАВ

Эта подсистема предназначена для обнаружения образцов ПОАВ и подсчёта их числа в ЭВМ. Она состоит из модулей управления, измерения длительности выполнения трассы, обработки матрицы, расчёта статистических характеристик и принятия решения (рис. 40).

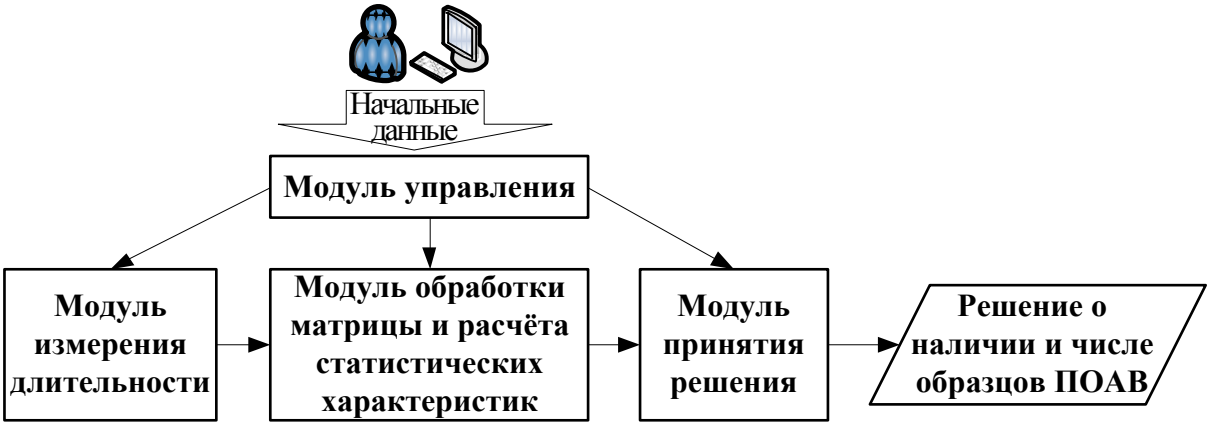


Рисунок 40 — Архитектура подсистемы выявления образцов ПОАВ

Входными величинами для модуля управления этой подсистемы являются пороговые значения, полученные в результате работы подсистем выработки пороговых значений и имитации действий нарушителя. На основе входных параметров формируются задания для трёх следующих модулей. Модуль управления реализован на языке программирования C++.

Модуль измерения длительности и модуль OMPCX описаны выше.

Модуль измерения длительности, получив задание, заполняет матрицу длительности выполнения трассы и передаёт её в модуль OMPCX, который вычисляет соответствующие статистики и передаёт их в модуль принятия решения.

Модуль принятия решения осуществляет сравнение полученных характеристик с пороговыми значениями, переданными в составе задания от

модуля управления, и принимает решение о наличии и количестве образцов ПОАВ.

Работа подсистемы выявления образцов ПОАВ аналогична подсистеме выработки пороговых значений за исключением того, что модуль ОМРСХ рассчитывает лишь указанные в задании статистические показатели и только для определённых уровней фильтрации.

Модуль принятия решения реализован на языке программирования С++.

4.6 Порядок использования средства обнаружения ПОАВ

Средство обнаружения ПОАВ предназначено для эксплуатации в 32-х разрядных ОС семейства Windows: Windows 2000, Windows XP, Windows Server 2003 и 2008, Windows Vista, Windows 7.

Средство обнаружения должно быть запущено от имени администратора.

Модуль ПОАВ подсистемы имитации действия нарушителя работает в режиме страничной адресации без поддержки режима расширенных физических адресов (Physical Address Extension, PAE). Для отключения этого режима необходимо выполнить действия, указанные в табл. 10.

Таблица 10 – Действия для отключения режима PAE в зависимости от ОС

ОС	Действия для отключения режима PAE
Windows XP	В файл boot.ini добавить параметры для загрузки: «/noexecute=AlwaysOff /nopae»
Windows Vista, Windows 7	Выполнить в интерпретаторе CMD, запущенного от имени администратора, следующие команды: bcdedit /set {current} nx AlwaysOff bcdedit /set {current} pae ForceDisable

На предварительном этапе пошаговой методики обнаружения нелегитимного ПОАВ (раздел 3.4) перед получением пороговых значений необходимо по методике из раздела 4.4 выполнить следующее.

1. Подготовить к работе подсистему имитации действий нарушителя.
2. Получить набор матриц длительности выполнения трассы для случаев присутствия и отсутствия ПОАВ.

3. По каждой полученной матрице вычислить соответствующие статистики.
4. Сравнением статистик длительности выполнения трассы для случаев присутствия и отсутствия ПОАВ получить пороговые значения идентификационных показателей для обнаружения ПОАВ.

На оперативном этапе пошаговой методики выполняются процедуры 1-3, а полученные тестовые статистики на шаге 4 сравниваются с соответствующими пороговыми значениями, полученными на предварительном этапе. По результатам сравнения формируется заключение о наличии или отсутствии ПОАВ в тестируемой ЭВМ.

4.7 Выводы

В настоящей главе получены следующие результаты:

– разработаны требования к средству обнаружения ПОАВ и построена его архитектура;
– разработаны форматы передаваемых данных между компонентами системы;
– изложен порядок использования средства обнаружения ПОАВ при обследовании ЭВМ.

Приведенные в этой главы сведения могут быть использованы в качестве практического руководства по использованию представленных в данной работе методов обнаружения ПОАВ в ЭВМ.

5 Внедрение методики обнаружения нелегитимного программного обеспечения, использующего технологию аппаратной виртуализации

В данной главе рассмотрены примеры практического применения полученных автором результатов при решении конкретных прикладных задач в трёх проектах.

В первом проекте разработанное средство обнаружения ПОАВ было использовано в составе системы обеспечения информационной безопасности для проверки отсутствия ПОАВ в части парка ЭВМ Государственного научного центра Российской Федерации федерального государственного унитарного предприятия «Центральный научно-исследовательский институт химии и механики им. Д.И. Менделеева» (ГНЦ РФ ФГУП «ЦНИИХМ»).

Во втором проекте разработанное программное средство обнаружения было использовано как самостоятельная программа для контроля отсутствия негласного использования ПОАВ при подготовке к вводу в эксплуатацию части парка ЭВМ федерального государственного унитарного предприятия «Научно-исследовательского института стандартизации и унификации» (ФГУП «НИИСУ»).

Осуществлено также внедрение результатов работы в образовательный процесс кафедры «Криптология и дискретная математика» Национального Исследовательского Ядерного Университета «Московского Инженерно-Физического Института» (НИЯУ МИФИ). Основные результаты работы, связанные с исследованием способов сокрытия и обнаружения программного обеспечения, были добавлены в учебный курс «Безопасность операционных систем». В рамках внедрения осуществлялась подготовка материала лабораторных работ для ознакомления студентов с новейшими технологиями сокрытия ПО и для обучения подходам к обнаружению скрытого ПО.

5.1 Использование предложенного средства обнаружения в системе обеспечения информационной безопасности ГНЦ РФ ФГУП «ЦНИИХМ им. Д.И. Менделеева»

Основные направления деятельности предприятия ГНЦ РФ ФГУП «ЦНИИХМ им. Д.И. Менделеева» (далее ЦНИИХМ) связаны с проведением «фундаментальных, поисковых и прикладных научно-исследовательских, опытно-конструкторских и технологических работ, в том числе создание энергосберегающих технологий, прецизионных и нанометрических технологий контроля и других» [52]. В 2005 г. ЦНИИХМ был передан в ведение Федеральной службе по техническому и экспортному контролю (ФСТЭК России). В настоящее время ЦНИИХМ – это одно из наиболее динамично развивающихся и перспективных предприятий оборонно-промышленного комплекса России, поэтому особенно актуальна задача своевременного обнаружения и противодействия новейшим инженерно-техническим средствам разведки.

Приведённый на рис. 41 фрагмент корпоративной сети ЦНИИХМ, состоит из двух контуров. Первый контур подключён к сети Интернет с помощью сетевых коммуникаций, второй не имеет такого доступа (онлайн и оффлайн соответственно).

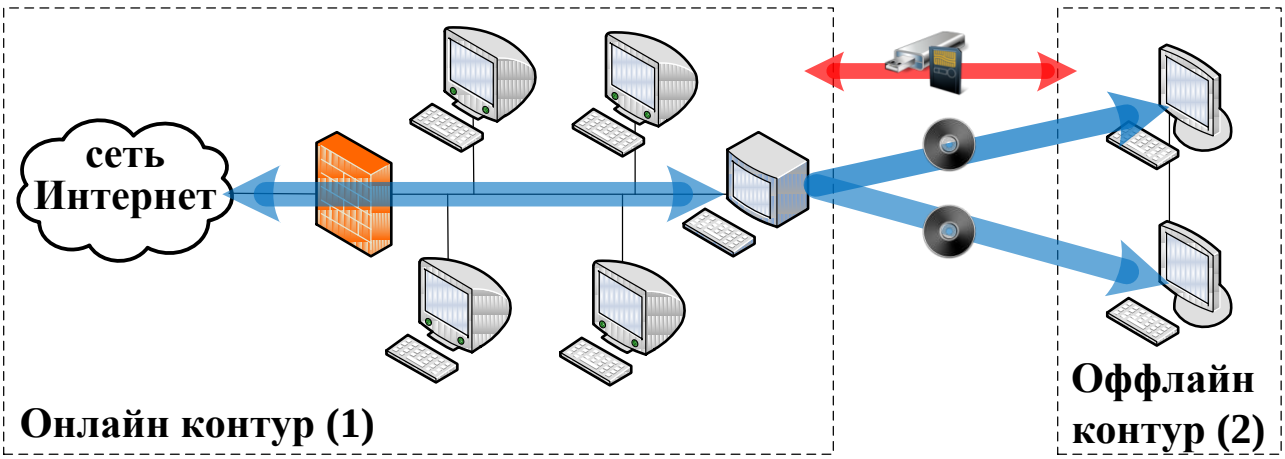


Рисунок 41 — Схема фрагмента корпоративной сети ЦНИИХМ

Информационное взаимодействие между двумя контурами осуществляется с помощью внешних носителей. Специфика данных,

обрабатываемых на ЭВМ второго контура, не позволяет подключать их к сети Интернет и осуществлять установку обновлений ПО с помощью штатных средств. Поэтому ЭВМ второго контура не защищены от угроз, использующих даже известные уязвимости.

Разработанная сотрудниками ЦНИИХМ система обеспечения информационной безопасности (СОИБ) позволяет повышать уровень защищённости ЭВМ, находящихся во втором контуре за счёт установки актуальных обновлений для прикладного и системного ПО.

СОИБ представляет собой самостоятельное программное средство, имеющее собственную базу данных обновлений (БДО) для таких ПО, как ОС Windows, Microsoft Office, Adobe Acrobat Reader и др.

БДО представляет собой отдельный файл, хранящий информацию о пакетах обновлений. Пакеты обновлений хранятся в соответствующих папках для каждого ПО. В текущей версии СОИБ для реализации БДО используется библиотека SQLite [102].

При запуске СОИБ на ЭВМ первого контура автоматически происходит сбор информации об актуальных обновлениях заданного ПО, в результате оператор получает список обновлений. Оператор может указать в полученном списке как отдельные пакеты обновлений для определённого ПО, так и выбрать все пункты списка. В результате отмеченные обновления будут загружены из сети Интернет и зарегистрированы в БДО.

Для передачи СОИБ с обновлённой БДО на ЭВМ второго контура используются оптические компакт-диски, внешние жёсткие диски и флеш-карты с механической блокировкой от записи.

При запуске СОИБ на входящую во второй контур ЭВМ происходит сбор сведений об установленном ПО и его пакетах обновлений и поиск в БДО сведений об актуальных обновлениях для каждого ПО. В результате оператору предоставляется список потенциально уязвимого ПО и перечень актуальных пакетов обновлений. Оператор может отметить как отдельные пакеты

обновлений, так и выделить весь список целиком, в результате на ЭВМ будут установлены отмеченные обновления.

Внедрение разработанного средства обнаружения осуществлялось путём включения его подсистемы выявления образцов ПОАВ в состав СОИБ (рис. 42).

Рисунок 42 — Архитектура модернизированной системы обеспечения информационной безопасности ЦНИИХМ

Для внедрения были выбраны одиннадцать новых ЭВМ с процессорами Intel Xeon X5600 без ОС для работы во втором контуре.

Для реализации предварительного этапа методики обнаружения ПОАВ на каждую ЭВМ была установлена ОС Windows 7 Professional, с помощью СОИБ были получены пороговые значения путём совместной работы подсистем имитации действия нарушителя и выработки пороговых значений, после чего установлены актуальные обновления.

В ЦНИИХМе для каждого сотрудника составляется список необходимого программного обеспечения и оформляется в виде заявки, в соответствии с которой периодически закупается указанное ПО.

Для реализации оперативного этапа обнаружения ПОАВ в соответствии с заявками пользователей производилась установка программного обеспечения на ЭВМ с ОС, после чего с помощью СОИБ инсталлировались актуальные обновления ПО и производилось обнаружение ПОАВ.

Таким образом, в результате проведённых мероприятий одиннадцать ЭВМ были готовы к эксплуатации в защищённом от нелегального ПОАВ режиме.

5.2 Применение предложенного средства обнаружения для контроля отсутствия негласного программного обеспечения, использующего технологию аппаратной виртуализации в классе ЭВМ парка ФГУП «НИИСУ»

Федеральное государственное унитарное предприятие «Научно-исследовательский институт стандартизации и унификации» (далее НИИСУ) было создано в 1956 году с целью «организации работ по стандартизации и унификации, каталогизации, проведению научно-исследовательских и опытно-конструкторских работ по разработке, изготовлению и испытаниям продукции оборонного и гражданского назначения, выпускаемой предприятиями промышленности» [32]. В настоящее время НИИСУ является ведущим научно-исследовательским предприятием авиационной промышленности России и находится в ведении Министерства промышленности и торговли Российской Федерации (Минпромторг России).

Для выполнения НИОКР «Фосса» в НИИСУ были приобретены пятнадцать ЭВМ с процессорами Intel Core i7 995.

Реализованное средство обнаружения ПОАВ было использовано как самостоятельная программа для проверки заданных ЭВМ. Процесс подготовки ЭВМ к эксплуатации был разделён на три этапа.

На первом этапе осуществлялась запись новой микропрограммы BIOS с использованием аппаратного программатора «ТРИТОН», что гарантировало отсутствие ПОАВ в BIOS.

На втором этапе осуществлялась установка 32-разрядной ОС Windows 7 Ultimate.

Затем производилась выработка пороговых значений для случаев присутствия и отсутствия ПОАВ, после чего ЭВМ подключались к сети Интернет и осуществлялось обновление ОС.

На третьем этапе осуществлялась установка следующего программного обеспечения: Microsoft Office 2010 (Word, Excel, Power Point), Microsoft Visual Studio 2010 Ultimate, Visual Assist, Google Chrome, IDA, Matlab, Windows Driver Kits, Far Manager. Осуществляемая при этом проверка указывала на отсутствие ПОАВ в этом программном обеспечении.

В результате был успешно подготовлен и передан в эксплуатацию ряд ЭВМ. Реализованное средство обнаружения ПОАВ позволило обеспечить контроль отсутствия ПОАВ как на этапе ввода в эксплуатацию ЭВМ, так и на этапе их эксплуатации.

5.3 Разработка лабораторных работ для курса «Безопасность операционных систем» кафедры «Криптология и дискретная математика» НИЯУ МИФИ

На основе теоретических и практических материалов данной диссертационной работы был расширен существующий учебный курс «Безопасность операционных систем», читающийся для студентов шестого семестра на кафедре «Криптология и дискретная математика» НИЯУ МИФИ.

Проводимые лабораторные работы по данному курсу посвящены изучению архитектуры ОС Windows и включают в себя исследование и разработку драйверов для ОС Windows, среди которых можно выделить следующие: драйвер-фильтр для клавиатуры PS/2, драйвер-перехватчик системных сервисов на примере ПО Skype и драйвер для получения копии адресного пространства выбранного процесса.

Курс «Безопасность операционных систем» был расширен за счёт предоставления студентам информации о механизмах сокрытия и обнаружения программного обеспечения, в том числе стеганографическими и техническими (руткит-механизмами), работающими как внутри, так и вне ОС.

Особое внимание уделено новым процессорам фирмы Intel и AMD с поддержкой технологии аппаратной виртуализации. Был представлен материал о возможностях применения новой технологии как для решения задач защиты информации, так и для реализации скрытого программного обеспечения различного назначения. Студенты знакомились с подходами, которые могут быть использованы нарушителем для сокрытия ПОАВ, и средствами их обнаружения, в числе которых и разработанное в данной работе.

Был представлен анализ режима системного управления для реализации скрытого программного обеспечения.

Особое внимание было уделено новейшей проблеме – обнаружению вложенных образцов ПОАВ. Была представлена методика обнаружения нелегитимного ПОАВ.

Расширенный курс в отличие от иностранных курсов Г. Хоглунда (G. Hoglund) «Offensive Aspects of Rootkit Technology» [103] и Дж. Рутковской (J. Rutkowska), Р. Войтчука (R. Wojtczuk) и А. Терешкина (A. Tereshkin) «Understanding Stealth Malware Training» [146], содержит анализ статистических характеристик длительности выполнения трассы для случаев отсутствия и присутствия ПОАВ и методику обнаружения нелегитимного ПОАВ.

К проведению лабораторных работ предлагается привлекать специалистов, непосредственно занимающихся проблемой обнаружения программных закладок и каналов утечек информации. Выполнение поставленных заданий тесно связано с программированием на C/C++, Assembler x86 и работой с технической документацией на английском языке, а также с закреплением навыков работы с широко используемыми средствами разработки и отладки приложений: Microsoft Visual Studio, Windows Driver Kit (WDK), Debugging Tools for Windows (WinDbg).

В результате прохождения курса студенты получают не только важные теоретические знания и практические навыки использования новейшей технологии аппаратной виртуализации, но и сведения о применении этой

технологии как для нарушения информационной безопасности, так и для повышения защищённости ЭВМ от угроз.

5.4 Выводы

Программное средство обнаружения нелегитимного ПОАВ использовано в составе системы обеспечения информационной безопасности в Государственном научном центре Российской Федерации федеральном государственном унитарном предприятии «Центральный научно-исследовательский институт химии и механики им. Д.И. Менделеева».

Программное средство обнаружения было использовано как самостоятельная программа для контроля отсутствия негласного использования ПОАВ при подготовке к вводу в эксплуатацию части парка ЭВМ федерального государственного унитарного предприятия «Научно-исследовательского института стандартизации и унификации».

Теоретические и прикладные результаты, полученные в ходе выполнения диссертационной работы, использованы в учебном курсе «Безопасность операционных систем» кафедры «Криптология и дискретная математика» НИЯУ МИФИ.

Указанные внедрения подтверждены соответствующими актами.

Заключение

В ходе выполнения работы были получены следующие научные и практические результаты:

1. Проведен анализ существующих способов обнаружения программного обеспечения, использующего технологию аппаратной виртуализации. Обоснована необходимость разработки нового подхода к обнаружению ПОАВ, в том числе вложенных образцов.
2. Построена модель нарушителя, позволившая проанализировать возможные угрозы обрабатываемой в ЭВМ информации.
3. Представлены в терминах теории графов модели выполнения трассы для процессоров с поддержкой аппаратной виртуализации в случаях отсутствия (присутствия) одного образца (нескольких образцов) ПОАВ. Предложенные модели позволили выявить закономерности длительностей выполнения трассы, которые легли в основу разработанных подходов к обнаружению ПОАВ в ЭВМ.
4. Предложен критерий присутствия образца ПОАВ. Получен теоретически и подтверждён экспериментально вывод о том, что статистические характеристики длительности выполнения трассы зависят от наличия ПОАВ. В случае присутствия ПОАВ значение и вариабельность длительности выполнения трассы существенно больше, чем в случае его отсутствия.
5. Разработана методика обнаружения нелегитимного ПОАВ, состоящая из двух этапов – предварительного и оперативного. На предварительном этапе выявляются пороговые значения статистик. На оперативном этапе на основе полученных статистик осуществляется обнаружение ПОАВ в непрерывном режиме.
6. Построена архитектура и разработано программное средство обнаружения программного обеспечения, использующего технологию аппаратной виртуализации.

7. Разработанное программное средство обнаружения ПОАВ применено в составе системы обеспечения информационной безопасности для контроля отсутствия ПОАВ в ряде ЭВМ парка ГНЦ РФ ФГУП «ЦНИИХМ», что позволило контролировать отсутствие нелегальных образцов ПОАВ.
8. Реализованное программное средство обнаружения ПОАВ использовано при подготовке к вводу в эксплуатацию ряда ЭВМ парка ФГУП «НИИСУ», что позволило проконтролировать отсутствие нелегальных образцов ПОАВ.
9. Материалы диссертационной работы были использованы в лабораторных работах для учебного курса «Безопасность операционных систем» кафедры «Криптология и дискретная математика» НИЯУ МИФИ.
10. Результаты работы могут быть использованы при разработке руководящего документа ФСТЭК РФ, регламентирующего проверку средств вычислительной техники и автоматизированных систем с поддержкой аппаратной виртуализации.
11. С использованием разработанной методики в программном обеспечении Disk Director 2010 компании Acronis был обнаружен образец ПОАВ, сведений о котором нет ни в технической документации, ни на официальном сайте этой компании.

Изложенные в данной работе материалы могут быть использованы при проведении мероприятий по противодействию техническим разведкам и по защите информации от утечки по техническим каналам.

Список использованных источников

1 Бей И. Взаимодействие разноязыковых программ в Microsoft Windows. Руководство программиста. – М.: Вильямс, 2005. – 880 с.

2 Бейкер С., Уотерман Ш., Иванов Д. Под перекрестным огнем: критически важная инфраструктура в эпоху кибервойны. – http://www.mcafee.com/ru/resources/reports/rp

3 Бекман И.Н. Метод моментов (Параметрические моменты от кинетических кривых в обработке и интерпретации результатов диффузионных экспериментов). – profbeckman.narod.ru/MOM.htm

4 Болотов П. Принципы работы кэш-памяти – http://alasir.com/articles/cache_principles/tlb_virtual_memory_rus.html

5 Бурдаев О.В., Иванов М.А., Тетерин И.И. Ассемблер в задачах защиты информации / Под ред. И. Ю. Жукова – М.: Кудиц-Образ, 2002. – 320 с.

6 Буховец А.Г., Москалев П.В., Богатова В.П., Бирючинская Т.Я. Статистический анализ данных в системе R – http://gis-lab.info/docs/books/moskalev2010_statistical_analysis_with_r.pdf

7 Васнев С.А. Статистика: Учебное пособие. М.: МГУП, 2001. – 170 с.

8 Вентцель Е.С. Теория вероятностей. – 5-е изд., испр. – М.:Издательский центр «Академия», 2003. – 576 с.

9 ГОСТ Р ИСО 5725 2002 Точность (правильность и прецизионность) методов и результатов измерений, части 1-6, М.: ИПК Издательство стандартов, 2002

10 Гук М., Юров В. Процессоры Pentium III, Athlon и другие. – СПб.: Питер, 2000. – 480 с.

11 Егоров В.Б., Матвеев Е.А. Регионы времени как объекты операционной системы общего назначения // Информатика и ее применения. – М.: Торус Пресс, 2008, №4, Т. 2. – с. 74-84.

12 Елисеева И.И., Юзбашев М.М. Общая теория статистики: Учебник / Под ред. И. И. Елисеевой. – 5-е изд., перераб. и доп. – М.: Финансы и статистика, 2004. – 656 с.

13 Касперски К. Blue pill/red pill - the matrix has windows longhorn. – http://www.insidepro.com/kk/165/165r.shtml

14 Касперски К. Разгон и торможение Windows NT – http://www.insidepro.com/kk/030/030r.shtml

15 Клименко А.В., Алексеев К.Д. Обзор аппаратных средств и API сервисов определения времени в персональном компьютере // Математичні машини і системи, 2009, № 3. – с. 137–143.

16 Кобзарь А.И. Прикладная математическая статистика. Для инженеров и научных работников. - М.: ФИЗМАТЛИТ, 2006. – 816 с.

17 Коркин И.Ю. Выявление вложенных мониторов виртуальных машин // Системы высокой доступности. 2011, №2, т.6 – с. 76-77.

18 Коркин И.Ю. Технологии сокрытия вредоносных программ и новые способы противодействия им // Безопасность информационных технологий, 2009, № 4, с. 43-46.

19 Коркин И.Ю. Способ обнаружения скрытых процессов в ОС Windows. В сб. научных трудов 16-й Всероссийской конференции «Проблемы информационной безопасности в системе высшей школы». – М.: 2009. – с. 111-112.

20 Коркин И.Ю., Петрова Т.В., Тихонов А.Ю. Метод обнаружения аппаратной виртуализации в компьютерных системах. В сб. научных трудов 17-й Всероссийской конференции «Проблемы информационной безопасности в системе высшей школы». М.: 2010. – с. 114-115.

21 Коркин И.Ю. Критерий обнаружения монитора виртуальных машин в компьютерных системах. В сб. материалов XIX Общероссийской научно-технической конференции «Методы и Технические Средства Обеспечения Безопасности Информации». СПб.: 2010. – с. 113-114.

22 Коркин И.Ю., Петрова Т.В., Тихонов А.Ю. Новый подход к выявлению аппаратной виртуализации в компьютерных системах. XIV Международная телекоммуникационная конференция студентов и молодых учёных «Молодёжь и наука». Тезисы докладов. Ч. 3. М.:НИЯУ МИФИ, 2010. – с. 241-242.

23 Коркин И.Ю. Метод выявления аппаратной виртуализации в компьютерных системах на основе механизма кэширования // Безопасность Информационных Технологий. 2011, №1. – с. 101-103.

24 Коркин И.Ю. Статистическая идентификация режимов работы компьютерных систем. Сб. науч. тр.. XV конференция «Телекоммуникации и новые информационные технологии в образовании». М.:НИЯУ МИФИ, 2011. – с. 163-165.

25 Коркин И.Ю. Обнаружение вложенных мониторов виртуальных машин методами математической статистики. В сб. материалов XX Общероссийской научно-технической конференции «Методы и Технические Средства Обеспечения Безопасности Информации». СПб.: 2011. – с. 146-147.

26 Коркин И.Ю. Новые статистические показатели и методы для обнаружения мониторов виртуальных машин в компьютерных системах // Естественные и технические науки. 2011, № 4. – с. 498-502.

27 Корнфельд М.И. Погрешность и надежность простейших экспериментов // Успехи физических наук. 1965 — Т.85. Выпуск 3. с. 533-542.

28 Кухар А. Rootkit: новые угрозы и средства борьбы с ними. – http://itc.ua/articles/rootkit_novye_ugrozy_i_sredstva_borby_s_nimi_25721

29 Лукацкий А. Что будет актуальным в области ИБ в ближайшие годы? – http://www.ruscrypto.org/netcat_files/File/infosec.2008.001.zip

30 Луценко А. Красная пилюля - аппаратная виртуализация в контексте информационной безопасности. – http://www.vmgu.ru/articles/red-pill-virtualization-security

31 Марков А.С., Пугачев И.Б. Программный метод обеспечения безопасности загрузки операционной среды. – http://www.npo-echelon.ru/about/articles/mdz.php

32 Научно-исследовательский институт стандартизации и унификации – http://www.niisu.com

33 Никитин Е., Шрамко В. Всегда ли на замке? Как обезопасить компьютер модулем доверенной загрузки? – http://samag.ru/archive/article/1068

34 Николаев А. С. Особенности обеспечения безопасности информации с помощью монитора виртуальных машин // Безопасность информационных технологий, 2010. № 1. – с. 97-98.

35 Рагойша Г.А. Хронометрирование и синхронизация событий в компьютеризованном эксперименте – http://pdeis.at.tut.by/rdtsc.htm

36 Розенберг Г.С., Шитиков В.К., Брусиловский П.М. Экологическое прогнозирование (функциональные предикторы временных рядов) – Тольятти: ИЭВБ РАН, 1994. – 228 с.

37 Руководящий документ. Концепция защиты средств вычислительной техники и автоматизированных систем от несанкционированного доступа к информации.

38 Русаков В. Возможно ли заразить BIOS? – http://www.infosecurity.ru/_gazeta/content/110916/exp1.shtml

39 Светозаров В.В. Элементарная обработка результатов измерений. Учебное пособие. – М.: МИФИ, 1983. – 52 с.

40 Семенихин С.В., Ревякин В.А.,. Ананьев Л.И. и др. ОС Linux и режим реального времени. – www.mcst.ru/doc/semenih_090610.doc

41 Сидорова М., Луценко А. Миф 3. Концепции работы аппаратной виртуализации у разных производителей - одинаковы!? – http://www.risspa.ru/node/313

42 Силаков Д. Использование аппаратной виртуализации в контексте информационной безопасности // Труды Института системного программирования РАН, 2011 Т. 20. – с. 25-36.

43 СмартНЭТ Групп. Виртуализация на основе процессоров семейства x86 (аппаратная поддержка). – http://www.smartnet.ru/node/496

44 Сорокина С.И., Тихонов А.Ю., Щербаков А.Ю. Программирование драйверов и систем безопасности. СПб.: БХВ-Петербург, 2003. – 256 с.

45 Страуструп Б. Язык программирования C++. – 3-е изд. – СПб.: Невский диалект, М.: Бином, 1999. – 991 с.

46 Таненбаум Э. Архитектура компьютера. 5-е изд. – СПб.: Питер, 2007. – 844 с.

47 Таненбаум Э. Современные операционные системы. 3-е издание. СПб.: Питер, 2010. – 1120 с.

48 Тритон универсальный программатор. – http://www.triton-prog.ru/

49 Тюрин Ю.Н., Макаров А.А. Анализ данных на компьютере, М.: Инфра-М, Финансы и статистика, 1995. – 384 с.

50 Фог А. Оптимизация для процессоров семейства Pentium: 22. Команды передачи управления и переходов. – http://www.wasm.ru/article.php?article=1010023

51 Харари Ф. Теория графов. – М.: Мир, 1973. – 300 с.

52 Центральный научно-исследовательский институт химии и механики. – http://www.cniihm.ru.

53 Чекин Р.Н. Современные угрозы безопасности обработки информации со стороны встроенного программного обеспечения BIOS. Пенза: ТУСУР, 2010. – с. 54-56.

54 Щиголев Б.М. Математическая обработка наблюдений. – М.: Физматгиз, 1962. – 344 с.

55 R_T_T. Китайские закладки. Непридуманная история о виртуализации, безопасности и шпионах // Хакер. 2011, № 12 (155) – с. 30-35.

56 ACPI. Advanced Configuration & Power Interface. – http://www.acpi.info/

57 Acronis Inc – www.acronis.com

58 Adams K. BluePill detection in two easy steps. – http://x86vmm.blogspot.com/2007/07/bluepill-detection-in-two-easy-steps.html

59 Allen M. Kolmogorov-Smirnov Test for Discrete Distributions. – http://www.stormingmedia.us/54/5484/A548420.html

60 AMD64 Architecture Programmer's Manual Volume 2: System Programming. – support.amd.com/us/Processor_TechDocs/APM_V2_24593.pdf

61 AMD Revision Guide for AMD NPT Family 0Fh Processors. – http://support.amd.com/us/Processor_TechDocs/33610.pdf

62 Athreya M. Subverting linux on-the-fly using hardware virtualization technology. – http://arch.ece.gatech.edu/pub/athreya.pdf

63 Barbosa E. Detecting Blue Pill. SyScan Conference, 2007.

64 Becher M., Dornseif M., Klein C. FireWire: all your memory are belong to us. – http://simson.net/ref/2005/2005-firewire-cansecwest.pdf

65 Bing S. Software virtualization based rootkits. Black Hat EU, 2007

66 Blue Pill Project. – www.bluepillproject.org

67 Bluepillstudy. – http://bluepillstudy.googlecode.com

68 Blunden B. The Rootkit Arsenal: Escape and Evasion in the Dark Corners of the System. Jones & Bartlett Publishers, 2009 – 908 p.

69 Boileau A. Hit by a Bus: Physical Access Attacks with Firewire. – http://www.security-assessment.com/files/presentations/ab_firewire_rux2k6-final.pdf

70 Brian D. A Hardware-Based Memory Acquisition Procedure for Digital Investigations. – http://grandideastudio.com/wp-content/uploads/tribble_paper.pdf

71 Broersma M. Rootkits Evade Hardware Detection – http://www.pcworld.com/article/129601/rootkits_evade_hardware_detection.html

72 Bulygin Y. CPU side-channels vs. virtualization malware: the good, the bad, or the ugly. – http://c7zero.info

73 Bulygin Y. Chipset based approach to detect virtualization malware a.k.a. DeepWatch. Black Hat USA, 2008.

74 Chen H., Zhang F., Chen C., Yang Z., Chen R., Zang B. Tamper-Resistant Execution in an Untrusted Operating System Using a Virtual Machine Monitor. – http://ppi.fudan.edu.cn/system/publications/paper/chaos-ppi-tr.pdf

75 Cooper D., Polk W., Regenscheid A., Souppaya M. BIOS Protection Guidelines. Recommendations of the National Institute of Standards and Technology, Gaithersburg, MD 20899-8930.

76 Daniel P., Bovet M. Understanding the Linux Kernel 3rd Edition. – http://book.opensourceproject.org.cn/kernel/kernel3rd/index.html?page=opensource/0596005652/understandlk-chp-6-sect-1.html

77 Datta A., Franklin J., Garg D., Kaynar D. A Logic of Secure Systems and its Application to Trusted Computing. – http://www.cs.cmu.edu/~dg/papers/cmu-cylab-09-001.pdf

78 David F., Chan E., Carlyle J., Campbell R. Cloaker: Hardware Supported Rootkit Concealment. In IEEE Symposium on Security and Privacy, 2008.

79 Davies F. Hypervisor Malware. Honors Thesis. – http://ivanlef0u.fr/repo/windoz/hvm/ThesisB.pdf

80 Davis M., Bodmer S., LeMasters A. Hacking Exposed: Malware & Rootkits Secrets & Solutions. The McGraw-Hill Companies, 2010. – 400 p.

81 Derock A. HVM Virtual Machine Monitor, A Powerful Concept for Forensic and Anti-Forensic. – http://www.recherche.eu/data/derock_hacklu09.pdf

82 Desnos A., Filiol E. Detection of an HVM rootkit (aka BluePill-like). – http://www.esiea-recherche.eu/~desnos/papers/hyp.pdf

83 Dinaburg A., Royal P., Sharif M., Lee W. Ether: Malware Analysis via Hardware Virtualization Extensions. ACM Computer and Communications Security Conference'08, Alexandria, Virginia, USA

84 Douglas H., Gehrmann C. Secure Virtualization and Multicore Platforms State-of-the-Art report. SICS Technical Report, T2009:14A, ISSN: 1100-3154

85 Duflot L., Etiemble D., Grumelard O. Using CPU System Management Mode to Circumvent Operating System Security Functions. – http://www.ssi.gouv.fr/fr/sciences/fichiers/lti/cansecwest2006-duflot-paper.pdf

86 Embleton S., Sparks S., Zou C. SMM Rootkits: A New Breed of OS Independent Malware. // Proc. of the 4th international conference on Security and privacy in communication networks. Istanbul, Turkey, 2008. Article #11.

87 Embleton S., Sparks S. SMM Rootkits. Black Hat USA 2008.

88 Embleton S. The VMM framework. – http://www.mobile-download.net/tools/software/vmxcpu.rar

89 Erez M. Branch Prediction. – http://cva.stanford.edu/classes/ee482a/scribed/lect03.pdf

90 Facchinetti S., Chiodini P. Exact and Approximate Critical Values of Kolmogorov-Smirnov Test for Discrete Random Variables. – http://www.sis-statistica.it/files/pdf/atti/rs08_spontanee_2_3.pdf

91 Ferrie P. Attacks on More Virtual Machine Emulators. – http://www.symantec.com/avcenter/reference/Virtual_Machine_Threats.pdf

92 Fisher D. Researchers unveil persistent BIOS attack methods. – http://threatpost.com/en_us/blogs/researchers-unveil-persistent-bios-attack-methods-031909

93 Fisher-Ogden J. Hardware Support for Efficient Virtualization. Intel Technology Journal (2006) Volume: 17, Issue: 03, Publisher: Citeseer.

94 Franklin J., Luk M., McCune J. Detecting the Presence of a VMM through Side-Effect Analysis. CMU-CS-05-201, Pittsburgh, PA 15213.

95 Fritsch H. Analysis and detection of virtualization-based rootkits. – http://www.mnm-team.org/pub/Fopras/frit08/PDF-Version/frit08.pdf

96 Fog A. Instruction tables. Lists of instruction latencies, throughputs and micro operation breakdowns for Intel, AMD and VIA CPUs. – http://www.agner.org/optimize/instruction_tables.pdf

97 Fog A. The microarchitecture of Intel, AMD and VIA CPUs. An optimization guide for assembly programmers and compiler makers. – http://www.agner.org/optimize/microarchitecture.pdf

98 Garfinkel T., Adams K., Warfield A., Franklin J. Compatibility is Not Transparency: VMM Detection Myths and Realities. In the 11th Workshop on Hot Topics in Operating Systems (HOTOS-X), 2007.

99 Godiyal A., Nguyen A., Schear N. A Lightweight Hypervisor for Malware Analysis. – http://ivanlef0u.fr/repo/todo/HyperVisorMalware.pdf

100 Goel S. Digital Forensics and Cyber Crime: First International ICST Conference, ICDF2C 2009, Albany, NY, USA, 2009, Springer 2010

101 Greenstadt R. Virtualization and Security. – http://www.coursehero.com/file/2781942/CS475-09-05

102 Hipp R. SQLite Home Page. – http://www.sqlite.org

103 Hoglund G., Butler J. ROOTKIT: Advanced 2nd Generation Digital Weaponry. Black Hat Training Course. – http://www.blackhat.com/html/bh-dc-07/train-bh-dc-07-gh-adv.html

104 Hyper-V. – http://www.microsoft.com/windowsserver2008/ru/ru/virtualization/hyperv.aspx

105 IBM Research. Virtual Trusted Platform Module. – https://researcher.ibm.com/researcher/view_project.php?id=2850

106 Intel Corporation. IA-PC HPET (High Precision Event Timers) Specification. – http://www.intel.com/hardwaredesign/hpetspec_1.pdf

107 Intel Corporation. Methods and systems to manage machine state in virtual machine operations. Patent № 7793286, Publication № US 2004/0123288 A1.

108 Intel Corporation . Intel 64 and IA-32 Architectures Software Developer's Manual Volume 3B: System Programming Guide, Part 2.

109 Intel Corporation. Using the RDTSC Instruction for Performance Monitoring. – http://www.ccsl.carleton.ca/~jamuir/rdtscpm1.pdf.

110 Intel Corporation. Intel 64 and IA-32 Architectures Application Note TLBs, Paging-Structure Caches, and Their Invalidation.

111 Jiang X. Enabling Internet worms and malware investigation and defense using virtualization. – http://docs.lib.purdue.edu/dissertations/AAI3251634

112 Jones S. Implicit operating system awareness in a virtual machine monitor. – research.cs.wisc.edu/adsl/Publications/stjones-thesis07.ps

113 King S., Chen P., Wang Y., Verbowski C., Wang H., Lorch J. SubVirt: Implementing malware with virtual machines, Proceedings of the 2006 IEEE Symposium on Security and Privacy, 2006.

114 Korkin I. Strong approach to hardware-VM rootkits detection. Hakin9 Extra Magazine, English Edition. Issue 6/2011. – pp. 30-35. ISSN 1733-7186.

115 Lauradoux C. Detecting Virtual Rootkits with Cover Channels. – http://perso.citi.insa-lyon.fr/claurado/publis/Lau08a.pdf

116 Lawson N. Don’t Tell Joanna. The Virtualized Rootkit Is Dead. Black Hat, 2007, USA

117 Liliputing. Intel: Netbooks helping notebook sales match desktop sales. – http://liliputing.com/2008/08/intel-netbooks-helping-notebook-sales-match-desktop-sales.html

118 Lioy A., Ramunno G., Vernizzi D. Trusted-Computing Technologies for the Protection of Critical Information Systems. Springer Berlin. Volume 54 / 2009

119 Matlab documentation. Statistics Toolbox. kstest2 - Two-sample Kolmogorov-Smirnov test. – http://www.mathworks.com/help/toolbox/stats/kstest2.html

120 Matlab. Use MATLAB from Microsoft Excel. – http://www.mathworks.com/products/excellink

121 Matthieu S. Multi-Processors and KdVersionBlock. – http://www.msuiche.net/2009/01/05/multi-processors-and-kdversionblock

122 Medina P. Detecting Rootkits in Memory Dumps. – http://www.terena.org/activities/tf-csirt/meeting27/oesterberg-rootkits.pdf

123 Medley D. Captain USAF, Virtualization Technology Applied to Rootkit Defense. THESIS., AFIT/GCE/ENG/07-08

124 Miller C., Zovi D. Virtiol sourse code. – www.wiley.com/go/machackershandbook

125 Miller C., Zovi D. The Mac Hacker's Handbook. Wiley, 2009. – 284 p.

126 Mills D. Network Time Protocol. Request for Comments: 958. – http://www.ietf.org/rfc/rfc958.txt

127 Mills D. Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI. – http://tools.ietf.org/html/rfc2030

128 Moffie M. Investigating the utility of software semantics for host-based intrusion detection systems. Computer Engineering Dissertations. Paper 1. – http://hdl.handle.net/2047/d10018177

129 MSDN. DbgPrint. – http://msdn.microsoft.com/en-us/library/windows/hardware/ff543632(v=vs.85).aspx

130 MSDN. ZwMapViewOfSection Routine. – http://msdn.microsoft.com/en-us/library/ff566481(v=vs.85).aspx

131 Myers M., Youndt S. An Introduction to Hardware-Assisted Virtual Machine (HVM) Rootkits. – http://www.megasecurity.org/papers/hvmrootkits.pdf

132 Naraine R. Hardware-based rootkit detection proven unreliable. – http://www.zdnet.com/blog/security/hardware-based-rootkit-detection-proven-unreliable/109

133 Nguyen A., Schear N., Jung H., Godiyal A., King S. MAVMM: Lightweight and Purpose Built VMM for Malware Analysis. Computer Security Applications Conference, 2009. ACSAC '09. Annual. – pp. 441-450.

134 Nie C. Dynamic Root of Trust in Trusted Computing. – http://www.tml.tkk.fi/Publications/C/25/papers/Nie_final.pdf

135 North Security Labs. Hypersight Rootkit Detector. – http://northsecuritylabs.com

136 Paoloni G. How to Benchmark Code Execution Times on Intel IA-32 and IA-64 Instruction Set Architectures. – http://download.intel.com/embedded/software/IA/324264.pdf

137 Petroni N., Fraser T., Molina J., Arbaugh W. Copilot – a Coprocessor-based Kernel Runtime Integrity Monitor. – http://www.jesusmolina.com/publications/2004NPTF.pdf

138 Popek G., Goldberg R. Formal Requirements for Virtualizable Third Generation Architectures. Communications of the ACM. Volume 17, Issue 7, 1974. – pp. 412-421.

139 Poroshyn R. Stuxnet: The Contemporary Way to Destroy a Uranium Processing Facility. –

http://www.associatedcontent.com/article/7850990/stuxnet_the_contemporary_way_to_destroy.html?cat=15

140 Ports D., Garfinkel T. Towards Application Security on Untrusted Operating Systems. In Proceedings of the 3rd Workshop on Hot Topics in Security, San Jose, CA, USA, 2008, USENIX.

141 Raffetseder T., Kruegel C., Kirda E. Detecting system emulators, Information Security, Vol. No. 2007, pp. 1-18.

142 Ramachandran M. New client virtualization usage models using Intel® Virtualization Technology // Intel Technology Journal. Volume 10, Issue 3, 2006.

143 Ramos J. Security Challenges with Virtualization. – http://docs.di.fc.ul.pt/jspui/bitstream/10455/3282/1/Thesis-JRamos-FCUL.pdf

144 Riemersma T. Periodic Interrupts with the Real Time Clock. – http://www.compuphase.com/int70.txt

145 Robin J., Irvine C. Analysis of the Intel Pentium's Ability to Support a Secure Virtual Machine Monitor. SSYM'00 Proceedings of the 9th conference on USENIX Security Symposium - Volume 9.

146 Rutkowska J. Understanding Stealth Malware Training. – http://www.invisiblethingslab.com/itl/Services.html

147 Rutkowska J. Beyond The CPU: Defeating Hardware Based RAM Acquisition (part I: AMD case). Black Hat DC, 2007.

148 Rutkowska J. Virtualization Detection vs. Blue Pill Detection. – http://theinvisiblethings.blogspot.com/2007/08/virtualization-detection-vs-blue-pill.html

149 Rutkowska J. Virtualization – the other side of the coin. – http://www.invisiblethings.org/papers/NLUUG-virtualization.ppt

150 Rutkowska J. Introducing Blue Pill. – http://theinvisiblethings.blogspot.com/2006/06/introducing-blue-pill.html

151 Rutkowska J. Security Challenges in Virtualized Environments. RSA Conference, San Francisco, 2008.

152 Rutkowska J., Tereshkin A. IsGameOver(). Anyone? Black Hat Briefings, USA, 2007.

153 Rutkowska J., Tereshkin A. Bluepilling the Xen Hypervisor. Black Hat USA, 2008.

154 Sahgal N., Rodgers D. Understanding Intel® Virtualization Technology (VT). – http://www.docstoc.com/docs/522280/Understanding-Intel-Virtualization-Technology

155 Sailer R., Valdez E.,; Jaeger T.,; Perez R.,; Doorn L., Griffin G.,; Berger S. sHype: Secure Hypervisor Approach to Trusted Virtualized Systems. IBM Research Report RC23511 (W0502-006), 2005.

156 Sallam A. The truths and myths about Blue Pill and virtualized malware. – http://blogs.mcafee.com/mcafee-labs/the-truths-and-myths-about-blue-pill-and-virtualized-malware

157 Seshadri A., Luk M., Qu N., Perrig A. SecVisor: A Tiny Hypervisor to Provide Lifetime Kernel Code Integrity for Commodity OSes. – http://www.sosp2007.org/papers/sosp079-seshadri.pdf

158 Shafranovich Y. Common Format and MIME Type for Comma-Separated Values (CSV) Files. – http://tools.ietf.org/html/rfc4180

159 Sharif M., Lee W., Cui W. Secure In-VM Monitoring Using Hardware Virtualization. ACM Computer and Communications Security Conference'09, Chicago, Illinois, USA.

160 Shields T. Survey of Rootkit Technologies and Their Impact on Digital Forensics. – http://www.donkeyonawaffle.org/misc/txs-rootkits_and_digital_forensics.pdf

161 Steo . Vitriol: The VT-x Rootkit – Another VM Rootkit. – http://www.antirootkit.com/blog/category/vm-rootkits

162 Sun Microsystems. – www.virtualbox.org

163 Suzaki K., Iijima K., Yagi T., Anh N., Nakamura M., Muhetoh S.TPM + Internet Virtual Disk + Platform Trust Services = Internet Client. – http://openlab.jp/oscircular/ASPLOS08-poster-leaflet.pdf

164 VMware Inc. – www.vmware.com

165 VMware. Timekeeping in VMware Virtual Machines, VMware® ESX® 4.0/ESXi™ 4.0, VMware Workstation 7.0. – http://www.vmware.com/files/pdf/Timekeeping-In-VirtualMachines.pdf

166 Wang Z., Jiang X. HyperSafe: A Lightweight Approach to Provide Lifetime Hypervisor Control-Flow Integrity. Proceedings of the 31st IEEE Symposium on Security & Privacy, 2010.

167 Wassenberg J. Timing Pitfalls and Solutions. – http://algo2.iti.kit.edu/wassenberg/timing/timing_pitfalls.pdf

168 Williams P., Spafford E. CuPIDS: An exploration of highly focused, co-processor-based information system protection. The International Journal of Computer and Telecommunications Networking archive. Volume 51 Issue 5, 2007.

169 Wojtczuk R., Rutkowska J. Attacking Intel® Trusted Execution. Technology Black Hat DC, 2009.

170 Wojtczuk R., Rutkowska J., Tereshkin A. Another Way to Circumvent Intel Trusted Execution Technology. – http://invisiblethingslab.com/resources/misc09/Another%20TXT%20Attack.pdf

171 Wojtczuk R. Subverting the Xen hypervisor. Black Hat USA, 2008.

172 Zmudzinski K. Methods for selecting cores to execute system management interrupts. Application number: 11/966,341 Publication number: US 2009/0172229 A1.

173 Zovi D. Hardware Virtualization-Based Rootkits. Black Hat USA, 2006.

Приложение А. Фрагмент исходного кода программы для построения графиков теоретических распределений

Исходный текст написан на языке MATLAB. Ниже приведён фрагмент листинга исходного кода программы построения графиков распределения в случаях отсутствия ПОАВ, присутствия с использованием и без компрометации счётчика.

```
NumExp=100;

for (i=1:NumExp)
   x(i,1)=i;
end

t=2200;
p=0.0005;
n0=10;
ns=200;
nv=200;

for i=(1:NumExp)
   tau=binornd(n0,p);
   Tnv(i,1)=(n0+ns*tau)*t;
end


for i=(1:NumExp)
   tau=binornd(n0+n0*nv,p);
   Trv(i,1)=(n0+n0*nv+ns*tau)*t;
end


for i=(1:NumExp)
   tau=binornd(n0+n0*nv,p);
   Tcv(i,1)=(n0+n0*nv+ns*tau-200*n0)*t;
end

% 1. Построение точечного графика
figure('Name','Точечные графики в случае ПР и ОТ','NumberTitle','off');
xlabel('Номера измерений', 'FontName', 'Times New Roman', 'FontSize', 16);
ylabel('Время выполнения трассы в тактах', 'FontName', 'Times New Roman', 'FontSize', 16);
hold on; grid on

plot(x,Tnv,'ob', 'MarkerSize', 5, 'LineWidth', 3 );
hold on;
plot(x,Trv,'xk');
hold on;
plot(x,Tcv,'xr');
```

Приложение Б. Фрагмент исходного кода модуля обработки матрицы и расчёта статистических характеристик

Исходный текст написан на языке Matlab.

```
% How to use: clc; clear r; r = disp_mom_filtr_jump(TR, 300); clear TR

function [res_matr] = disp_mom_filtr_jump(T_matrix, Jump_val)

res_matr = 0;
mom4_disp = 0;
% Расчёт значений по столбцам (начало)
for i=1:length(T_matrix(1,:))
   res_matr(1,i) = mean(T_matrix(:,i));

   val = 0; val = var_series_theshold(T_matrix(:,i), 0.00);
   res_matr(1 + 1, i) = length(val(:,1));

   val = 0; val = var_series_theshold(T_matrix(:,i), 0.02);
   res_matr(1 + 2, i) = length(val(:,1));

   val = 0; val = var_series_theshold(T_matrix(:,i), 0.05);
   res_matr(1 + 3, i) = length(val(:,1));

   val = 0; val = var_series_theshold(T_matrix(:,i), 0.10);
   res_matr(1 + 4, i) = length(val(:,1));
   fprintf('%i,', i ); fprintf('%i,', val(:,1) );  fprintf('\n');

   val = 0; val = var_series_theshold(T_matrix(:,i), 0.15);
   res_matr(1 + 5, i) = length(val(:,1));

   val = 0; val = var_series_theshold(T_matrix(:,i), 0.20);
   res_matr(1 + 6, i) = length(val(:,1));

   mom4_disp = mom4_disp_column_jump(T_matrix(:,i), 0.00, Jump_val);
   res_matr(7 + 1, i) = mom4_disp(1);
   res_matr(7 + 6 + 1, i) = mom4_disp(2);

   mom4_disp = mom4_disp_column_jump(T_matrix(:,i), 0.02, Jump_val);
   res_matr(7 + 2, i) = mom4_disp(1);
   res_matr(7 + 6 + 2,i) = mom4_disp(2);

   mom4_disp = mom4_disp_column_jump(T_matrix(:,i), 0.05, Jump_val);
   res_matr(7 + 3, i) = mom4_disp(1);
   res_matr(7 + 6 + 3, i) = mom4_disp(2);

   mom4_disp = mom4_disp_column_jump(T_matrix(:,i), 0.10, Jump_val);
   res_matr(7 + 4, i) = mom4_disp(1);
   res_matr(7 + 6 + 4, i) = mom4_disp(2);

   mom4_disp = mom4_disp_column_jump(T_matrix(:,i), 0.15, Jump_val);
```

```
    res_matr(7 + 5, i) = mom4_disp(1);
    res_matr(7 + 6 + 5, i) = mom4_disp(2);

    mom4_disp = mom4_disp_column_jump(T_matrix(:,i), 0.20, Jump_val);
    res_matr(7 + 6, i) = mom4_disp(1);
    res_matr(7 + 6 + 6, i) = mom4_disp(2);

    clear mom4_disp
end
% Расчёт значений по столбцам (конец)

% нахождение min и max по столбцам
min_max_res_clumn = min_max(res_matr);
n_colunms = length(res_matr(1,:)); % число столбцов в матрице
n_rows = length(res_matr(:,1)); % число строк в матрице

for i=1:n_rows
    res_matr(i, n_colunms + 1 ) = min_max_res_clumn(i,1);
    res_matr(i, n_colunms + 2 ) = min_max_res_clumn(i,2);
    res_matr(i, n_colunms + 3 ) = mean(res_matr(i, 1:10));
end


% Расчёт значений по всей матрице (начало)
T_column_full = [
    T_matrix(:,1) ; T_matrix(:,2) ; T_matrix(:,3) ; T_matrix(:,4) ; T_matrix(:,5) ;
    T_matrix(:,6) ; T_matrix(:,7) ; T_matrix(:,8) ; T_matrix(:,9) ; T_matrix(:,10)
    ];

new_column = length(res_matr(1,:)) + 1; % номер столбца для вставки

res_matr(1, new_column) = mean(T_column_full);
val = 0; val = var_series_theshold(T_column_full, 0.00);
res_matr(1 + 1, new_column) = length(val(:,1));

val = 0; val = var_series_theshold(T_column_full, 0.02);
res_matr(1 + 2, new_column) = length(val(:,1));

val = 0; val = var_series_theshold(T_column_full, 0.05);
res_matr(1 + 3, new_column) = length(val(:,1));

val = 0; val = var_series_theshold(T_column_full, 0.10);
res_matr(1 + 4, new_column) = length(val(:,1));
fprintf('%i,', new_column ); fprintf('%i,', val(:,1) );  fprintf('\n');

val = 0; val = var_series_theshold(T_column_full, 0.15);
res_matr(1 + 5, new_column) = length(val(:,1));

val = 0; val = var_series_theshold(T_column_full, 0.20);
res_matr(1 + 6, new_column) = length(val(:,1));

mom4_disp = mom4_disp_column_jump(T_column_full, 0.00, Jump_val);
```

```
res_matr(7 + 1, new_column) = mom4_disp(1);
res_matr(7 + 6 + 1, new_column) = mom4_disp(2);

mom4_disp = mom4_disp_column_jump(T_column_full, 0.02, Jump_val);
res_matr(7 + 2, new_column) = mom4_disp(1);
res_matr(7 + 6 + 2, new_column) = mom4_disp(2);

mom4_disp = mom4_disp_column_jump(T_column_full, 0.05, Jump_val);
res_matr(7 + 3, new_column) = mom4_disp(1);
res_matr(7 + 6 + 3, new_column) = mom4_disp(2);

mom4_disp = mom4_disp_column_jump(T_column_full, 0.10, Jump_val);
res_matr(7 + 4, new_column) = mom4_disp(1);
res_matr(7 + 6 + 4, new_column) = mom4_disp(2);

mom4_disp = mom4_disp_column_jump(T_column_full, 0.15, Jump_val);
res_matr(7 + 5, new_column) = mom4_disp(1);
res_matr(7 + 6 + 5, new_column) = mom4_disp(2);

mom4_disp = mom4_disp_column_jump(T_column_full, 0.20, Jump_val);
res_matr(7 + 6, new_column) = mom4_disp(1);
res_matr(7 + 6 + 6, new_column) = mom4_disp(2);
% Расчёт значений по всей матрице (конец)

% Расчёт центра тяжести по строкам (начало)
columns = length(res_matr(1,:)); % число столбцов в матрице

for i=1:length(res_matr(:,1))
   ct = ct_vector(res_matr(i, 1:10));
%     res_matr(i, columns + 1) = ct(1); % декартовые коорд.
%     res_matr(i, columns + 2) = ct(2); % декартовые коорд.
   res_matr(i, columns + 1) = ct(3); % полярные коорд. (ро)
   res_matr(i, columns + 2) = ct(4); % полярные коорд. (фи)
   clear ct
end
% Расчёт центра тяжести по строкам (конец)



%       Функция реализует фильтр низких частот для входного столбца
%       T - входной столбец
%       threshold - пороговое значение частотности: if (series(i) > threshold) { add() }
%
%   Если пороговое значение указано - 0, то столбец не фильтруется
%
function [var_series] = var_series_theshold(T, threshold)

%   1. Определение частот элементов вектора от 0 до tmax [doc tabulate в
% Help и Statistics на сайте
% http://www.nsu.ru/matlab/MatLab_RU/default.asp.htm]

% t - вектор столбец длительности выполнения трассы, передаваемый из Excel-таблиц
```

```
%     с опытными данными Иг, Ал и др.
% 2. Выявление и удаление редких выбросов из исх. ряда
%    ==========================================
%   Построение вариационного ряда
%   ************************************************************
var_series_with_0 = my_tabulate(T);
%   Удаление из массива tt несуществующих в массиве t элементов
%    в случае, если даты - целые числа (если дробные, в
%     ф. VariazionRyad_Poligon.m)
length_of_var = length(var_series_with_0(:,1));
var_series_without_0 = [0, 0];
k=0;
for i=1:length_of_var
   if var_series_with_0(i,3) ~= 0
      k = k + 1;
      var_series_without_0(k,:) = [var_series_with_0(i), var_series_with_0(i,2)];
   end
end
clear k i
clear var_series_with_0


%   3. Определение относительной частоты вариантов длительности выполнения трассы
%   Во втором столбце var_series_without_0 находятся
relative_freq = var_series_without_0(:,2) ./ sum(var_series_without_0(:,2));
var_series_without_0 = [var_series_without_0, relative_freq];
clear relative_freq

length_of_var = length(var_series_without_0(:,1));
var_series_without_0_filtr = [0, 0, 0];

if (threshold > 0)
   k = 0;
   for i=1:length_of_var
      if var_series_without_0(i,3) >  threshold
         k = k + 1;
         var_series_without_0_filtr(k,:) = [var_series_without_0(i,1), var_series_without_0(i,2),
var_series_without_0(i,3)];
      end
   end
else
  var_series_without_0_filtr = var_series_without_0;
end

clear length_of_var
% Трёхмерный массив var_series_without_0 содержит в 1-м столбце значения длительности
выполнения трассы t_i,
% во 2-м - их частоту m_i, в третьем - относительную частоту p_i

var_series = var_series_without_0_filtr;
```

```
function [ var_ryad ] = my_tabulate( T )
%MY_TABULATE Summary of this function goes here
%   Detailed explanation goes here

var_ryad = [0 0];
len_T = length(T);
for i=1:len_T
   result = my_tabulate_check_presence(T(i), var_ryad);
   if result(2)
      % увеличить номер встречаемости
      var_ryad(result(1), 2) = var_ryad(result(1), 2) + 1;
   else
      % добавить элемент в вариационный ряд
      len_var = length(var_ryad(:,1));
      var_ryad(len_var + 1, 1) = T(i);
      var_ryad(len_var + 1, 2) = 1;
   end
end

var_ryad = sortrows(var_ryad);

var_ryad (:,3) = var_ryad(:,2)./len_T;

end
```

my_tabulate

```
%       Функция реализует фильтр низких частот для входного столбца с учётом
%       порога частотности и возвращает отфильтрованный столбец
%
% ПРИМЕР ИСПОЛЬЗОВАНИЯ:
%
% 1) Входные значения:
% Tr3 - столбец значений длительности выполнения трассы, переданный из Excel
% 0.02 - пороговое значение
%
% 2) Вызов функции:
% t_res = column_filtr(Tr3, 0.02);
%
% 3) Возвращаемые данные
% t_res - столбец отфильтрованных значений
%
% В результате работы функции в области вывода Matlab будут указаны номера
% элементов-выбросов

function [T_filtr] = column_filtr(T_in, threshold_in)

% I) составляем вариационный ряд = вариант-частота встречаемости
var_series_with_0 = my_tabulate(T_in); %   Построение вариационного ряда
%   удаляем из массива нули
length_of_var = length(var_series_with_0(:,1));
```

```
var_series_without_0 = [0, 0];
k=0;
for i=1:length_of_var
   if var_series_with_0(i,3) ~= 0
      k = k + 1;
      var_series_without_0(k,:) = [var_series_with_0(i), var_series_with_0(i,2)];
   end
end
clear k i clear var_series_with_0 % удаляем переменные, после их использования

% II) правим вариационный ряд = вариант-относительная частота встречаемости
relative_freq = var_series_without_0(:,2) ./ sum(var_series_without_0(:,2));
var_series_without_0 = [var_series_without_0, relative_freq];
clear relative_freq % удаляем переменные, после их использования

% III) Копируем полученный вариационный ряд (var_series_without_0) в
% результирующий вар. ряд (var_series_without_0_filtr) с учётом фильтрации

var_series_without_0_filtr = [0, 0, 0];

threshold = threshold_in;

if (threshold > 0)
   k = 0;
   for i=1:length(var_series_without_0(:,1))
      if var_series_without_0(i,3) >  threshold
         k = k + 1;
         var_series_without_0_filtr(k,:) = [var_series_without_0(i,1), var_series_without_0(i,2),
var_series_without_0(i,3)];
      end
   end
else
  var_series_without_0_filtr = var_series_without_0;
end

clear i k % удаляем переменные после их использования

% IV) Копируем значения исходного ряда в выходной ряд с учётом вариационного
% ряда

T_filtr = 0; % выходной (отфильтрованный столбец)

if (threshold > 0)
   k = 0;
   for i=1:length(T_in)
      if check_presence(T_in(i), var_series_without_0_filtr(:,1))
         k = k + 1;
         T_filtr(k) = T_in(i);
      else
%           fprintf('Выброс в позиции %d (удалён)\n', i);
      end
   end
```

```
else
   T_filtr = T_in';
end

% V) Работа функции окончена, в выходном стоблце (T_filtr) содержатся
% отфильтрованные данные исходного столбца (T_in) с учётом порога
% (threshold)

T_filtr = T_filtr';


% функция расчёта центра тяжести входного вектора
% возвращает вектор [ A B C D ]
% где (A,B) - координаты центра тяжести
% C - длина радиус-вектора
% D - полярный угол в градусах
function [result_COLUMN] = ct_vector(T_vector)

T = round(T_vector);

var = var_series_theshold(T, 0);

% close(figure(1));
% plot(var(:,1), var(:,3), '-ok','LineWidth',3,'MarkerFaceColor','k','MarkerSize',4)
% grid on
% xlabel('\bfНомера вариантов длительности выполнения трассы, \iti')
% ylabel('\bfОтн. частота,  \itp_i')
% axis([0 max(var(:,1))+1 0 max(var(:,3))+0.05])
% hold on

 x = var(:, 1);
 y = var(:, 3);

 k = length(x); % число уровней в вариац. ряду

 if k==1 % Если в вар-ном ряду всего 1 уровень, относ. частота
          % которого равна 1 (т. е. полигон - это одна точка)
   x_ct=x(1); % абсцисса и  ордината ЦТП
   y_ct=y(1);
     % Вывод на график полигона полученного ЦТ
     %  plot(x_ct, y_ct, 'ob', 'MarkerFaceColor','b')

elseif k>1 % Если в вар-ном ряду более 1-го уровня
    % определение в цикле ЦТ (k-1) столбцов полигона
    for i=2:k

        if y(i-1)==y(i) % Если узлы (i-1)-го столбца полигона на одной
                        % горизонтали.
           % Определение координат ЦТ и площади этого столбца
           x_ct_st(i-1)=(x(i-1)+x(i))/2;
           y_ct_st(i-1)=y(i)/2;
           f_st(i-1)=(x(i)-x(i-1))*y(i);
```

```
        % Вывод на график полигона полученного ЦТ (i-1) столбца
        % plot(x_ct, y_ct, 'ob', 'MarkerFaceColor','b')

    elseif y(i-1)>y(i) % Если пик у столбца слева.
        %   Определение координат ЦТ и площади прямоугольника
        % (i-1)-го столбца
        x_pr=(x(i-1)+x(i))/2;
        y_pr=y(i)/2;
        f_pr=(x(i)-x(i-1))*y(i);
        % Определение координат ЦТ и площади треугольника столбца
        x_tr=x(i-1)+(x(i)-x(i-1))/3;
        y_tr=y(i)+(y(i-1)-y(i))/3;
        f_tr=(x(i)-x(i-1))*(y(i-1)-y(i))/2;
       % Определение координат ЦТ и площади (i-1)-го столбца полигона
        x_ct_st(i-1)=(x_pr*f_pr+x_tr*f_tr)/(f_pr+f_tr);
        y_ct_st(i-1)=(y_pr*f_pr+y_tr*f_tr)/(f_pr+f_tr);
        f_st(i-1)=f_pr+f_tr;
        % Вывод на график полигона полученного ЦТ (i-1)-го столбца
        %plot(x_ct, y_ct, 'ob', 'MarkerFaceColor','b')

    elseif y(i-1)<y(i) % Если пик у столбца справа.
        %   Определение координат ЦТ и площади прямоугольника
        % (i-1)-го столбца
        x_pr=(x(i-1)+x(i))/2;
        y_pr=y(i-1)/2;
        f_pr=(x(i)-x(i-1))*y(i-1);
        % Определение координат ЦТ и площади треугольника столбца
       x_tr=x(i-1)+(x(i)-x(i-1))*2/3;
       y_tr=y(i-1)+(y(i)-y(i-1))/3;
       f_tr=(x(i)-x(i-1))*(y(i)-y(i-1))/2;
      % Определение координат ЦТ и площади (i-1)-го столбца полигона
      x_ct_st(i-1)=(x_pr*f_pr+x_tr*f_tr)/(f_pr+f_tr);
      y_ct_st(i-1)=(y_pr*f_pr+y_tr*f_tr)/(f_pr+f_tr);
      f_st(i-1)=f_pr+f_tr;
      % Вывод на график полигона полученного ЦТ (i-1)-го столбца
      % plot(x_ct, y_ct, 'ob', 'MarkerFaceColor','b')
    end
  end

  % Координаты ЦТ суммы (k-1) обработанных столбцов полигона
  x_ct= (sum(x_ct_st.*f_st))/sum(f_st);
  y_ct= (sum(y_ct_st.*f_st))/sum(f_st);

end

% ЦТ в полярных координатах
Ro=sqrt(x_ct^2+y_ct^2); % длина радиуса-вектора
fi=180/pi*atan(y_ct/x_ct); % угол в градусах

% Для удобства экспорта полученных координат ЦТП в Excel-таблицы
CT=[x_ct; y_ct];
Polar=[Ro; fi];
```

```
result_COLUMN = [CT'   Polar'];

% plot(x_ct, y_ct,'ok','MarkerFaceColor','k','MarkerSize',8)
% pause


% Расчёт среднего арифметического и дисперсии по выбранному столбцу с
% учётом фильтрации по вар.ряду и найденных разрывов
%

% Пример использования:
%
% 1) Входные значения:
% TV3(:,3) - столбец значений длительности выполнения трассы, переданный из Excel
% 0.02 - пороговое значение уровня частотности
% 1000 - пороговое значение уровня разрыва (скачка)
%
% 2) Вызов функции:
% mean_disp_column_jump(TR3(:,1), 0.02, 1000);
%
% 3) Возвращаемые данные - дисперсия и момент 4-ого порядка для столбца
%
% В результате работы функции в области вывода Matlab будут
% выведены рассчитанные среднее арифметическое и дисперсия

function [res] = mom4_disp_column_jump(T_column_in, threshold_frequence_in,
threshold_jump_in)

% 1) Входной столбец T_column передаётся через Excel
T_column = T_column_in;

% 3) Фильтрация столбца с учётом заданного уровня фильтрации
% threshold_in = 0.05;
threshold_frequence = threshold_frequence_in;
T_column_filtr = column_filtr(T_column, threshold_frequence);
threshold_jump = threshold_jump_in;
mom4_column = 0;
disp_column = 0;

if length(T_column_filtr) > 1
   % 4) Определяем, присутствует ли разрыв, и для каждого участка рассчитываем
   % статистики

   diff_line = diff(T_column_filtr);


  fprintf('%i \n', max(abs(diff_line)) )


   i_start = 1;

%     plot (T_column_filtr, '.r');
```

```
%     hold on
%     plot (diff_line, 'o');

%      расчёт статистик
%     fprintf('\n/````````````````\\\n');
   i=2;
   num_jumps = 0;
   while (i<length(diff_line))
       if ( threshold_jump < abs( diff_line(i) ) )
%             && ( abs( diff_line(i) ) < 4000)
          i_finish = i-1;
          one_line = T_column_filtr(i_start:i_finish); % организуем выборку с начала(**условно)
и до разрыва
          koef = (i_finish-i_start)/length(diff_line); % весовой коэфициент
          mom4_column = mom4_column + ( sum((one_line - mean(one_line)).^4)./length(one_line)
) * koef; % центральный момент 4-го порядка
          disp_column = disp_column + ( std(one_line, 1)^2 ) * koef;
          i = i + 2;
          i_start = i;
          num_jumps = num_jumps + 1;
%           fprintf('\nНайден скачок № %i\n', i );
       end;
       i = i + 1;
   end;
%     fprintf('\n\\_____________/\n');

%     fprintf('%i ', num_jumps );

   i_finish = length(diff_line) + 1;
   one_line = T_column_filtr(i_start:i_finish); % организуем выборку с начала(**условно) и до
разрыва
   koef = (i_finish-i_start)/length(diff_line); % весовой коэфициент
   mom4_column = mom4_column + ( sum((one_line - mean(one_line)).^4)./length(one_line) ) *
koef; % центральный момент 4-го порядка
   disp_column = disp_column + ( std(one_line, 1)^2 ) * koef;
%     fprintf('\nУровень фильт. по частотности=%0.2f\nУровень величины скачка=%f\n',
threshold_frequence_in, threshold_jump_in );
%     fprintf('Среднее арифм.=%0.2f\nДисперсия=%0.2f\n', res(1), res(2));
else
%     fprintf('\nУровень фильт.=%f\n скачки не найдены\n', threshold);
end

res = [disp_column, mom4_column];
```