РОССИЙСКАЯ ФЕДЕРАЦИЯ

(19) **RU** (11) **2 522 019**(13) **C1**

(51) МПК
G06F 21/56 (2013.01)

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(21)(22) Заявка: **2012156443/08, 25.12.2012**

(24) Дата начала отсчета срока действия патента:
25.12.2012

Приоритет(ы):
(22) Дата подачи заявки: **25.12.2012**

(45) Опубликовано: **10.07.2014** Бюл. № **19**

(56) Список документов, цитированных в отчете о поиске: **US 2012/0036561 A1, 09.02.2012. US 2012/0096555 A1, 19.04.2012. US 8239939 B2, 07.08.2012. CN 102750475 A, 24.10.2012. US 7191441 B2, 13.03.2007. RU 2313126 C2, 20.12.2007**

Адрес для переписки:
125212, Москва, Ленинградское шоссе, 39а, стр. 3, ЗАО "Лаборатория Касперского", Управление по интеллектуальной собственности, Надежде Васильевне Кащенко

(72) Автор(ы):
Павлющик Михаил Александрович (CA)

(73) Патентообладатель(и):
Закрытое акционерное общество "Лаборатория Касперского" (RU)

RU 2522019 C1

(54) СИСТЕМА И СПОСОБ ОБНАРУЖЕНИЯ УГРОЗ В КОДЕ, ИСПОЛНЯЕМОМ ВИРТУАЛЬНОЙ МАШИНОЙ

(57) Реферат:

Изобретение относится к системам и способам обнаружения вредоносного программного обеспечения, код которого исполняется виртуальной машиной. Технический результат заключается в повышении безопасности виртуальной машины. Модифицируют код виртуальной машины для отслеживания исключений внутри виртуальной машины и управления виртуальной машиной. Отслеживают исключения внутри виртуальной машины. Останавливают виртуальную машину при наступлении исключения. Получают информацию о контексте исключения, содержащего данные о событиях виртуальной машины, приведших к этому исключению. Анализируют контекст исключения на наличие поведения, характерного для угрозы. Обнаруживают угрозу на основании анализа. 2 н. и 7 з.п. ф-лы, 4 ил.

RU 2522019 C1

Фиг.1

RU 2522019 C1

RUSSIAN FEDERATION

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY

(19) **RU** (11) **2 522 019**(13) **C1**

(51) Int. Cl.
G06F 21/56 (2013.01)

(12) ABSTRACT OF INVENTION

(21)(22) Application: **2012156443/08, 25.12.2012**

(24) Effective date for property rights:
25.12.2012

Priority:
(22) Date of filing: **25.12.2012**

(45) Date of publication: **10.07.2014** Bull. № **19**

Mail address:
125212, Moskva, Leningradskoe shosse, 39a, str. 3, ZAO "Laboratorija Kasperskogo", Upravlenie po intellektual'noj sobstvennosti, Nadezhde Vasil'evne Kashchenko

(72) Inventor(s):
Pavljushchik Mikhail Aleksandrovich (CA)

(73) Proprietor(s):
Zakrytoe aktsionernoe obshchestvo "Laboratorija Kasperskogo" (RU)

RU 2522019 C1

(54) SYSTEM AND METHOD OF DETECTING THREAT IN CODE EXECUTED BY VIRTUAL MACHINE

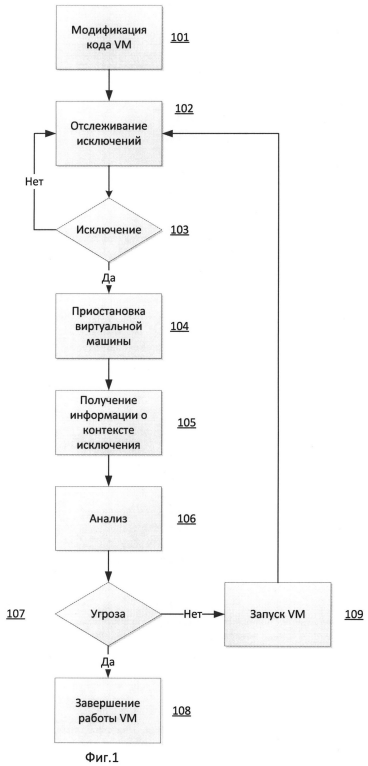
(57) Abstract:

FIELD: information technology.

SUBSTANCE: method includes modifying a virtual machine code for monitoring exceptions within the virtual machine and controlling the virtual machine; monitoring exceptions inside the virtual machine; shutting down the virtual machine when an exception occurs; obtaining exception context information containing data on virtual machine events leading to said exception; analysing the exception context for presence of behaviour typical of a threat; identifying the threat based on the analysis.

EFFECT: higher virtual machine safety.

9 cl, 4 dwg



Фиг.1

RU 2522019 C1

Область техники

Изобретение относится к системам и способам обнаружения вредоносного программного обеспечения, код которого исполняется виртуальной машиной.

Уровень техники

В настоящее время наблюдается резкое увеличение количества компьютерных угроз, программный код которых исполняется виртуальной машиной (например, Java Virtual Machine, Common Language Runtime, ActionScript Virtual Machine). Наибольшую опасность среди подобных угроз представляют эксплоиты.

Эксплоитом называется фрагмент программного кода или последовательность команд, использующая уязвимости в программном обеспечении и применяемая для проведения атаки на вычислительную систему. Опасность представляют не сами эксплоиты, а "полезная нагрузка", которую они с собой несут. Полезная нагрузка эксплоита - реализованный злоумышленником функционал, который при эксплуатации уязвимости на атакованной системе приводит к несанкционированной активации функционала. В качестве примера такого функционала можно привести загрузку вредоносного ПО. Эксплоиты могут использоваться как самостоятельно - для тестирования безопасности компьютерных систем, так и совместно с вредоносным программным обеспечением.

Из большого многообразия эксплоитов особо следует отметить те, для исполнения кода которых необходима виртуальная машина. Данный вид эксплоитов чаще всего используется для атак и является наиболее опасным по причине сложности его обнаружения.

Существуют два основных способа борьбы с подобного рода угрозами. Первый способ подразумевает устранение уязвимости, которая используется эксплоитом. Второй способ заключается в использовании специальных средств обнаружения эксплоитов и пресечения их активности. Средства могут быть как встроенные в само программное обеспечение (например, модель безопасности виртуальной машины Java), так и сторонние. Например, в патенте US 8239939 описана система, которая реализована в виде отдельного компонента, встраиваемого между клиентом и сервером, и путем модификации посылаемой клиенту информации позволяет снизить риск исполнения вредоносного кода браузером клиента.

Первый предложенный способ является самым надежным и логически верным, но у него есть два существенных недостатка. Во-первых, с момента обнаружения уязвимости до момента выпуска исправленной версии программного обеспечения проходит достаточно большой промежуток времени. Весь этот период пользователи уязвимого продукта остаются незащищенными. И, во-вторых, данный способ никак не защищает от уязвимостей "нулевого дня" (угроз, использующих ошибку или уязвимость в приложении или операционной системе и появившихся сразу после обнаружения данной уязвимости, но до выпуска соответствующего обновления).

Второй способ лишен указанных недостатков, но то, насколько он будет надежен, зависит от качества его технической реализации и, следует отметить, что подобные средства защиты сами могут быть уязвимы. Самыми распространенными решениями, использующими данный способ, являются обнаружение эксплоитов с использованием эвристических правил и сигнатурного анализа (анализа на идентичность анализируемого кода образцам кода известных компьютерных угроз); встроенные средства безопасности виртуальных машин. Применение сигнатур хорошо подходит для обнаружения уже известных эксплоитов. В случае если атакующий код будет модифицирован, данное решение окажется бесполезным. Указанного недостатка лишен эвристический метод,

но и он неэффективен при более глубокой модификации кода (например, путем обфускации), изменении алгоритма работы вредоносного кода, применении методов, препятствующих эмуляции кода.

Виртуальная машина - это программное обеспечение, которое формирует необходимый уровень абстракции, где достигается независимость от платформы, используемого оборудования. Виртуальные машины имеют собственные встроенные модели безопасности. Особо стоит отметить модель безопасности Java Virtual Machine (JVM), ее составляют четыре компонента -верификатор файлов классов (class file verifier), загрузчик классов (ClassLoader), менеджер безопасности (SecurityManager) и сама архитектура JVM. Поскольку байт-коды Java интерпретируются, то можно контролировать индексы массивов, что позволяет избежать переполнения буфера - самой распространенной и опасной ошибки периода выполнения ПО. Встроенные механизмы обработки исключительных ситуаций позволяют эффективно решать возникающие конфликты, а "сборщик мусора", который очищает неиспользуемую память, не дает возможности злоумышленнику просмотреть "мусорные" блоки памяти, которые могут содержать полезную информацию.

Менеджер безопасности, наиболее важный элемент в модели безопасности, это компонент, предоставляющий приложениям права в соответствии с установленной политикой безопасности. При возникновении ситуации, когда приложение пытается выполнить привилегированную операцию, менеджер безопасности проверяет права приложения и определяет легитимность подобного поведения. Менеджером безопасности по умолчанию является Java класс Java. long. SecurityManager, он включает в себя несколько методов для проверки критичных к политике безопасности операций.

Начиная с 2010 года, число направленных атак на JVM с применением эксплоитов резко возросло, и, как показали эти атаки, модель безопасности, предложенная создателями Java, на практике имеет серьезные недостатки в реализации, активно используемые злоумышленниками при проведении атак.

Таким образом, используемые в настоящее время способы обнаружения эксплоитов либо имеют ограниченную область применения, либо имеют недостатки, которые создают угрозу безопасности и в целом не обеспечивают надлежащей защиты.

Раскрытие изобретения

Настоящее изобретение предназначено для обнаружения эксплоитов и других угроз, программный код которых исполняется виртуальной машиной.

Технический результат заключается в повышении безопасности виртуальной машины за счет обнаружения угрозы в программном коде, выполняемом машиной, путем контроля модификаций кода виртуальной машины.

Способ обнаружения угроз в коде, исполняемом виртуальной машиной, в котором модифицируют код виртуальной машины для отслеживания исключений внутри виртуальной машины и управления виртуальной машиной; отслеживают исключения внутри виртуальной машины; останавливают виртуальную машину при наступлении исключения; получают информацию о контексте исключения, содержащего данные о событиях виртуальной машины, приведших к этому исключению; анализируют контекст исключения на наличие поведения характерного для угрозы; обнаруживают угрозу на основании анализа;

В частном случае к исключениям внутри виртуальной машины относятся критические события, которые могут вызвать нарушение правил, установленных политикой безопасности.

В другом частном случае модификацию кода виртуальной машины используют для

сбора статистической информации о выявленных угрозах.

Еще в одном частном случае модификация кода виртуальной машины осуществляется путем перегрузки методов класса.

Система обнаружения угроз в коде, исполняемом виртуальной машиной, содержит виртуальную машину, модифицированную модулем контроля, предназначенную для исполнения анализируемого кода; модуль контроля, предназначенный для модификации кода виртуальной машины, отслеживания исключений в виртуальной машине, сбора информации о контексте исключения, отправки информации о контексте исключения модулю анализа и остановки работы виртуальной машины по указанию модуля анализа; модуль анализа, предназначенный для анализа информации о контексте исключения, полученной от модуля контроля на наличие угрозы и обнаружения угроза по результатам анализа.

В частном случае реализации системы модуль контроля используется для сбора статистической информации о выявленных угрозах.

В другом частном случае реализации системы наполнение баз осуществляется на основании собранной статистики о выявленных угрозах.

Еще в одном частном случае реализации системы анализ информации о контексте исключения осуществляется путем сравнения с внешней обновляемой базой шаблонов.

В другом частном случае реализации системы модификация виртуальной машины осуществляется путем перегрузки методов класса.

Краткое описание чертежей

Сопровождающие чертежи включены для обеспечения дополнительного понимания изобретения и составляют часть этого описания, показывают варианты осуществления изобретения и совместно с описанием служат для объяснения принципов изобретения.

Заявленное изобретение поясняется следующими чертежами.

Фиг.1 показывает структурную схему способа обнаружения угроз в коде, исполняемом виртуальной машиной.

Фиг.2 показывает пример системы обнаружения угроз в коде, исполняемом виртуальной машиной.

Фиг.3 показывает пример реализации частного случая системы обнаружения угроз в коде, исполняемом виртуальной машиной, которым является модификация кода Java Virtual Machine (JVM).

Фиг.4 показывает пример компьютерной системы общего назначения.

Хотя изобретение может иметь различные модификации и альтернативные формы, характерные признаки, показанные в качестве примера на чертежах, будут описаны подробно. Следует понимать, однако, что цель описания заключается не в ограничении изобретения конкретным его воплощением. Наоборот, целью описания является охват всех изменений, модификаций, входящих в рамки данного изобретения, как это определено приложенной формуле.

Описание изобретения

Объекты и признаки настоящего изобретения, способы для достижения этих объектов и признаков станут очевидными посредством отсылки к примерным вариантам осуществления. Однако настоящее изобретение не ограничивается примерными вариантами осуществления, раскрытыми ниже, оно может воплощаться в различных видах. Приведенное описание предназначено для помощи специалисту в области техники для исчерпывающего понимания изобретения, которое определяется только в объеме приложенной формулы.

Для обнаружения вредоносного кода и блокирования его исполнения виртуальной

машиной необходимо осуществлять контроль над самой виртуальной машиной и кодом, который она интерпретирует. Один из способов, позволяющих это сделать, - модификация кода самой виртуальной машины. На Фиг.1 изображен алгоритм, который позволяет обнаруживать вредоносный код, исполняемый виртуальной машиной. На этапе 101 осуществляется модификация кода виртуальной машины. Данное изменение позволяет контролировать исполнение кода виртуальной машиной, отслеживать события виртуальной машины, останавливать виртуальную машину, запускать и т.д. Далее на этапе 102 отслеживаются исключения виртуальной машины. События проверяются на этапе 103, если событие, вызванное виртуальной машиной, является исключением, то на этапе 104 работа виртуальной машины будет приостановлена. Под исключениями здесь и далее понимаются критические события, которые могут вызвать нарушение правил, установленных политикой безопасности. Пока виртуальная машина остановлена, собирается информация о контексте исключения 105, а затем на этапе 106 анализируется. Под контекстом понимается информация, описывающая условия возникновения исключения (как и чем вызвано, что предшествовало возникновению критического события и т.д.). Анализ контекста исключения происходит путем сравнения информации из контекста исключения, полученной на этапе 105, с шаблоном (например, стек вызовов, предшествовавших исключению, со стеком вызовов, перечисленных в шаблоне). Анализ производится с целью выявления угрозы в коде, вызвавшем исключение. Если код содержит угрозу 107, исполнение кода виртуальной машиной будет завершено 108, если код безопасен, виртуальной машине будет разрешено продолжить работу, и тогда на этапе 109 виртуальная машина снова будет запущена.

На Фиг.2 изображен частный случай реализации предложенного алгоритма. В код виртуальной машины 203 внедряется код модуля контроля 201, который отслеживает исключения и при их наступлении собирает информацию о контексте исключения, после чего передает собранную информацию модулю анализа 202. Модуль анализа производит сравнение полученного контекста с шаблонами, которые содержатся в постоянно обновляемой базе 204. На основании сравнения модуль анализа 202 делает вывод о безопасности кода, вызвавшего исключение, и сообщает о принятом решении модулю контроля 201. Модуль контроля 201 в зависимости от решения модуля анализа 202 разрешает виртуальной машине продолжить исполнение кода либо завершает ее работу.

На Фиг.3 показана схема реализации частного случая метода модификации кода JVM 302 путем внедрения кода модуля контроля 201 в адресное пространство виртуальной машины с целью обнаружения угроз. Внедрение кода осуществляет модуль управления 303, который постоянно загружен в системе и отслеживает запуск виртуальной машины. При старте JVM в системе модуль управления 303 модифицирует код виртуальной машины. Модуль контроля 201 является расширением уже существующей встроенной системы безопасности 301, на недостатки которой указывалось выше, и является надстройкой над ней. Модификация кода JVM заключается в перегрузке ряда методов класса SecurityManager. Модуль контроля 201 отслеживает исключения, характерные для вредоносного кода (доступ к файловой системе, работа с сетью и т.д.) и, в случае возникновения такого исключения, передает контекст исключения модулю анализа 202 через модуль управления 303. Модулем анализа 202 принимается решение о безопасности поведения кода, вызвавшего исключение, на основании анализа данных, полученных от модуля контроля 201, и в том случае, если поведение признается вредоносным, модуль управления дает команду модулю контроля 201 завершить исполнение кода. Модуль анализа 202 для сравнения использует шаблоны, содержащиеся в постоянно обновляемой базе 204.

В частном случае реализации данного изобретения, вышеописанная система может использоваться для обнаружения эксплоитов, например, использующих уязвимость CVE-2011-3544. Данная уязвимость основана на ошибке в классе sun.org.mozilla.javascript.internal.NativeError. Эксплуатация данной уязвимости позволяет злоумышленнику исполнять произвольный код на удаленной машине. Для обнаружения вредоносного кода, эксплуатирующего данную уязвимость, код JVM модифицируется путем перегрузки методов checkPermission и checkExec класса Java. lang. SecurityManager. Модуль контроля 201 будет отслеживать исключение, характерное для кода, использующего данную уязвимость, например попытку отключения «песочницы» (событие setSecurityManager). При наступлении такого события исполнение кода виртуальной машиной будет остановлено, и модулем контроля 201 будет собрана информация о контексте исключения, в том числе стек вызовов текущего потока, пример которого представлен ниже:

```
java.lang.System.setSecurityManager(Unknown Source)
sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
java.lang.reflect.Method.invoke(Unknown Source)
sun.reflect.misc.Trampoline.invoke(Unknown Source)


sun.reflect.misc.MethodUtil.invoke(Unknown Source)
sun.org.mozilla.javascript.internal.MemberBox.invoke(Unknown Source)
sun.org.mozilla.javascript.internal.NativeJavaMethod.call(Unknown Source)
sun.org.mozilla.javascript.internal.Interpreter.interpretLoop(Unknown Source)
sun.org.mozilla.javascript.internal.Interpreter.interpret(Unknown Source)
sun.org.mozilla.javascript.internal.InterpretedFunction.call(Unknown Source)
sun.org.mozilla.javascript.internal.ContextFactory.doTopCall(Unknown Source)
sun.org.mozilla.javascript.internal.ScriptRuntime.doTopCall(Unknown Source)
com.sun.script.javascript.ExternalScriptable.getDefaultValue(Unknown Source)
sun.org.mozilla.javascript.internal.ScriptRuntime.toString(Unknown Source)
sun.org.mozilla.javascript.internal.NativeError.getString(Unknown Source)
sun.org.mozilla.javascript.internal.NativeError.js_toString(Unknown Source)
sun.org.mozilla.javascript.internal.NativeError.toString(Unknown Source)
javax.swing.DefaultListCellRenderer.getListCellRendererComponent(Unknown
Source)
javax.swing.plaf.basic.BasicListUI.updateLayoutState(Unknown Source)
javax.swing.plaf.basic.BasicListUI.maybeUpdateLayoutState(Unknown Source)
javax.swing.plaf.basic.BasicListUI.getPreferredSize(Unknown Source)
javax.swing.JComponent.getPreferredSize(Unknown Source)
java.awt.FlowLayout.layoutContainer(Unknown Source)
java.awt.Container.layout(Unknown Source)
java.awt.Container.doLayout(Unknown Source)
java.awt.Container.validateTree(Unknown Source)
java.awt.Container.validate(Unknown Source)
sun.plugin.util.GrayBoxPainter$2.run(Unknown Source)
java.awt.event.InvocationEvent.dispatch(Unknown Source)
java.awt.EventQueue.dispatchEvent(Unknown Source)
java.awt.EventDispatchThread.pumpOneEventForFilters(Unknown Source)
java.awt.EventDispatchThread.pumpEventsForFilter(Unknown Source)
java.awt.EventDispatchThread.pumpEventsForHierarchy(Unknown Source)
java.awt.EventDispatchThread.pumpEvents(Unknown Source)
 java.awt.EventDispatchThread.run(Unknown Source)
```

Вся собранная информация, передается в модуль анализа 202, в котором содержится правило для анализа кадра стека на обнаружение в нем угрозы, использующей уязвимость CVE-2011-3544. На вход модуля анализа подается кадр стека, который проверяется на наличие в нем вызова методов, характерных для вышеописанной угрозы.

А именно, методов Java. lang. System. setSecurityManager И sun.org.mozilla.javascript.internal.NativeError.toString. Если кадр содержит эти методы, исполняемый код признается вредоносным, и модуль управления 303 передает команду модулю контроля на остановку работы виртуальной машины.

В еще одном частном случае система анализа для проверки может не использовать обновляемую базу, а осуществлять обнаружение на основе жестко заданных шаблонов.

В другом частном случае модификация виртуальной машины может осуществляться прежде, чем виртуальная машина будет загружена в память (например, на диске).

Кроме того, в частном случае реализации настоящего изобретения модуль контроля 201 может использоваться для сбора статистической информации о выявленных угрозах, на основе этой информации будет происходить наполнение базы 204.

Фиг.4 представляет пример компьютерной системы общего назначения, персональный компьютер или сервер 20, содержащий центральный процессор 21, системную память 22 и системную шину 23, которая содержит разные системные компоненты, в том числе память, связанную с центральным процессором 21. Системная шина 23 реализована, как любая известная из уровня техники шинная структура, содержащая в свою очередь память шины или контроллер памяти шины, периферийную шину и локальную шину, которая способна взаимодействовать с любой другой шинной архитектурой. Системная память содержит постоянное запоминающее устройство (ПЗУ) 24, память с произвольным доступом (ОЗУ) 25. Основная система ввода/вывода (BIOS) 26 содержит основные процедуры, которые обеспечивают передачу информации между элементами персонального компьютера 20, например, в момент загрузки операционной системы с использованием ПЗУ 24.

Персональный компьютер 20, в свою очередь, содержит жесткий диск 27 для чтения и записи данных, привод магнитных дисков 28 для чтения и записи на сменные магнитные диски 29 и оптический привод 30 для чтения и записи на сменные оптические диски 31, такие как CD-ROM, DVD-ROM и иные оптические носители информации. Жесткий диск 27, привод магнитных дисков 28, оптический привод 30 соединены с системной шиной 23 через интерфейс жесткого диска 32, интерфейс магнитных дисков 33 и интерфейс оптического привода 34 соответственно. Приводы и соответствующие компьютерные носители информации представляют собой энергонезависимые средства хранения компьютерных инструкций, структур данных, программных модулей и прочих данных персонального компьютера 20.

Настоящее описание раскрывает реализацию системы, которая использует жесткий диск 27, сменный магнитный диск 29 и сменный оптический диск 31, но следует понимать, что возможно применение иных типов компьютерных носителей информации 56, которые способны хранить данные в доступной для чтения компьютером форме (твердотельные накопители, флеш карты памяти, цифровые диски, память с произвольным доступом (ОЗУ) и т.п.), которые подключены к системной шине 23 через контроллер 55.

Компьютер 20 имеет файловую систему 36, где хранится записанная операционная система 35, а также дополнительные программные приложения 37, другие программные модули 38 и данные программ 39. Пользователь имеет возможность вводить команды и информацию в персональный компьютер 20 посредством устройств ввода (клавиатуры 40, манипулятора «мышь» 42). Могут использоваться другие устройства ввода (не отображены): микрофон, джойстик, игровая консоль, сканнер и т.п. Подобные устройства ввода по своему обычаю подключают к компьютерной системе 20 через последовательный порт 46, который в свою очередь подсоединен к системной шине,

но могут быть подключены иным способом, например, при помощи параллельного порта, игрового порта или универсальной последовательной шины (USB). Монитор 47 или иной тип устройства отображения также подсоединен к системной шине 23 через интерфейс, такой как видеоадаптер 48. В дополнение к монитору 47, персональный компьютер может быть оснащен другими периферийными устройствами вывода (не отображены), например колонками, принтером и т.п.

Персональный компьютер 20 способен работать в сетевом окружении, при этом используется сетевое соединение с другим или несколькими удаленными компьютерами 49. Удаленный компьютер (или компьютеры) 49 являются такими же персональными компьютерами или серверами, которые имеют большинство или все упомянутые элементы, отмеченные ранее при описании сущности персонального компьютера 20, представленного на Фиг.4. В вычислительной сети могут присутствовать также и другие устройства, например маршрутизаторы, сетевые станции, пиринговые устройства или иные сетевые узлы.

Сетевые соединения могут образовывать локальную вычислительную сеть (LAN) 50 и глобальную вычислительную сеть (WAN). Такие сети применяются в корпоративных компьютерных сетях, внутренних сетях компаний и, как правило, имеют доступ к сети Интернет. В LAN- или WAN-сетях персональный компьютер 20 подключен к локальной сети 50 через сетевой адаптер или сетевой интерфейс 51. При использовании сетей персональный компьютер 20 может использовать модем 54 или иные средства обеспечения связи с глобальной вычислительной сетью, такой как Интернет. Модем 54, который является внутренним или внешним устройством, подключен к системной шине 23 посредством последовательного порта 46. Следует уточнить, что сетевые соединения являются лишь примерными и не обязаны отображать точную конфигурацию сети, т.е. в действительности существуют иные способы установления соединения техническими средствами связи одного компьютера с другим.

В заключение следует отметить, что приведенные в описании сведения являются примерами, которые не ограничивают объем настоящего изобретения, определенного формулой. Специалисту в данной области становится понятным, что могут существовать и другие варианты осуществления настоящего изобретения, согласующиеся с сущностью и объемом настоящего изобретения.

Формула изобретения

1. Способ обнаружения угроз в коде, исполняемом виртуальной машиной, в котором:

а) модифицируют код виртуальной машины для

- отслеживания исключений внутри виртуальной машины;
- управления виртуальной машиной;

б) отслеживают исключения внутри виртуальной машины;

в) останавливают виртуальную машину при наступлении исключения;

г) получают информацию о контексте исключения, содержащего данные о событиях виртуальной машины, приведших к этому исключению;

д) анализируют контекст исключения на наличие поведения, характерного для угрозы;

е) обнаруживают угрозу на основании анализа.

2. Способ по п.1, в котором к исключениям внутри виртуальной машины относятся критические события, которые могут вызвать нарушение правил, установленных политикой безопасности

3. Способ по п.1, в котором модификацию кода используют для сбора статистической

информации о выявленных угрозах.

4. Способ по п.1, в котором модификация кода виртуальной машины осуществляется путем перегрузки методов класса.

5. Система обнаружения угроз в коде, исполняемом виртуальной машиной, содержит:

а) виртуальную машину, модифицированную модулем контроля, предназначенную для исполнения анализируемого кода;

б) модуль контроля, предназначенный для

- модификации кода виртуальной машины;
- отслеживания исключений в виртуальной машине;
- сбора информации о контексте исключения;
- отправки информации о контексте исключения модулю анализа;
- остановки работы виртуальной машины по указанию модуля анализа;

в) модуль анализа, предназначенный для

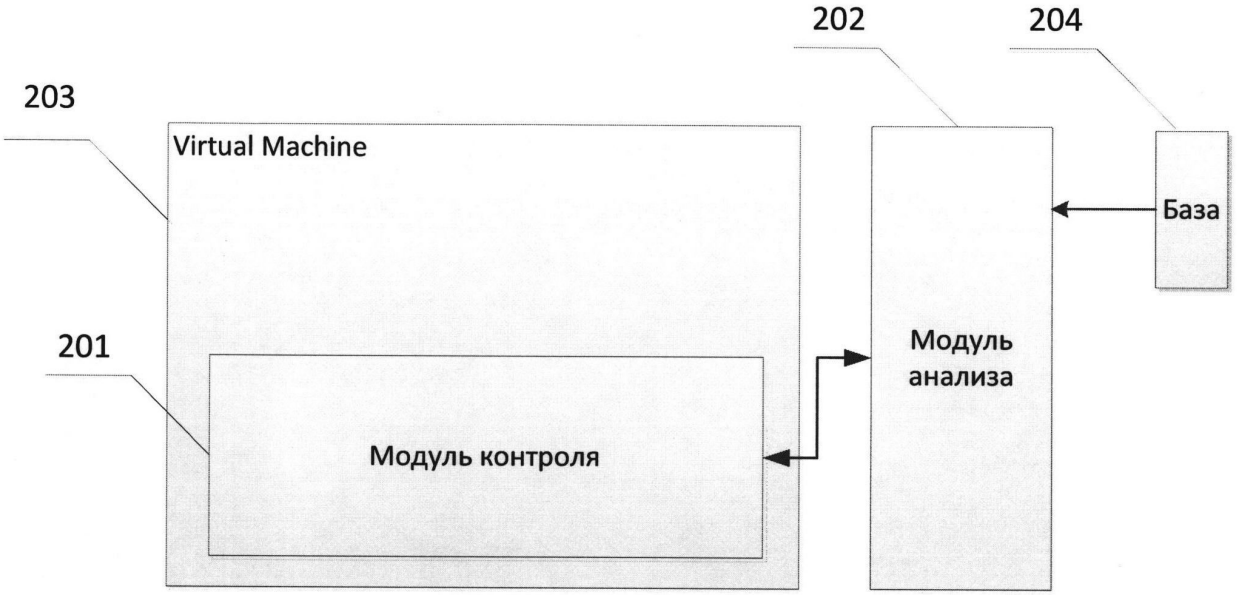
- анализа информации о контексте исключения, полученной от модуля контроля на наличие угрозы;
- обнаружения угрозы по результатам анализа.

6. Система по п.5, в которой модуль контроля дополнительно используется для сбора статистической информации о выявленных угрозах.

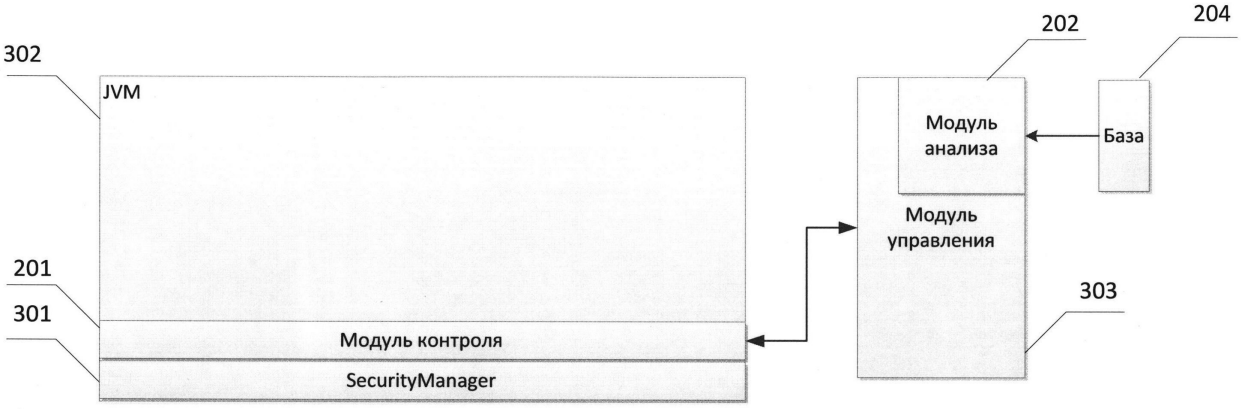
7. Система по п.6, в которой наполнение баз осуществляется на основании собранной статистической информации о выявленных угрозах.

8. Система по п.5, в которой анализ информации о контексте исключения осуществляется путем сравнения информации о контексте с внешней обновляемой базой шаблонов.

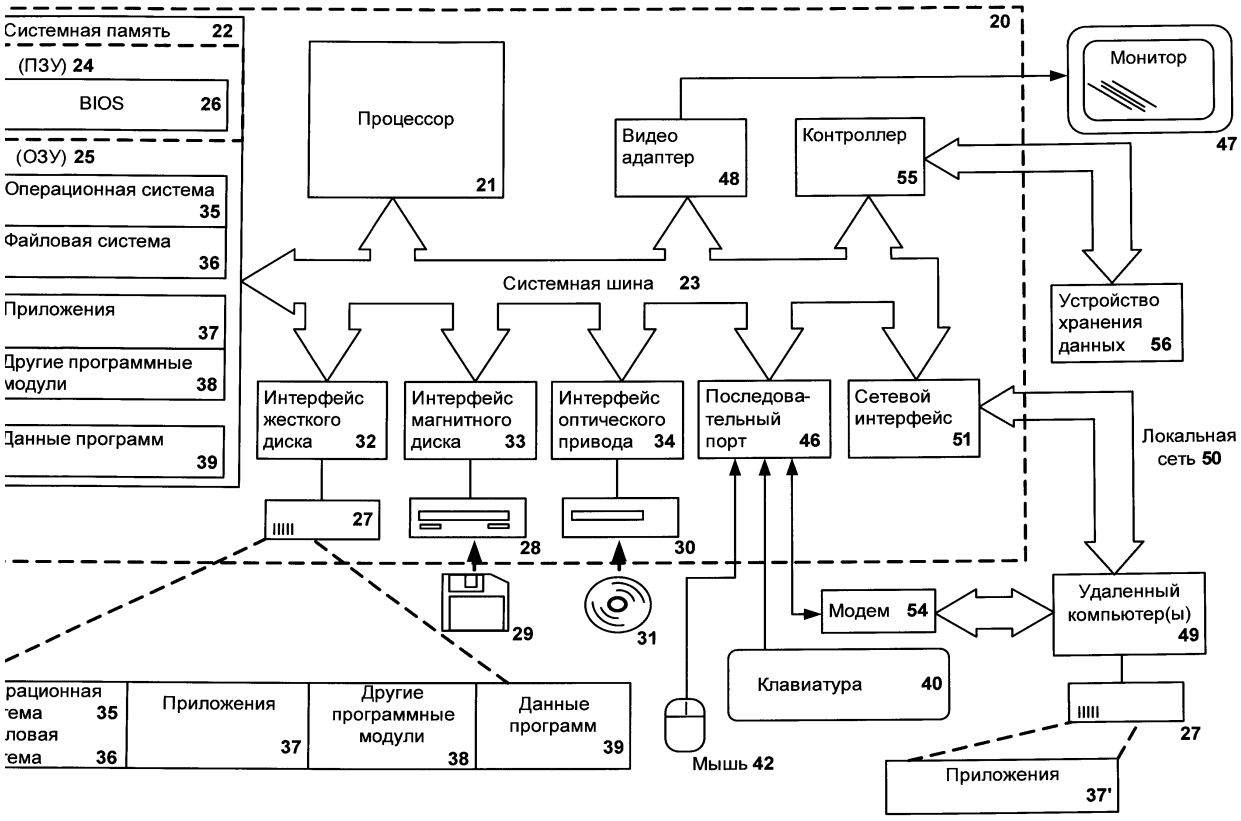
9. Система по п.5, в которой модификация виртуальной машины осуществляется путем перегрузки методов класса.


202
204
203
Virtual Machine
База
201
Модуль анализа
Модуль контроля

Фиг.2


202
204
302
JVM
Модуль анализа
База
Модуль управления
201
303
301
Модуль контроля
SecurityManager

Фиг.3


Системная память 22
(ПЗУ) 24
BIOS 26
(ОЗУ) 25
Операционная система 35
Файловая система 36
Приложения 37
Другие программные модули 38
Данные программ 39
20
Процессор 21
Видео адаптер 48
Контроллер 55
Монитор 47
Системная шина 23
Устройство хранения данных 56
Интерфейс жесткого диска 32
Интерфейс магнитного диска 33
Интерфейс оптического привода 34
Последовательный порт 46
Сетевой интерфейс 51
Локальная сеть 50
27
28
30
29
31
Модем 54
Удаленный компьютер(ы) 49
Клавиатура 40
Мышь 42
Операционная система 35
Файловая система 36
Приложения 37
Другие программные модули 38
Данные программ 39
27
Приложения 37'

Фиг.4