РОССИЙСКАЯ ФЕДЕРАЦИЯ

(19) **RU** (11) **2 390 836** (13) **C2**

(51) МПК
G06F 21/00 (2006.01)

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ,
ПАТЕНТАМ И ТОВАРНЫМ ЗНАКАМ

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(21), (22) Заявка: **2004124072/09, 06.08.2004**

(24) Дата начала отсчета срока действия патента:
06.08.2004

(30) Конвенционный приоритет:
07.08.2003 US 10/638,199

(43) Дата публикации заявки: **27.01.2006**

(45) Опубликовано: **27.05.2010** Бюл. № **15**

(56) Список документов, цитированных в отчете о поиске: **EP 1055990 A1, 29.11.2000. RU 2154855 C2, 20.08.2000. US 2002/0046275, 18.04.2002. US 6185678 B1, 06.02.2001. US 6408391 B1, 18.06.2002.**

Адрес для переписки:
129090, Москва, ул. Б.Спасская, 25, стр.3, ООО "Юридическая фирма Городисский и Партнеры", пат.пов. Ю.Д.Кузнецову, рег.№ 595

(72) Автор(ы):
УИЛМЭН Брайан Марк (US),
ИНГЛЕНД Пол (US),
РЕЙ Кеннет Д. (US),
КАПЛАН Кейт (US),
КУРИЕН Варугис (US),
МАРР Майкл Дэвид (US)

(73) Патентообладатель(и):
МАЙКРОСОФТ КОРПОРЕЙШН (US)

(54) ОТОБРАЖЕНИЕ ДОСТОВЕРНОСТИ ИЗ ВЫСОКОНАДЕЖНОЙ СРЕДЫ НА НЕЗАЩИЩЕННУЮ СРЕДУ

(57) Реферат:

Изобретение относится к области компьютерной защиты. Изобретение повышает защищенность компьютера при исполнении объектов в незащищенной среде. На одной машине, которая имеет объекты, исполняемые в незащищенной среде, и объекты, исполняемые в высоконадежной среде, достоверность объектов высоконадежной среды отображается на объекты в незащищенной среде. В системе имеется контролирующий агент, исполняемый в высоконадежной среде, который контролирует незащищенную среду и отклоняет разрешение на изменения в незащищенной среде без приема санкционирования через защищенный вход. Изобретение может быть применено, например, на платформе защищенной вычислительной базы следующего поколения (NGSCB) компании Microsoft, где традиционная операционная система вмещает в себя защищенную операционную систему (например, нексус). 3 н. и 57 з.п. ф-лы, 4 ил.

RU 2390836 C2

RU 2390836 C2

ФИГ.3

RU 2390836 C2

RUSSIAN FEDERATION

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY,
PATENTS AND TRADEMARKS

(19) **RU** (11) **2 390 836** (13) **C2**

(51) Int. Cl.
G06F 21/00 (2006.01)

(12) ABSTRACT OF INVENTION

(21), (22) Application: **2004124072/09, 06.08.2004**

(24) Effective date for property rights:
06.08.2004

(30) Priority:
07.08.2003 US 10/638,199

(43) Application published: **27.01.2006**

(45) Date of publication: **27.05.2010 Bull. 15**

Mail address:
129090, Moskva, ul. B.Spasskaja, 25, str.3, OOO "Juridicheskaja firma Gorodisskij i Partnery", pat.pov. Ju.D.Kuznetsovu, reg.№ 595

(72) Inventor(s):
UILMEhN Brajan Mark (US),
INGLEND Pol (US),
REJ Kennet D. (US),
KAPLAN Kejt (US),
KURIEN Varugis (US),
MARR Majkl Dehvid (US)

(73) Proprietor(s):
MAJKROSOFT KORPOREJShN (US)

(54) AUTHENTICITY DISPLAY FROM HIGHLY RELIABLE MEDIUM TO NON-SECURE MEDIUM

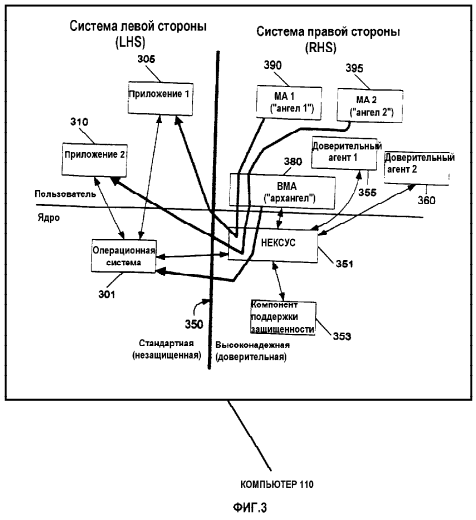
(57) Abstract:

FIELD: information technology.

SUBSTANCE: in one machine which has objects executed in an non-secure medium and objects executed in a highly reliable medium, authenticity of objects of the highly reliable medium is displayed on the objects in the non-secure medium. The system has a monitoring agent executed in the highly reliable medium which monitors the non-secure medium and prevents change in the non-secure medium without receiving authorisation through a protected input. The invention can be used, for instance on Microsoft's next-generation secure computing base (NGSCB) platform, where a traditional operating system incorporates a secure operating system (e.g. nexus).

EFFECT: increased security of a computer when executing objects in a non-secure medium.

60 cl, 4 dwg



ФИГ.3

RU 2390836 C2

RU 2390836 C2

Область техники

Изобретение относится у области компьютерной защиты. Более конкретно, изобретение относится к использованию множества сред исполнения (например, операционных систем) на одном вычислительном устройстве и обеспечивает способы, которые поддерживают достоверность таких операционных систем или сред.

Предшествующий уровень техники

Первые компьютеры были способны только выполнять единственную программу в каждый данный момент времени. Однако в настоящее время следует ожидать, что компьютеры будут обеспечивать выполнение нескольких различных частей программного обеспечения одновременно. Например, типовые многозадачные операционные системы могут исполнять несколько прикладных программ одновременно на одной машине. Ввиду данных обстоятельств, а также ввиду развития совместно используемой открытой сети (т.е. Интернет), защищенность и конфиденциальность стали двумя важными и сложными проблемами, вставшими перед компьютерной отраслью. По мере того как компьютеры все в большей степени распространяются для использования в домашних условиях, на работе и в школе, потребители и бизнес-клиенты в одинаковой степени все больше осознают важность проблем защищенности и конфиденциальности. Совершенствование возможностей программного обеспечения и аппаратных средств защищать целостность цифровой информации и конфиденциальности компьютерных пользователей становится критически важным направлением для разработчиков программного обеспечения и аппаратных средств. Компания Microsoft Corporation (Redmond, WA) выпустила платформу персонального компьютера с защищенной вычислительной базой следующего поколения (NGSCB), которая обеспечивает защищенность и конфиденциальность операционной системы.

В обычной компьютерной платформе NGSCB на компьютере 110, как показано на фиг.2, защищенная система «правой стороны» (RHS) работает во взаимосвязи с традиционной системой «левой стороны» (LHS) и центральным процессорным блоком (CPU). Система RHS разработана для защиты от злоумышленного программного обеспечения при сохранении открытости операционной системы. В случае платформы NGSCB приложения исполняются в пространстве защищенной памяти, которое весьма устойчиво по отношению к типовой фальсификации и вмешательству. В типовом случае в компьютере 110 имеется один микропроцессорный набор, который использует как система LHS, так и система RHS. Таким образом, системы LHS и RHS являются логическим, но физически обусловленным, разделением компьютера 110.

Система «левой стороны» (LHS) содержит традиционные приложения 205, 210, такие как Microsoft Word и Microsoft Excel, вместе с обычной операционной системой 201, такой как операционная система Microsoft Windows. Хотя показаны два приложения, в типовом случае может быть реализовано любое их количество.

Система «правой стороны» (RHS) содержит доверительные агенты 255, 260 вместе с элементом 251, определенным как «нексус» (связующее звено). Нексус является операционной системой высокой гарантии, которая обеспечивает определенный уровень гарантии, что касается ее поведения, и может содержать весь код режима ядра на системе RHS. Например, нексус может быть реализован для работы с защищенной информацией (например, криптографическими ключами и т.д.), которая не должна разглашаться, путем обеспечения защищенной памяти, которая гарантирована от утечки информации во внешнюю среду помимо нексуса, и путем разрешения только

некоторым сертифицированным приложениям осуществлять исполнение в рамках нексуса и получать доступ к защищенной памяти. Нексус 251 не должен взаимодействовать с главной операционной системой 201 любым способом, который позволит событиям, совершающимся в главной операционной системе 201, компрометировать поведение нексуса 251. Нексус 251 может обеспечить возможность исполнения всем приложениям или владелец машины может конфигурировать стратегию машины, согласно которой нексус 251 разрешает исполнение только некоторым агентам. Иными словами, нексус 251 будет обеспечивать исполнение любого агента, в отношении которого владелец машины выдаст ему команду на исполнение. Владелец машины может также сообщить нексусу, что не следует исполнять.

Нексус 251 изолирует доверительные агенты 255, 260, управляет передачей данных к доверительным агентам 255, 260 и от них и криптографически закрывает сохраняемые данные (например, сохраняемые на накопителе жестких дисков). Более конкретно, нексус 251 обеспечивает исполнение в режиме ядра в доверительном пространстве и предоставляет базовые сервисы доверительным агентам 255, 260, таким как установление механизмов обработки для информационного обмена с доверительными агентами и другими приложениями, и специальные доверительные сервисы, такие как аттестация платформ аппаратных средств/программного обеспечения или среду исполнения и герметизацию (закрытие, защиту) и разгерметизацию (раскрытие) секретных данных. Аттестация характеризует возможность участку кода в цифровом виде подписывать или иным образом аттестовать сегмент данных и дополнительно гарантировать получателю, что данные были созданы недоступным для подделки, криптографически идентифицируемым стеком программного обеспечения.

Доверительный агент есть программа, часть программы или сервис, который исполняется в пользовательском режиме в защищенном пространстве. Доверительный агент 255, 260 обращается к нексусу 251 за связанными с обеспечением защиты сервисами и критическими общими сервисами, такими как управление памятью. Доверительный агент может сохранять секретные данные с использованием закрытой памяти и аутентифицируется с использованием сервисов аттестации нексуса. Каждый доверительный агент или объект управляет своей собственной областью доверия и им нет необходимости зависеть друг от друга.

Система RHS также содержит компонент поддержки защищенности (SSC) 253, который использует пару ключей инфраструктуры открытых ключей (PKI) вместе с функциями шифрования для обеспечения защищенного состояния.

Платформа защищенной вычислительной базы следующего поколения (платформа NGSCB) обеспечивает такие свойства, как «аттестация», «закрытая память» и «строгая изоляция процесса». Аттестация позволяет другим компьютерам знать, что конкретный компьютер в действительности является компьютером, в качестве которого он себя представляет, и исполняет программное обеспечение, которое он представляет как исполняемое. Поскольку программное обеспечение и аппаратные средства платформы NGSCB являются криптографически удостоверяемыми по отношению к пользователю и к другим компьютерам, программам и сервисам, система может удостовериться, что другие компьютеры и процессы являются заслуживающими доверия (достоверными), прежде чем использовать их или совместно использовать информацию. Таким образом, аттестация позволяет пользователю открывать выбранные характеристики

операционной среды внешним запросчикам.

Закрытая («запечатанная») память позволяет пользователю шифровать информацию таким образом, чтобы она была доступна только для достоверного (доверительного) приложения. Это может включать в себя приложение, которое создало эту информацию, в первую очередь, или любое приложение, являющееся достоверным для приложения, являющегося собственником указанных данных. Поэтому закрытая память позволяет программе хранить секретные данные, которые не могут быть извлечены невыверенными (не заслуживающими доверия) программами, подобными вирусу или троянскому коню (программа, которая выдает себя за другую программу с целью получения информации).

Строгая изоляция процесса обеспечивает высоконадежное (доверительное) пространство путем выделения защищенной области (RHS). Операции, выполняемые в системе RHS, защищены и изолированы от системы LHS, что делает их значительно более защищенными от атак.

Платформа NGSCB также обеспечивает защищенный ввод и защищенный вывод. В случае платформы NGSCB нажатия клавиш шифруются пред тем, как они будут считаны программным обеспечением, и расшифровываются после того, как они попадают в систему RHS. Это означает, что злоумышленное программное обеспечение не может быть использовано для записи, похищения или модифицирования вводов данных посредством нажатия клавиш. Аналогичная ситуация имеет место для защищенного вывода. Информация, которая появляется на экране, может быть представлена пользователю так, чтобы никто другой не мог перехватить и считать ее. В совокупности, все эти мероприятия позволяют пользователю знать с высокой степенью достоверности, что программное обеспечение на его компьютере выполняет то, что оно должно выполнять.

Несмотря на существенные ресурсы обеспечения высокой защищенности, доступные для системы RHS, система LHS остается незащищенной. Настоящее изобретение направлено на решение этих и других недостатков современных высоконадежных компьютерных систем.

Сущность изобретения

Настоящее изобретение обеспечивает механизм для отображения достоверности объектов в доверительной (высоконадежной) среде на объекты в незащищенной среде.

Описаны системы и способы, в которых предусмотрена незащищенная среда и высоконадежная среда. Основной контролирующий агент исполняется в высоконадежной среде. Основной контролирующий агент контролирует незащищенную среду.

В соответствии с одним вариантом осуществления контролирующий агент ассоциирован с приложением, причем контролирующий агент контролирует ассоциированное с ним приложение на наличие событий или линий поведения, которые могут указывать на атаку (попытку проникновения в защищенную систему). Свойство доверительности (надежности, защищенности) контролирующего агента позволяет обнаруживать и надежным образом сообщать об этих событиях/линиях поведения, тем самым отображая достоверность высоконадежной (доверительной) среды на незащищенную среду. Основной контролирующий агент может одобрять, запрещать или модифицировать событие незащищенной среды, о котором сообщено или которое обнаружено контролирующим агентом. Например, ситуации, о которых сообщено как о требующих покрытия, такие как сообщение аппаратного средства о попытке перемещения глобальной таблицы дескрипторов (GDT) в нексус, в свою

очередь обусловят сообщение основному контролирующему агенту. Ситуацией обнаружения, например, может быть случай, когда основной контролирующий агент (для ОС) или контролирующий агент (для некоторого приложения), например, обнаруживает проблему по сканированию памяти ненадежным приложением.

В другом варианте осуществления основной контролирующий агент реагирует на ввод данных, полученный с защищенного входа. Например, основной контролирующий агент может отказать в разрешении изменений в незащищенной среде без получения санкционирования через защищенный вход. В качестве другого примера, основной контролирующий агент может отказать в разрешении изменений в незащищенной среде, если только изменения не предписываются пакетом программ, который подписан уполномоченной стороной.

Еще в одном варианте осуществления контролирующий агент использует закрытую память для хранения секретных данных для операционной системы или приложения, помещенных в незащищенной среде. Контролирующий агент может отказать в открытии секретных данных операционной системе или приложению, если только операционная система или приложение не имеют дайджеста, согласованного с владельцем секретных данных. Альтернативно, контролирующий агент может отказать в открытии секретных данных операционной системе или приложению, если только операционная система или приложение не содержатся в списке дайджестов, которые могут считывать секретные данные.

В соответствии с другими особенностями, контролирующий агент использует тест для определения того, запрашиваются ли секретные данные правомочным объектом. Один такой тест включает в себя исследование стеков объектов и гарантирование, что стеки имеют правомочное содержимое стеков. Кроме того, контролирующий агент может редактировать состояние незащищенной среды для преобразования ее в защищенную или приемлемую иным образом. Состояние может содержать начальную конфигурацию или опцию сообщения об ошибке.

Дополнительные признаки и преимущества изобретения поясняются в последующем описании приведенных для примера вариантов осуществления, которые иллюстрируются чертежами.

Краткое описание чертежей

Вышеописанная сущность изобретения, а также последующее детальное описание предпочтительных вариантов осуществления поясняются со ссылками на чертежи. В целях иллюстрации изобретения на чертежах показаны приведенные для примера структуры, соответствующие изобретению, однако, изобретение не ограничивается приведенными конкретными способами и техническими средствами.

На чертежах представлено следующее:

Фиг.1 - блок-схема иллюстрирующая приведенную для примера вычислительную среду, в которой могут быть реализованы аспекты, соответствующие изобретению,

Фиг.2 - блок-схема существующей системы NGSCB, содержащей как высоконадежную среду, так и незащищенную среду,

Фиг.3 - блок-схема приведенной для примера системы отображения, соответствующей настоящему изобретению,

Фиг.4 - блок-схема последовательности операций способа отображения в соответствии с настоящим изобретением.

Детальное описание предпочтительных вариантов осуществления изобретения

Обзор

На одной машине, имеющей объекты, исполняемые в незащищенной среде, и

объекты, исполняемые в высоконадежной среде, настоящее изобретение обеспечивает механизм отображения достоверности объектов в высоконадежной среде на объекты в незащищенной среде. Изобретение направлено на механизм, используемый в условиях, когда в первой среде исполнения (например, операционной системе) помещается вторая среда исполнения. Изобретение применимо к ситуации, характерной для использования созданной компанией Microsoft платформы защищенной вычислительной базы следующего поколения (NGSCB), где обычная операционная система (например, операционная система Windows) вмещает защищенную операционную систему (например, нексус). Описаны различные механизмы, которые позволяют второй среде защищать свою достоверность (надежность) по отношению к первой среде.

Иллюстративная вычислительная среда

Фиг.1 иллюстрирует пример подходящей среды 100 вычислительной системы, в которой могут быть реализованы аспекты настоящего изобретения. Среда 100 вычислительной системы является лишь примером подходящей вычислительной среды и не предназначается для ограничения объема использования или функциональных возможностей изобретения. Вычислительная среда 100 также не должна интерпретироваться как имеющая какую-либо зависимость или требование по отношению к любому компоненту или комбинации компонентов, показанных в составе иллюстративной вычислительной среды 100.

Настоящее изобретение может быть реализовано в других средах или конфигурациях универсальных или специализированных вычислительных систем. Примеры хорошо известных вычислительных систем, сред и/или конфигураций, которые могут подходить для использования с изобретением, включают, не ограничиваясь указанным, персональные компьютеры, серверные компьютеры, миниатюрные или портативные устройства, мультипроцессорные системы, системы на основе микропроцессоров, приставки, мобильные телефоны, программируемые приборы бытовой электроники, сетевые ПК, мини-компьютеры, универсальные компьютеры, распределенные вычислительные среды, которые включают любые из вышеуказанных систем или устройств и т.п.

Изобретение может быть описано в общем контексте команд, исполняемых компьютером, таких как программные модули, исполняемые на компьютере. В общем случае программные модули включают в себя стандартные программы, программы, объекты, компоненты, структуры данных и т.д., которые выполняют конкретные задачи или реализуют некоторые абстрактные типы данных. Изобретение может также быть реализовано в распределенных вычислительных средах, где задачи выполняются удаленными устройствами обработки, которые связаны коммуникационной сетью или иной средой передачи данных. В распределенной вычислительной среде программные модули размещаются как в локальных, так и удаленных компьютерных носителях данных, включая запоминающие устройства.

На фиг.1 представлена приведенная для примера система для реализации изобретения, включающая в себя универсальное вычислительное устройство в форме компьютера 110. Компоненты компьютера 110 могут включать в себя, не ограничиваясь указанным, блок 120 обработки, системную память 130 и системную шину 121, которая связывает различные системные компоненты, включая системную память, с блоком 120 обработки. Системная шина 121 может быть любой из различных типов шинных структур, включая шину памяти или контроллер памяти, шину периферийных устройств, локальную шину, использующую любую из

разнообразных шинных архитектур. В качестве примера, но не ограничения, такие архитектуры включают в себя шину ISA (Архитектура, соответствующая промышленному стандарту), шину MCA (Микроканальная архитектура), усовершенствованную шину ISA (EISA), локальную шину VESA (Ассоциации по стандартам в области видео электроники), шину соединения периферийных компонентов (PCI), также известную как шина Mezzanine.

Компьютер 110 в типовом случае включает в себя множество считываемых компьютером сред (носителей). Считываемые компьютером носители могут представлять собой любые известные носители, к которым компьютер 110 может осуществлять доступ, и включают в себя энергозависимые и энергонезависимые носители, съемные и несъемные носители. К примеру, но не в качестве ограничения, считываемые компьютером носители могут содержать компьютерные носители записи и коммуникационную среду. Компьютерные носители записи включают в себя энергозависимые и энергонезависимые носители, съемные и несъемные носители, реализованные любым методом или по любой технологии для хранения информации такой, как считываемые компьютером команды, структуры данных, программные модули или иные данные. Компьютерные носители записи содержат, не ограничиваясь указанным, оперативную память (RAM, ОЗУ), постоянную память (ROM, ПЗУ), электронно-стираемую программируемую постоянную память (EEPROM, ЭСППЗУ), память с групповой перезаписью (флэш-память) или другие технологии памяти, CD-ROM, универсальные цифровые диски (DVD) или иные устройства памяти на оптических дисках, магнитных кассетах, магнитных лентах, устройства памяти на магнитных дисках или иные магнитные устройства памяти, или любые иные носители, которые могут быть использованы для хранения желательной информации и к которым может быть обеспечен доступ компьютера 110. Коммуникационная среда (среда передачи) в типовом случае воплощает считываемые компьютером команды, структуры данных, программные модули или иные данные в модулированном сигнале данных, таком как несущее колебание или иной транспортный механизм (механизм передачи), и включает в себя любую среду доставки информации. Термин «модулированный сигнал данных» означает сигнал, у которого одна или более характеристик установлены или изменяются таким образом, чтобы кодировать информацию в сигнале. В качестве примера, но не ограничения, коммуникационная среда включает в себя проводные среды, такие как проводная сеть или прямое проводное соединение, и беспроводную среду передачи, такую как акустическая, радиочастотная, инфракрасная и другая беспроводная среда передачи. Комбинации любых вышеуказанных сред также должны быть включены в объем носителей (сред), считываемых компьютером.

Системная память 130 включает в себя компьютерный носитель записи в форме энергозависимой и/или энергонезависимой памяти такой, как постоянная память (ПЗУ, ROM) 131 и оперативная память (ОЗУ, RAM) 132. Базовая система ввода/вывода (BIOS) 133, содержащая базовые подпрограммы, которые способствуют переносу информации между элементами в компьютере 110, например, при запуске, в типовом случае сохранена в ПЗУ 131. ОЗУ 132 в типовом случае содержит данные и/или программные модули, которые непосредственно доступны и/или обрабатываются блоком 120 обработки. В качестве примера, но не ограничения, на фиг.1 показаны операционная система 134, прикладные программы 135, другие программные модули 136 и программные данные 137.

Компьютер 110 может также включать в себя другие съемные/несъемные,

энергозависимые/энергонезависимые компьютерные носители записи. Например, на фиг.1 показан дисковод 141 жестких дисков для считывания с несъемного, энергонезависимого магнитного носителя и записи на него дисковод 151 магнитных дисков для считывания со съемного энергонезависимого магнитного диска 152 и записи на него, и дисковод 155 оптических дисков для считывания со съемного энергонезависимого оптического диска 156 или записи на оптический диск, такой как, например, ПЗУ на компакт-диске (CD-ROM) или иные оптические носители записи. Другие съемные и несъемные, энергозависимые и энергонезависимые компьютерные носители записи, которые могут быть использованы в приведенной для примера операционной среде, включают в себя, не ограничиваясь указанным, кассеты на магнитных лентах, карты флэш-памяти, DVD, цифровые видеомагнитные ленты, твердотельные ОЗУ, твердотельные ПЗУ и т.п. Дисковод 141 жестких дисков в типовом случае соединен с системной шиной 121 посредством интерфейса несъемной памяти, такого как интерфейс 140, и дисковод 151 магнитных дисков и дисковод 155 оптических дисков соединены с системной шиной 121 в типовом случае посредством интерфейса схемной памяти, такого как интерфейс 150. Подразумевается, что настоящее изобретение также может быть реализовано на встроенном микропроцессоре, в котором центральный процессорный блок и вся память находятся на одном кристалле в одной компоновке.

Дисководы и связанные с ними считываемые компьютером носители, описанные выше и показанные на фиг.1, обеспечивают хранение считываемых компьютером команд, структур данных, программных модулей и других данных для компьютера 110. На фиг.1, например, показано, что дисковод 141 жесткого диска хранит операционную систему 144, прикладные программы (приложения) 145, другие программные модули 146 и программные данные 147. Заметим, что эти компоненты могут быть теми же самыми или отличающимися от операционной системы 134, прикладных программ 135, других программных модулей 136 и программных данных 137. Операционная система 144, прикладные программы 145, другие программные модули 146 и программные данные 147 обозначены отличающимися ссылочными позициями для иллюстрации того, что они, как минимум, являются другими копиями. Пользователь может вводить команды и информацию в компьютер 110 посредством устройств ввода, например, клавиатуры 162 и координатно-указательного устройства 161, такого как мышь, трекбол или сенсорная панель. Другие устройства ввода (не показаны) могут включать в себя микрофон, джойстик, игровую панель, спутниковую параболическую антенну, сканер и т.п. Эти и другие устройства ввода часто соединяются с блоком 120 обработки через интерфейс 160 пользовательского ввода, связанный с системной шиной, но могут быть соединены и посредством других интерфейсов и структур шин, таких как параллельный порт, игровой порт или универсальная последовательная шина (USB) и т.д. Монитор 191 или иное устройство отображения также соединено с системной шиной 121 через интерфейс, например, такой как видеоинтерфейс 190. Помимо монитора, компьютеры также могут включать в себя другие периферийные устройства вывода, например, громкоговорители 197 и принтер 196, которые могут быть соединены через интерфейс 195 устройств вывода.

Компьютер 110 может работать в сетевой среде с использованием логических соединений с одним или более удаленными компьютерами, такими как удаленный компьютер 180. Удаленный компьютер 180 может представлять собой ПК, портативное устройство, сервер, маршрутизатор, сетевой ПК, одноранговое

устройство или другой обычный сетевой узел и в типовом случае включает в себя многие или все из элементов, описанных выше применительно к компьютеру 110, хотя на фиг.1 показано только устройство 181 памяти. Логические соединения, показанные на фиг.1, включают в себя локальную сеть (LAN) 171 и глобальную сеть (сеть широкого охвата - WAN) 173, но могут включать в себя и другие сети. Такие сетевые среды являются общеизвестными в офисах, компьютерных сетях предприятий, интранетах и в Интернет.

При использовании в сетевой среде локальной сети (LAN) компьютер 110 соединяется с локальной сетью 171 через сетевой интерфейс или адаптер 170. При использовании в сетевой среде глобальной сети (WAN) компьютер 110 в типовом случае включает в себя модем 172 или иное средство для установления связи в глобальной сети 173, такой как Интернет. Модем 172, который может быть внутренним или внешним, соединен с системной шиной 121 через интерфейс 160 пользовательского ввода или иной подходящий механизм. В сетевой среде программные модули, изображенные по отношению к компьютеру 110, или их части могут быть сохранены в удаленном устройстве памяти. В качестве примера, но не ограничения, фиг.1 иллюстрирует удаленные прикладные программы 185 как хранящиеся в устройстве 181 памяти. Следует иметь в виду, что показанные сетевые соединения приведены для примера и что могут быть использованы и другие средства установления канала связи между компьютерами.

Иллюстративные варианты осуществления

Как описано выше, в технике известно, что компьютер может быть конфигурирован для обеспечения двух отдельных сред: доверительной (высоконадежной) и незащищенной. Обычный код, достоверность которого не подтверждена (например, код, линия поведения которого не проверена или не может быть исключена из возможного применения со злонамеренной целью), исполняется в незащищенной среде. Обычное прикладное программное обеспечение, такое как игры, текстовые процессоры, электронные таблицы и т.д., а также обычные операционные системы, драйверы устройств и отладочные программы, обычно относят к незащищенной категории. Код, достоверность которого подтверждена каким-либо образом, может исполняться в доверительной среде. Некоторая часть компьютерной памяти (т.е. «изолированная» или «отгороженная барьером») предназначена для доступа только в доверительной среде.

В последующем описании агент интерпретируется как «доверительный» (достоверный, высоконадежный), если он был реализован в соответствии с защищенной процедурой, предназначенной для сохранения его целостности или выявления нарушения его целостности. Например, агент может быть реализован посредством доверительной (высоконадежной) процедуры, которая проверяет идентификационные данные агента и сред, в которой он исполняется (аттестация), ему может быть назначено местоположение в защищенной памяти (закрытой памяти), которая недоступна никакому другому агенту, как достоверному, так и недостоверному, и он может иметь возможность скрытия секретных данных. Такой доверительный агент может быть уникально и надежно идентифицирован.

В доверительной среде имеются ограничения относительно того, что может делать код. Например, имеется меньше доверительных программных интерфейсов приложения (API) (в сравнении с очень богатым набором API в типовой системе LHS), агенты, исполняемые в доверительной среде, могут осуществлять информационный обмен друг с другом через ограниченные формальные механизмы межпроцессорной

связи (IPC), и агенты могут иметь доступ к более ограниченному и примитивному набору API и сервисов для представления текста и изображений пользователю. Эти ограничения снижают сложность и, следовательно, «поверхность» (область) атак доверительной среды и доверительных агентов, которые работают в ней.

Незащищенная среда, с другой стороны, подобна среде, в типовом случае создаваемой операционной системой на «открытой» вычислительной системе (например, на персональном компьютере, портативном компьютере и т.д.) - т.е. почти любой код может исполняться в такой незащищенной среде, и код, исполняемый в стандартной среде, имеет полный доступ к большому, богатому набору программных сервисов и интерфейсов. Незащищенная среда и доверительная среда могут быть разделены на подсреды. Например, незащищенная среда может быть разделена на незащищенный пользовательский режим (где исполняются обычные приложения) и незащищенный режим ядра (где исполняется обычная операционная система). Аналогичным образом, доверительная среда может быть разделена на доверительный пользовательский режим (где исполняются специальные достоверные приложения) и доверительный режим ядра (где исполняется доверительная операционная система, которая создает доверительную среду для достоверных приложений).

Когда доверительная (высоконадежная) и незащищенная среды сосуществуют на одной и той же компьютерной системе, высоконадежная среда может предпринимать шаги для гарантирования того, чтобы на ее доверительность не могло повлиять то, что происходит в незащищенной среде, или любой код пользовательского режима в высоконадежной среде. Варианты осуществления настоящего изобретения обеспечивают механизм для отображения или иного использования доверительности высоконадежной стороны в пользу незащищенной стороны.

На фиг.3 представлена блок-схема одного варианта осуществления системы отображения в соответствии с настоящим изобретением, а на фиг.4 представлена блок-схема последовательности операций одного варианта способа отображения в соответствии с настоящим изобретением. Система LHS исполняемой на компьютере 110 системы подобна описанной выше со ссылкой на фиг.2. Два приложения 305, 310 исполняются во взаимосвязи с операционной системой 301. Части системы RHS также сходны с тем, что описано со ссылкой на фиг.2. Два доверительных агента 355, 360 исполняются с нексусом 351 и SSC 353. Понятно, что любое число приложений может исполняться на системе LHS, и любое число доверительных агентов может исполняться на системе RHS.

На фиг.3 показана система, в которой операционная система 301 и нексус 351 исполняются на одном компьютере 110. Логическое разделение 350 между операционной системой 301 и нексусом 351 обеспечивает осуществление некоторого информационного обмена при защите нексуса 351 от событий, инициированных в операционной системе 301.

В варианте осуществления, показанном на фиг.3, операционная система (ОС) 301 является главной операционной системой («хозяином»), а нексус 351 является «гостем» в ОС 301. То есть ОС 301 обеспечивает некоторые сервисы и ресурсы для нексуса 351, такие как память и время процессора. В одном варианте осуществления логическое разделение 350 позволяет нексусу 351 основываться на определенных ресурсах операционной системы 301, в то же время позволяя нексусу 351 защищаться от действий (как злонамеренных, так и добросовестных), которые возникают в операционной системе 301 и могут обусловить поведение нексуса противоположно поведенческим спецификациям. Например, нексус 351 и доверительные ресурсы,

связанные с ним, например, SSC 353, могут управлять логическим разделением. Понятно, что изобретение не ограничивается конкретной формой нексуса 351. Подразумеваются механизмы, которые обеспечивают возможность построения разделения 350 так, чтобы обеспечить баланс взаимодействия и защиты.

Следует отметить, что фиг.3 показывает операционную систему 301 в качестве «хозяина», а нексус - в качестве «гостя». В принципе такая характеристика относится к тому, что в этих примерах операционная система 301 обеспечивает определенную инфраструктуру операционной системы, которая используется операционной системой 301 и нексусом 351 (например, драйверы устройств, планирование и т.д.). Нексус 351 является «гостем» в том смысле, что он может основываться на определенных ресурсах инфраструктуры операционной системы 301, а не обеспечивать их самостоятельно. Однако следует отметить, что параметры того, что делает операционную систему «хозяином» или «гостем», являются гибкими. Следует иметь в виду, что описываемые методы могут быть применены к взаимодействию любых двух или более операционных систем, исполняемых на той же самой машине (или даже на одном и том же наборе соединенных машин). Две или более операционных систем, которые исполняются на одной машине, являются примерами «среды», которая может потребовать взаимодействия друг с другом на одной машине, хотя понятно, что изобретение не ограничивается традиционными операционными системами.

Отображение (проецирование) представляет собой механизм, посредством которого часть производительности и свойств доверительных агентов (на системе RHS) может быть распространена на код системы LHS. Согласно одному примеру отображение позволяет возможности платформы NGSCB применить к существующему коду. Например, вместо переноса приложения, такого как Microsoft Excel, на систему RHS, отображение в соответствии с настоящим изобретением позволяет создать контролирующего агента (также называемого здесь «ангелом», т.е. средством поддержки) для приложения (также называемого здесь «смертным» («простым смертным»), т.е. подверженным отказам), что, в свою очередь, позволяет существующему приложению исполняться с множеством из тех же самых полезных свойств, что и у доверительного агента. Отображение может быть применено как к операционной системе системы LHS (например, Microsoft Windows), так и к любой прикладной программе системы LHS (например, Microsoft Office), для которой желателен некоторый уровень доверительных операций. Отображение также может быть применено к драйверам устройств системы LHS. Таким образом, как описано ниже, отображение позволяет доверительному агенту защищать, гарантировать, аттестовать и расширять операционные системы, сервисы и программы системы LHS.

На фиг.3 показан контролирующий агент 390 (MA1), который соответствует приложению 305, и контролирующий агент 395 (MA2), который соответствует приложению 310 (этап 400 на фиг.4). Каждый контролирующий агент или «ангел» (поддерживающее средство) защищает ассоциированное с ним приложение.

В одном варианте осуществления разработчик представляющего интерес объекта LHS (например, приложения), таким образом, создает «ангела», который защищает объект системы LHS. Это позволяет разработчику обеспечивать «ангела» глубоким знанием приложения, которое он контролирует. Такой «ангел» может быть более чувствительным к аномалиям в приложении, которое он контролирует и, таким образом, защищает и проверяет его более эффективно. Например, основной контролирующий агент, созданный разработчиком операционной системы, может

воплощать детальное знание об управлении памятью операционной системы, что позволяет ему быстро идентифицировать вызывающие подозрения операции в памяти.

В другом варианте осуществления «ангел» может предпринимать корректирующие или превентивные меры, если он обнаруживает аномалии или вызывающие подозрения действия в ассоциированном с ним приложении. Например, «ангел» может обнаружить попытку ассоциированного приложения изменить ключевую переменную в памяти, полагаемую инвариантной создателем приложения, перехватить запись в переменную. Такая операция записи, вероятно, будет указывать по меньшей мере на искажение кода приложения, если только не полное разрушение, например, недоброжелательным, например, вирусным кодом. Короче говоря, «ангел» действует как контролирующий агент для наблюдения за негативными или подозрительными действиями в ассоциированном с ним приложении для принятия соответствующего корректирующего или превентивного действия. Его действия могут быть ограничены, чтобы предотвратить нанесение ущерба ассоциированному с ним приложению. «Ангел» может быть присоединен к конкретному объекту, например, программе или приложению, или к группе таких объектов, программ и/или приложений.

Основной контролирующий агент (также называемый здесь «архангелом») (BMA) 380 ассоциирован с базовой операционной системой (т.е. ОС 301 системы LHS) (блок 410). В одном варианте осуществления основной контролирующий агент 380 создается создателем ОС 301 системы LHS. Это позволяет основному контролирующему агенту 380 воплощать детальное знание об ОС системы LHS, что делает его более чувствительным к аномальному поведению ассоциированной операционной системы.

Например, «архангел» может знать формат базы данных виртуальных адресов, базы данных процессов и базы данных PFN (номеров страничных блоков) и на этой основе обнаруживать случаи, когда мошеннические драйверы устройств реализовали противоправные отображения на процессы путем отображения PFN, которые они не должны иметь. Таким образом, «архангел» может обнаруживать отображения, выполненные не администратором памяти (например, мошенническим драйвером устройства), и может обнаружить перекрестные отображения процессов, которых не должно быть.

В таком случае «архангел» может установить соглашение с ОС, которая может подвергнуться изменениям. ОС и «архангел» могут прийти к соглашению, что база данных PFN должна быть всегда последовательной (непротиворечивой), какая бы блокировка ни поддерживалась, и что эта последовательность должна быть представлена контрольной суммой. Так, с периодическими интервалами, «архангел» должен проверять блокировку и при обнаружении ее раскрытия (это переменная памяти и, следовательно, может легко проверяться) переходить к контрольной сумме базы данных PFN. Если «архангел» обнаруживает, что контрольная сумма не согласуется, то он делает вывод, что база данных PFN фальсифицирована.

Кроме того, «архангел» может знать переменные управления для отладчика ядра и воздействовать на переменные управления для блокирования использования отладчика ядра.

Дополнительный пример включает загрузку процесса: контролировать загрузчик, администратор кэша, обработчик ошибки из-за отсутствия страницы и т.д., чтобы гарантировать корректную загрузку корректных битов в процесс пользовательского режима (или любого другого модуля, загружаемого в систему) или с надлежащей подписью, возможно с перечислением в списке кэшей, поддерживаемом в таблице,

известной «архангелу». «Архангел» может иметь возможность упреждения необходимости для загрузчика, обработчика ошибки из-за отсутствия страницы и т.п. в отображении кода/данных в процесс (страничной подкачки (листания) и т.д.). Система RHS может поддерживать физические страницы системы LHS для данного процесса блокированными («запертыми») (даже для ОС системы LHS) до тех пор, пока ОС не выполнит хорошо известные функции. Система RHS управляет таблицами страниц для процессов системы LHS. Таким образом, имеется ряд механизмов, которые разработчик «архангела» должен включить в «архангела» для ограничения ненадлежащего поведения.

Другой пример включает в себя повышение устойчивости («упрочнение», «закалку») процесса. Имеются известные испытанные механизмы для модифицирования одним процессом другого процесса. «Архангел» может обеспечить то, что все карты совместно используемой памяти, а также копирование данных в пространство другого процесса и из него будут ограничены. Другой пример включает ядро только для считывания, в котором все «текстовые» страницы (страницы кода) ядра и драйверы устройств закрыты для доступа.

«Архангел» 380 также поддерживает отображение на «ангелов» на «по-процессной» основе (ограниченный доступ). Это означает, что «ангелы», которые подобны агентам в том, что система будет исполнять любого «ангела», о котором запрашивает пользователь (согласованно с пользовательской стратегией), и которые не являются частью вектора аттестации, как определено ниже (т.е. «архангел» в действительности является частью конфигурации машины), могут причинить ущерб, например, посягая на конфиденциальность системы левой стороны, оказывая воздействия на приложения («простых смертных»), для применения к которым они не предназначены. Поэтому желательно, чтобы «ангелы» были жестко связаны с конкретными приложениями («простыми смертными»). Предпочтительно это обеспечивается путем обеспечения возможности «ангелу» оказывать влияние только на приложение («смертного»), которое запускает «ангела», или обеспечения возможности «ангелу» применяться только к «смертному», который согласуется с объявленным дайджестом в декларации «ангела», причем проверка дайджеста осуществляется «архангелом», и только после того, как «смертное» приложение вызывает «ангела» дайджестом для его запуска. Это желательно потому, что обеспечивает надежность и практичность, позволяя каждому поставщику приложения писать программу-«ангел» для своего приложения и позволяя любому пользователю использовать его без риска причинения ущерба или нарушения конфиденциальности чего-либо еще.

Таким образом, «архангел» является как агентом, который следит за системой LHS, так и агентом, который предлагает сервисы другим «ангелам». Поскольку «архангел» обладает наиболее детальным знанием структуры процессов системы LHS, он, по всей вероятности, будет тем «архангелом», который принимает решение о том, какой «ангел» может быть связан с каким из процессов системы LHS. Ограничение означает, что «ангел» (который не является частью вектора аттестации нексуса) может затронуть только процессы, которые он запускает или которые вызывают его для их защиты. Это предохраняет «ангела» от действия случайным образом на процессы системы LHS. Такое подразделение («архангел» обладает возможностями на уровне ОС и подтверждается подобно нексусу, «ангелы» получают ограниченные возможности на уровне приложения и могут свободно исполняться подобно любому другому агенту) является желательным.

В одном варианте осуществления «ангелы» могут включать «архангела» (и по

расширению основную ОС системы LHS) в свои вектора аттестации. Вектор аттестации представляет собой список дайджестов, релевантных для обеспечения защиты компонентов, которые устанавливают релевантную для обеспечения защиты конфигурацию объекта. Например, дайджест для агента может включать в себя данные о самой машине или материнской плате, нексусе и самом агенте, вместе с другой информацией. Этот стек чисел является строгим, надежным указателем того, что представляет собой агент и что представляет собой среда, в которой исполняется агент. Это позволяет другому объекту удостовериться, имеет ли он дело с «реальным агентом» или нет. Вектора аттестации накапливаются (компонуются последовательно), так что дайджест агента не является частью вектора для нексуса, но дайджест нексуса является частью дайджеста для агента. Поэтому если вектор аттестации некоторого объекта включается в другой объект, то это означает, что они все связаны друг с другом вместе в распознаваемую защищенную конфигурацию. Свойство аттестации состоит в том, что она очень строго идентифицирует релевантную для обеспечения защиты конфигурацию системы.

Иными словами, вектор аттестации представляет собой список значений дайджестов, который определяет идентификационные данные программного обеспечения системы RHS. Предпочтительно программное обеспечение, загруженное в систему RHS, классифицируется (каталогизируется) перед загрузкой, и сам процесс хорошо изолируется, так чтобы он не мог быть изменен. Это представляет собой индуктивный процесс: аппаратные средства «подписывают» дайджест нексуса (обеспечивают удостоверение дайджеста нексуса), а нексус в свою очередь аттестует (удостоверяет) агента. Таким путем внешняя сторона может установить подлинность этих дайджестов сравнением с известным списком, чтобы определить, санкционирует ли эта внешняя сторона программное обеспечение, исполняемое в системе. «Ангел» и «архангел», поскольку он исполняются в системе RHS, имеют хорошо определенные идентификационные данные кода. По этой причине эти идентификационные данные кода могут быть перечислены в векторе аттестации, который описывает среду, в которой исполняется код системы LHS. Поскольку «ангел» не может полностью контролировать исполнение кода системы LHS, этот оператор идентификационных данных кода не является строгим в той же степени, как оператор идентификационных данных кода агента системы RHS, но означает, что данная часть кода системы LHS исполняется при ограничениях «ангела», «архангела» и нексуса, которые имеют строгие идентификационные данные кода.

Варианты осуществления «архангела» могут открывать некоторый набор программных интерфейсов приложений (API) «ангелам» для обеспечения поддержки для некоторых функций и/или свойств «ангелов». Например, для операции памяти «архангел» будет желательно иметь посредником. «Ангелу» может потребоваться проверить код закрытого приложения по виртуальному адресу VA=100. Однако он может не знать, на какой физический адрес он отображается. Нексус не имеет сведений о таких структурах. Поэтому «архангел» (который знает, как работает ОС системы LHS) использует базовые сервисы нексуса (которые может вызывать только «архангел») для осуществления считывания из релевантной памяти ядра системы LHS. «Архангел» использует данные из памяти ОС системы LHS для вычисления корректных отображений для памяти приложений системы LHS. «Ангелу» затем сообщается, какой адрес закрытого приложения соответствует адресу «ангела», и «ангел» затем может проверять это содержимое и продолжать обработку. Иными словами, для «ангелов», связанных с процессами (т.е. ангелов, которые применяются

только к авторизованным процессам, но не перемещаются произвольным образом между состояниями системы LHS), желательно, чтобы «архангел» интерпретировал структуры данных системы LHS.

Дополнительная приведенная для примера функция включает обеспечение защищенного канала IPC (межпроцессорной связи), который разрешает доступ к данным только приложению системы LHS и «ангелу» системы RHS. Ядро системы LHS обычно может просматривать все страницы, которые проходят по каналу IPC между системой LHS и системой RHS, но только если доступ к этим страницам может быть обеспечен под бдительным надзором «архангела», то будет обеспечена надежная гарантия того, что данный процесс (процесс, контролируемый конкретным «ангелом») может получить доступ к данным в канале. Другая приведенная для примера функция обеспечивает «ангелу» способность контролировать, какие модули (например, DLL (динамически подсоединяемые библиотеки)) и какие версии этих модулей могут быть загружены в пространство процесса для данного процесса.

В качестве доверительного (высоконадежного) объекта «архангел» 380 имеет доступ к памяти, ассоциированной с системой LHS, и уведомляется обо всем, что происходит в системе LHS. «Архангел» 380 является предварительно программируемым с совокупностью знаний, которую он использует для обнаружения несоответствий, чтобы определить, следует ли предпринять какие-либо действия в интересах обеспечения защиты. Например, «архангел» 380 может перехватывать определенные наборы событий системы LHS. Это могут быть события, которые разрешены системой LHS и которым не препятствует нексус 351 или доверительная среда, которой он управляет. Например, «архангел» 380 может обнаружить некорректные отображения на систему LHS (что было бы разрешено нексусом 351 в противном случае), которые указывают на возможные атаки или проблемы, связанные с защитой. «Архангел» 380 может также выполнять проверку на непротиворечивость.

Для варианта осуществления, показанного на фиг.3, каждый «ангел» связан или иным образом контролируется «архангелом» 380 и нексусом 351 (блок 420). «Архангел» 380 обуславливает связь между ангелом и ассоциированным с ним кодом системы LHS, что ограничивает возможности «ангела» влиять на конфиденциальность и защищенность, например, на систему LHS.

Желательно, чтобы поведение «ангелов» было ограничено влиянием только на процессы, с которыми они подразумеваются связанными, поскольку нексус 351 и «архангел» 380 будут исполнять любого «ангела», которого пользователь предписывает им для исполнения в соответствии с пользовательской стратегией. «Архангел» имеет возможности наравне с нексусом и будет детально исследован до того же самого уровня. Что касается «ангелов» и остальных агентов, нексус будет исполнять то, что будет предписано им пользователем. В то время как нексус и «архангелы» ограничены, обычные «ангелы» (подобно агентам) не ограничены (хотя пользователь может установить стратегии, которые предписывают нексусу исполнять или не исполнять агентов или «ангелов», подписанных, например, конкретным средством оценки).

Желательно, чтобы «ангелы» были ограничены. Например, «ангелу» с сигнатурным блоком, который указывает: «ангел для первой программы», не должно быть разрешено использовать память основой ОС системы LHS или использовать память других программ. Разрешение этого нарушило бы права многих пользователей и сделало бы использование «ангелов» опасным, а не полезным.

Поэтому «архангел» удостоверяется в том, что «ангелы» получают доступ только к программам системы LHS, к которым разрешается их доступ.

Доверительный агент предпочтительно имеет не больше возможностей, чем любая программа системы LHS. В частности, доверительный агент не может ни получить доступ к ОС системы LHS, ни управлять или изменять состояние конфигурации ОС системы LHS. Вместо этого, «ангелам» предпочтительно разрешается только проверять или модифицировать память «простых смертных» (приложений), к которым они применимы. Кроме того, в некоторых вариантах осуществления «архангелы» могут запретить «ангелам» изменять код «простого смертного», ограничивая «ангела» считыванием чего-либо в адресном пространстве пользовательского режима своего «смертного» и записью в его пространство памяти для чтения, не предназначенной для совместного использования. Однако некоторые механизмы требуют, чтобы вызов ангела «смертным» позволял возвратиться не в точку вызова, а в вычисленную точку возврата. Это позволяет «ангелу» обеспечивать запуск некоторых событий в известном корректном адресе «смертного», что является надежным методом противодействия атакам типа «трамплина» (создание программных секций, передающих управление от одной секции к другой) на основе искаженных стеков, изменяющих адреса возврата.

«Ангел» может только контролировать ассоциированный с ним объект или группу объектов (блок 430) и является не более доверительным, чем любой другой агент. «Ангел» не может контролировать или иными образом наблюдать неассоциированные объекты. В частности, «ангел» имеет одно или более из следующих свойств:

a. Ангел может контролировать память пользовательского режима только процесса или процессов, к которым он присоединен (т.е. «простого смертного») (в отличие от возможности, обычно присущей коду системы RHS - см. выше).

b. Только «архангел» может просматривать память режима ядра ОС системы LHS, к которой он присоединен.

c. «Ангел» может быть применен только к тем процессам системы LHS, которые вызывают и запрашивают его, или применим к процессам системы LHS, которые он запускает.

d. «Ангел» может быть ограничен декларативным форсированием. Например, нексус и/или «архангел» могут ограничить «ангела» осуществлять отображение только на те процессы, которые содержат исполняемые файлы, которые совпадают с исполняемыми файлами, заявленными в декларации для «ангела». Так, например, «ангел» для «хакерного инструментария» не может отображать на приложение системы LHS случайным или злонамеренным образом без изменения с чьей-либо стороны декларации для «ангела».

«Архангел» 380 может обусловить вышеприведенные ограничения (блоки 440 и 450). Для этой цели «архангелу» может быть обеспечен экстенсивный доступ к системе LHS, и в этом случае он является предметом детального исследования на уровне, соответствующем уровню для нексуса (т.е. интенсивного исследования). Например, «архангел» имеет возможности, превышающие ОС системы LHS, и, следовательно, превышающие возможности чего-либо, исполняемого в системе LHS. Иными словами, «архангел» может считывать любую память системы LHS, но не имеет особых возможностей в отношении системы RHS, таких как доступ к памяти ядра системы RHS или возможность наблюдения процессов других агентов, или ограничения, усиления, модификации и т.д. нексуса или других агентов системы RHS.

«Ангел» может только читать адресное пространство программы, к которой он применяется (т.е. «ангелы» имеют специальные возможности, которые применимы только к «простым смертным», к которым они применяются). «Архангел» может также читать всю память системы LHS (этап 440), предоставляя специфические для процесса сервисы таким образом, чтобы «ангелы» имели доступ только к адресному пространству программ, которые они контролируют и защищают.

«Ангел» может «проецировать» своего «подзащитного» по меньшей мере следующими путями (этапы 430 и 450):

a. Он может запереть или маркировать как «только для чтения» различные элементы памяти, возможно в координации с поведением «подзащитного», чтобы предотвратить возможные изменения (например, вирусные атаки) на «подзащитного».

b. Он может выполнять некоторые ключевые операции для «подзащитного» в своем доверительном пространстве.

c. Он может настаивать на конкретных мерах защиты «подзащитного», таких как ограничение того, какие изменения конфигурации могут быть осуществлены, или допускать такие изменения с санкции авторизованного лица, использующего механизм защищенного ввода.

d. Он может сканировать память и состояние «подзащитного» с желательными интервалами, отслеживая ошибки из-за несовместимости, искажения и т.д., и предупреждать пользователя или останавливать выполнение «подзащитного», прежде чем произойдет дальнейший ущерб или непреднамеренное/ неавторизованное действие.

e. Он может открыть закрытые/зашифрованные данные для «подзащитного» только в той мере, как это необходимо, для минимизации объема таких данных, которые могут подвергаться атакам в конкретный момент времени.

1. Он может использовать закрытую память для сохранения закрытыми секретных данных для системы LHS (или приложения системы LHS) и отказывать в предоставлении этих секретных данных любой системе LHS (или приложению системы LHS), которая не имеет дайджеста, согласованного с владельцем секретных данных или включенного в список как разрешенный владельцем секретных данных.

f. При наличии надлежащего API, он может изменить состояние исполнения «подзащитного»; т.е. он может направить потоки (цепочки выполняемых задач) в известные точки исполнения, переадресовывать поток управления в целевом приложении или выполнять вычисления и исполнение ветвления для целевого приложения. Он может также изменять состояние конфигурации, состояние запуска и т.п. для принудительного перевода объектов в приемлемые режимы для защищенной/корректной работы «подзащитного».

g. «Ангел» может вызвать «архангела» и запросить у «архангела» выполнения запрета, защиты, открытия или реакции по поручению «подзащитного».

h. «Ангел» может выделить (например, по вызову или путем проверки памяти) выходные данные из приложения, проверить действительность таких данных (например, путем определения контрольной суммы и т.п.) и затем представить эти данные с использованием защищенных аппаратных средств вывода.

Часть функциональных возможностей объекта или приложения может быть перемещена к «ангелу». Аналогичным образом, часть функциональных возможностей ядра системы LHS может быть перемещена к «архангелу». Разработчик приложения может реализовать некоторые из функций приложения в составе «ангела». Хотя это увеличит нагрузку на систему RHS, но вместе с тем позволит перенести выполнение функций в доверительной среде. Аналогичным образом, часть ОС 31 системы LHS

может быть перемещена к «архангелу» 380.

«Ангел» может загружаться или вызываться различными способами. Программа системы LHS, такая как приложение 305, может вызвать своего «ангела» 390. Таким способом, например, после запуска приложения происходит загрузка соответствующего «ангела». Альтернативно, «ангел» может быть вызван из системы RHS, и затем «ангел» вызывает соответствующий процесс или приложение системы LHS. «Ангел» использует «архангела» для вызова в систему LHS и запроса запуска приложения. «Архангел» затем связывает «ангела» с приложением. В одном варианте осуществления интерфейсы API, которые нексус и «архангел» предоставляют «ангелу» приложения, позволяют ему видеть только процессы, которые он создает и, возможно, его дочерние объекты.

В качестве другой альтернативы, программа системы LHS может быть вызвана декларацией и затем направлена в систему RHS, которая запускает «ангела», который формирует обратный вызов в систему LHS для запуска соответствующего процесса или приложения системы LHS. В типовом случае программа системы LHS запускается именованием файла, который ее содержит (т.е. API, например: «исполнитьс: \некоторыйкаталог\некоторыйдругойкаталог\некотораяпрограмма.exe»). Код системы RHS (агент или «ангел») запускается именованием декларации, и декларация указывает двоичный код. Он не зависит от местоположения. Таким образом, декларации, например, обычно подписаны и сертифицированы, так что их намного труднее сымитировать. Таким образом, возможным механизмом могло быть представление объединенной декларации «левой/правой сторон» в систему RHS (нексус), что привело бы к запуску как приложения системы LHS, так и связанного с ней «ангела», и связыванию их вместе. Кроме того, «ангел» может быть использован для запуска приложения как из системы LHS, так и из системы RHS.

В одном варианте осуществления изобретения «архангел» может подтвердить, что первоначально загруженное отображение кода процесса системы LHS согласуется с отображением декларированного целевого кода, ассоциированного с «ангелом». Отображение декларированного целевого кода может быть обеспечено посредством декларации «ангела». Это препятствует коду, который представляется «ангелом» для конкретного приложения, запускать вместо этого другое приложение, что обеспечивает дополнительную защиту от атак.

Соответственно некоторым вариантам осуществления изобретения «ангелу» не разрешается изменять отображение кода приложения или процесса системы LHS, которые связаны с ним. «Ангел» может считывать и записывать данные, но он может только считывать код.

Эти и другие сходные стратегии могут быть использованы для предотвращения исполнения «ангелов» без контроля или ограничения вне системы LHS и воспрепятствования нестандартным «ангелам» имитации с использованием программ и приложений системы LHS.

В дополнение к механизмам инициирования, описанным выше, имеются другие пути убедиться в том, что корректный «ангел» присоединен к корректному приложению системы LHS (или RHS) и остается присоединенным к нему. Исполняемое приложение может быть изменено атакующим до вызова его агента, или вирус системы LHS может перехватить и осуществить перестановку вызова на некоторого другого «ангела».

Варианты осуществления настоящего изобретения могут решать данную проблему путем обработки вызовов из приложения к его «ангелу» через доверительный объект,

такой как «архангел» или нексус. Например, «архангел» может классифицировать вызывающее приложение системы LHS и сравнивать дайджест со списком «санкционированных» дайджестов, связанных с «ангелом» системы RHS. Если они не совпадают либо из-за перестановки приложения системы LHS, либо из-за того, что вызов был модифицирован для указания на другого «ангела», вызов не исполняется и система может предупредить пользователя и/или предпринять любые другие действия.

Стратегия системы может быть использована для определения того, какие ангелы можно присоединять к каким из приложений системы LHS. Использование строгого механизма стратегии обеспечивает устойчивость к имитации, внесению нарушений в механизм инициализации для установки таких зависимостей.

В некоторых вариантах осуществления «ангел» имеет настраиваемый уровень или различные программируемые уровни проверки угроз, направленных на ассоциированные приложения. Чувствительность «ангела» к восприятию угрозы или атаки может настраиваться.

В дополнение к обеспечению отображения (например, с использованием защиты, охраны, уведомления) на ОС или приложения системы LHS, «ангел» может также быть применен к агенту, исполняемому в доверительной вычислительной среде. В таком случае целевой агент (обычно «параноидный» объект) доверяет «ангелу», присоединенному к нему. Это позволяет процессу внешнего наблюдателя препятствовать различным ошибкам и трюкам в целевом агенте. «Ангел» может приводить в исполнение инварианты защиты в противоположность сканированию на обнаружение ошибок в обеспечении защиты (например, как в случае обычной противовирусной технологии) и использовать жесткое разделение и защиту процессов, которые обеспечивает нексус.

В одном варианте осуществления агент является виртуальной машиной, представляющей «эффективно идентичную дублирующую копию» некоторой реальной машины, в которой запущено отображение ОС. Доверительная среда может позволить агенту доступ к памяти процесса виртуальной машины. Получающий доступ агент может контролировать память процесса для защиты виртуальной машины от атак из отображения, которое она содержит. Доверительная среда может позволить «ангелу» защитить отображение ОС на виртуальной машине и позволить «ангелу» отобразить приложения на виртуальную машину. Подразумевается, что те же механизмы, которые обычно применяются к приложениям системы LHS, также применимы к среде виртуальной машины.

В одном варианте осуществления изобретения нексус обеспечивает «архангела» интерфейсом API для проверки памяти и изменения (как минимум). Обеспечиваемая API поддержка организации ловушек и реагирования на попытки изменения структур управления облегчает отображение. Например, на архитектуре x86 посредством API может быть обеспечена защита для структур управления таких, как GDT, LDT, IDT, регистры отладки, TR (маршрутизация сообщений) и т.д. Здесь GDT - глобальная таблица дескрипторов, LDT - локальная таблица дескрипторов. Закрытие GDTR (регистра глобальной таблицы дескрипторов) прерывает атаки, которые зависят от деформирования значения виртуальных адресов так, чтобы позволить переход в такие точки программ, куда атакующий обычно не может перейти. IDT означает таблицу дескрипторов прерываний, которая контролирует маршрутизацию прерываний. Местоположение IDT указывается посредством IDTR (регистра таблицы дескрипторов прерываний). Закрытие IDTR обеспечивает более эффективное отображение путем прерывания атак, в которых атакующий

использует IDT и отсроченное прерывание для форсирования ветвления (условного перехода) в коде, который они не могли достичь иным образом.

Желательно, чтобы доверительная среда (т.е. RHS) и открытая среда (т.е. LHS) соединялись некоторым образом. Соединение позволяет доверительной среде исследовать состояние и информироваться о событиях в открытой среде. Указанные здесь решения работают в структурах, включающих в себя, без ограничений указанным, приведенные ниже структуры:

1. Системы RHS и LHS на одной и той же машине, причем система RHS может непосредственно анализировать память системы LHS (в то время как система LHS не может анализировать память системы RHS без разрешения).

2. Системы RHS и LHS на разных процессорах, возможно с различной памятью, но шина, сеть, порт или другие средства соединения позволяют системе RHS просматривать память системы LHS. Например, обслуживающий процессор ARM (компании Advanced RISC Machines) может исполнять полностью доверительный (высоконадежный) стек, и этот стек имеет возможность контролировать память системы на процессоре x86 MP. Например, можно использовать машину с основными процессорами x86 и обслуживающий процессор модели ARM или PowerPC и использовать механизмы настоящего изобретения для обеспечения возможности обслуживающему процессору контролировать программное обеспечение на основных процессорах.

3. Если система RHS может принимать уведомление о событиях системы LHS (например, изменения в картах), но не может изменять или препятствовать им, или не может просматривать память системы LHS, некоторая часть отображения (например, слабая часть) все еще возможна.

4. Система RHS может проверять память системы LHS по желанию, может контролировать (например, препятствовать или изменять) редактирования в системе LHS отображения памяти и структуры адресных преобразований, и контролировать, куда указывает вектор координации прерываний (но не требует управления контроллером прерываний, даже если такое управление обеспечивается, в этом обеспечивается необходимый баланс). Подразумевается, что определение списка состояний/событий, который система RHS желательным образом способна полностью контролировать для поддержки строго отображения, представляет собой задачу, которая должна быть поставлена перед каждой архитектурой процессора, и специалисту в данной области техники должно быть понятно, что этот список будет различаться для различных архитектур.

5. В одном варианте осуществления изменения в регистре x86 TR и установка регистров отладки аппаратных средств также управляются со стороны системы RHS.

В известных аппаратных средствах не гарантируется исполнение доверительной среды, поскольку это может зависеть от обычных аппаратных средств прерывания, таблицы координации прерываний системы LHS и т.д.

В аппаратных средствах, описанных выше, способность контролировать IDT (на процессоре x86 или эквивалентном ему) позволяет системе RHS гарантировать, что некоторое прерывание по ее выбору всегда будет исполнять код, который вызывает систему RHS.

Однако атака на систему LHS или ошибка может разрушить контроллер, отклонить прерывания и т.д. Подразумевается, что используется ATC (управление преобразованием адреса) для обеспечения того, что система RHS начнет исполнять каждое из них. Если система RHS использует ATC, она может модифицировать ATC

для сообщения приращения счетчику. Счетчик устанавливается на некоторое значение всякий раз, когда система RHS планирует исполнение «архангела». Если счетчик достигает нуля, то АТС знает, что «архангел» не исполнялся «слишком долго», и вызывает точку входа нексуса, который форсирует исполнение «архангела». Этот метод не гарантирует, что «архангел» исполняется в любое конкретное время, но гарантирует, что он будет исполняться спустя ряд операций редактирования памяти системы LHS. Таким образом, система LHS, которая находится в активном состоянии, будет в конечном счете приводить к исполнению «архангела».

Если система RHS может закрыть таблицу IDT, и система имеет надежный источник NMI (немаскируемых прерываний), то система RHS может форсировать обработчика NMI для вызова надлежащим образом.

В приведенном для примера варианте осуществления аппаратные средства содержат таймер, который форсирует прерывание в системе RHS после заданного числа импульсов сигнала времени.

Настоящее изобретение обеспечивает механизмы, которые позволяют доверительность одной вычислительной среды отобразить на вторую вычислительную среду. Две или более операционных систем, которые исполняются на одной машине, являются примерами «сред», которые могут потребовать взаимодействия друг с другом на одной машине, хотя понятно, что изобретение не ограничивается традиционной операционной системой. Более того, по меньшей мере некоторые из описанных методов могут быть использованы в общем случае для отображения доверительности из любого типа исполняемого объекта (например, любой части программного обеспечения) на любой другой тип объекта.

В случае, когда два объекта существуют рядом друг с другом на одной машине и требуют взаимодействия друг с другом, взаимодействие может принимать различные формы. Например, двум объектам может потребоваться передавать данные друг другу. В случае, когда объекты являются операционными системами (или некоторыми другими типами сред исполнения, таких как машины сценариев, которые исполняют сценарии на виртуальной машине) объектам может потребоваться взаимодействовать друг с другом некоторыми иными путями - например, совместным использованием памяти, совместным использованием времени процессора, совместным использованием ресурсов и обработкой прерываний. Изобретение обеспечивает методы, которыми два объекта могут вступать во взаимодействия этих типов друг с другом, при этом позволяя одному объекту отображать свой доверительный характер на другой объект.

Варианты осуществления, описанные выше, фокусируются на памяти в качестве контролируемого ресурса, однако, изобретение не ограничивается этим. Если имеются средства контроля защиты для ресурсов иных, чем память, основной контролирующий агент (например, «архангел») может использовать такие средства контроля, как доверенных представителей для расширения своей сферы доверительности. Например, если имеется защищенный NIC (сетевой адаптер), то основной контролирующий агент может использовать его, чтобы препятствовать посылке пакетов с определенными заголовками. В принципе, такой доверенный представитель должен только понимать инвариант измерений, например, заголовки, согласующиеся с <regexp>, и надежным образом уведомлять контролирующего агента об изменениях в инварианте.

Следует отметить, что приведенные выше примеры обеспечены только с целью пояснения и ни в коей мере не должны интерпретироваться как ограничивающие

настоящее изобретение. Хотя изобретение описано со ссылками на различные варианты осуществления, понятно, что использованные термины являются лишь терминами описания и иллюстрации, а не терминами ограничения. Кроме того, хотя изобретение описано со ссылками на конкретные средства, материалы и варианты осуществления, изобретение не ограничивается раскрытыми деталями, а распространяется на все функционально эквивалентные структуры, методы и применения, входящие в объем формулы изобретения. Специалисты в данной области техники на основе решений, представленных в описании, могут осуществить различные модификации и изменения без отклонения от объема и сущности изобретения.

Формула изобретения

1. Система для контроля незащищенной среды, содержащая
незащищенную среду,
высоконадежную среду и
по меньшей мере один основной контролирующий агент в высоконадежной среде, который контролирует незащищенную среду, причем упомянутый по меньшей мере один контролирующий агент отклоняет разрешение на изменения в незащищенной среде без приема санкционирования через защищенный вход.

2. Система по п.1, дополнительно содержащая множество контролирующих агентов, исполняемых в высоконадежной среде, причем каждый контролирующий агент контролирует по меньшей мере одно приложение, расширение или компонент, исполняемый в незащищенной среде.

3. Система по п.2, в которой каждый контролирующий агент ассоциирован с приложением, причем каждый контролирующий агент контролирует ассоциированное с ним приложение на наличие атаки или противоречивостей, при этом отображая доверительность высоконадежной среды на незащищенную среду.

4. Система по п.3, в которой каждый контролирующий агент может содержать часть приложения, с которым он ассоциирован.

5. Система по п.3, в которой каждый контролирующий агент имеет настраиваемый уровень проверки для определения угроз ассоциированному приложению.

6. Система по п.3, в которой каждый контролирующий агент может принимать защищенный ввод данных и переносить защищенный ввод в ассоциированное приложение.

7. Система по п.2, в которой по меньшей мере один из контролирующих агентов контролирует агенты приложений, исполняемых в высоконадежной среде.

8. Система по п.2, дополнительно содержащая другой контролирующий агент в высоконадежной среде, причем контролирующие агенты осуществляют информационный обмен друг с другом.

9. Система по п.1, в которой основной контролирующий агент обнаруживает противоречивости в незащищенной среде.

10. Система по п.9, дополнительно содержащая множество контролирующих агентов, которые обнаруживают противоречивости в приложениях, исполняемых в незащищенной среде.

11. Система по п.10, дополнительно содержащая дополнительные контролирующие агенты, которые обнаруживают противоречивости в приложениях, исполняемых в защищенной среде.

12. Система по п.1, в которой по меньшей мере один из основных контролирующих

агентов санкционирует или отвергает событие в незащищенной среде.

13. Система по п.12, в которой по меньшей мере один основной контролирующий агент содержит защищенный вход для приема ввода данных, причем санкционирование или отклонение основным контролирующим агентом базируется на принятом вводе данных.

14. Система по п.1, в которой по меньшей мере один из основных контролирующих агентов отклоняет разрешение на изменения в незащищенной среде, если только изменения не описаны пакетом, который заверен подписью полномочной стороны.

15. Система по п.1, дополнительно содержащая контролирующий агент, исполняемый в высоконадежной среде, причем контролирующий агент контролирует по меньшей мере одно приложение, расширение или компонент, исполняемый в незащищенной среде, при этом контролирующий агент использует закрытую память для хранения секретных данных для операционной системы или приложения, помещенного в незащищенной среде.

16. Система по п.15, в которой контролирующий агент отказывает в открытии секретных данных операционной системе или приложению, если только операционная система или приложение не имеют дайджест, который согласован с владельцем секретных данных.

17. Система по п.15, в которой контролирующий агент отказывает в открытии секретных данных операционной системе или приложению, если только операционная система или приложение не включены в список дайджестов, которые могут читать секретные данные.

18. Система по п.15, в которой контролирующий агент использует предварительно определенный тест для определения того, запрашиваются ли секретные данные легитимным объектом.

19. Система по п.18, в которой предварительно определенный тест включает проверку стеков объекта и удостоверение, что стеки имеют легальное содержимое стеков.

20. Система по п.1, дополнительно содержащая контролирующий агент, исполняемый в высоконадежной среде, причем контролирующий агент контролирует по меньшей мере одно приложение, расширение или компонент, исполняемый в незащищенной среде, причем контролирующий агент редактирует состояние незащищенной среды, чтобы сделать ее защищенной или иным образом приемлемой.

21. Система по п.20, в которой состояние включает в себя начальную конфигурацию или опцию сообщения об ошибке.

22. Система по п.1, в которой основной контролирующий агент обнуляет физическую память, которая не принадлежит к известной хорошей конфигурации незащищенной среды или к высоконадежной среде.

23. Система по п.1, в которой незащищенная среда включает в себя базовую систему ввода/вывода (BIOS), программно-аппаратные средства или загрузчик.

24. Система по п.1, дополнительно содержащая нексус для исполнения основного контролирующего агента при начальной загрузке.

25. Система по п.1, дополнительно содержащая счетчик в высоконадежной среде, причем счетчик используется для определения того, должен ли исполняться основной контролирующий агент.

26. Система по п.25, в которой счетчик отсчитывает число операций редактирования незащищенной памяти.

27. Способ контроля незащищенной среды, заключающийся в том, что

обеспечивают незащищенную среду,

обеспечивают по меньшей мере один основной агент в высоконадежной среде и

контролируют незащищенную среду на наличие атаки или противоречивостей для отображения доверительности высоконадежной среды на незащищенную среду, причем упомянутый по меньшей мере одни основной контролирующий агент отклоняет разрешение на изменения в незащищенной среде без приема санкционирования через защищенный вход.

28. Способ по п.27, в котором дополнительно

обеспечивают множество контролирующих агентов, исполняемых в высоконадежной среде, и

контролируют по меньшей мере одно приложение, расширение или компонент, исполняемый в незащищенной среде.

29. Способ по п.28, в котором дополнительно

ассоциируют каждого контролирующего агента с приложением и контролируют каждое ассоциированное приложение на наличие атаки или противоречивостей.

30. Способ по п.28, в котором дополнительно ассоциируют приложение с одним из контролирующих агентов и переносят часть приложения в контролирующий агент, чтобы упомянутая часть приложения содержалась в высоконадежной среде.

31. Способ по п.28, в котором дополнительно ассоциируют приложение с контролирующим агентом и настраивают уровень проверки в контролирующем агенте для определения угроз ассоциированному приложению.

32. Способ по п.28, в котором дополнительно ассоциируют приложение с контролирующим агентом, принимают защищенный ввод данных в контролирующем агенте и переносят защищенный ввод в ассоциированное приложение.

33. Способ по п.28, в котором контролирующие агенты осуществляют информационный обмен друг с другом.

34. Способ по п.27, в котором основной контролирующий агент санкционирует или отвергает событие в незащищенной среде.

35. Способ по п.34, в котором основной контролирующий агент получает ввод данных с защищенного входа.

36. Способ по п.27, в котором основой контролирующий агент отклоняет разрешение на изменения в незащищенной среде, если только изменения не описаны пакетом, который заверен подписью полномочной стороны.

37. Способ по п.27, в котором дополнительно

обеспечивают множество контролирующих агентов, исполняемых в высоконадежной среде, при этом один из контролирующих агентов использует закрытую память для хранения секретных данных для операционной системы или приложения, помещенного в незащищенной среде.

38. Способ по п.37, в котором контролирующий агент отказывает в открытии секретных данных операционной системе или приложению, если только операционная система или приложение не имеют дайджест, который согласован с владельцем секретных данных.

39. Способ по п.37, в котором контролирующий агент отказывает в открытии секретных данных операционной системе или приложению, если только операционная система или приложение не включены в список дайджестов, которые могут читать секретные данные.

40. Способ по п.37, дополнительно содержащий использование предварительно определенного теста для определения того, запрашиваются ли секретные данные

легитимным объектом.

41. Способ по п.40, в котором предварительно определенный тест включает проверку стеков объекта и удостоверение, что стеки имеют легальное содержимое стеков.

42. Способ по п.27, в котором дополнительно

обеспечивают множество контролирующих агентов, исполняемых в высоконадежной среде,

причем один из контролирующих агентов редактирует состояние незащищенной среды, чтобы сделать ее защищенной или иным образом приемлемой.

43. Способ по п.42, в котором состояние включает в себя начальную конфигурацию или опцию сообщения об ошибке.

44. Способ по п.27, в котором основной контролирующий агент обнуляет физическую память, которая не принадлежит к известной хорошей конфигурации незащищенной среды или к высоконадежной среде.

45. Способ по п.27, в котором незащищенная среда включает в себя базовую систему ввода/вывода (BIOS), программно-аппаратные средства или загрузчик.

46. Способ по п.27, дополнительно содержащий исполнение основного контролирующего агента при начальной загрузке посредством нексуса.

47. Способ по п.27, дополнительно содержащий определение того, должен ли исполняться основной контролирующий агент в ответ на состояние счетчика.

48. Способ по п.47, в котором счетчик отсчитывает число операций редактирования незащищенной памяти.

49. Способ по п.27, в котором обеспечивают множество контролирующих агентов, которые обнаруживают противоречивости в приложениях, исполняемых в незащищенной среде.

50. Способ по п.49, дополнительно содержащий обеспечение дополнительных контролирующих агентов, которые обнаруживают противоречивости в приложениях, исполняемых в защищенной среде.

51. Система для контроля незащищенной среды, содержащая высоконадежную среду, имеющую по меньшей мере одно из операционной системы, программно-аппаратных средств и базовой системы ввода/вывода (BIOS), незащищенную среду и

по меньшей мере один контролирующий агент, исполняемый в высоконадежной среде и ассоциированный с по меньшей мере одним из операционной системы, программно-аппаратных средств, системы BIOS и приложения, исполняемого в незащищенной среде, причем упомянутый по меньшей мере один контролирующий агент отклоняет разрешение на изменения в незащищенной среде без приема санкционирования через защищенный вход.

52. Система по п.51, в которой по меньшей мере один контролирующий агент содержит множество контролирующих агентов, причем каждый из контролирующих агентов имеет ассоциированную с ним функцию.

53. Система по п.51, в которой высоконадежная среда исполняет архитектуру первого процессора, а незащищенная среда исполняет архитектуру второго процессора, причем система дополнительно содержит основной контролирующий агент, исполняемый на первом процессоре.

54. Система по п.51, в которой высоконадежная среда и незащищенная среда исполняются на одном и том же процессоре, причем система дополнительно содержит основной контролирующий агент, исполняемый в высоконадежной среде.

55. Система по п.51, в которой высоконадежная среда исполняется на первом

процессоре, а незащищенная среда исполняется на втором процессоре, причем первый и второй процессоры имеют возможность работать как в высоконадежном режиме, так и в незащищенном режиме.

56. Система по п.51, дополнительно содержащая нексус и основной контролирующий агент, помещенные в высоконадежной среде, причем основной контролирующий агент присоединен, связан или компилирован в нексус.

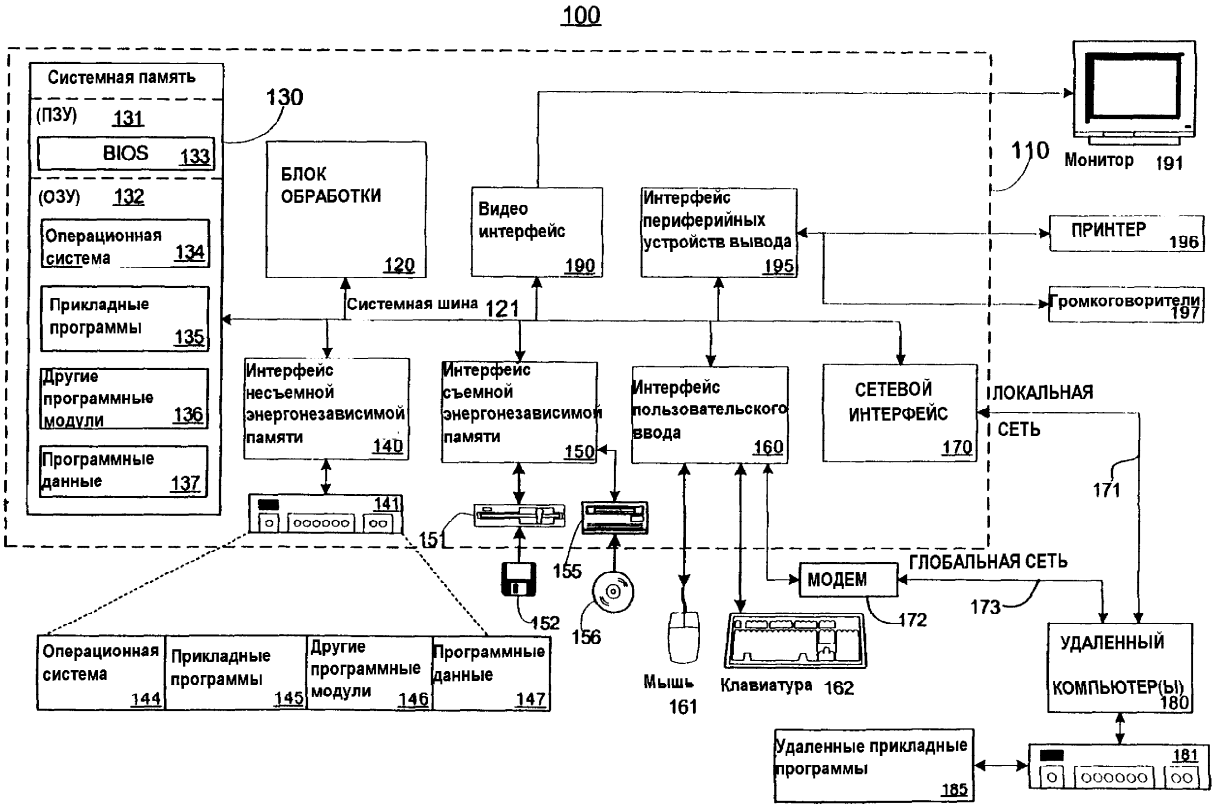
57. Система по п.51, дополнительно содержащая нексус и основной контролирующий агент, помещенные в высоконадежной среде, причем основной контролирующий агент является процессором пользовательского режима, работающим на нексусе.

58. Система по п.51, дополнительно содержащая основной контролирующий агент, помещенный в высоконадежной среде и разработанный в той же или связанной конструктивной среде, что и операционная система незащищенной среды.

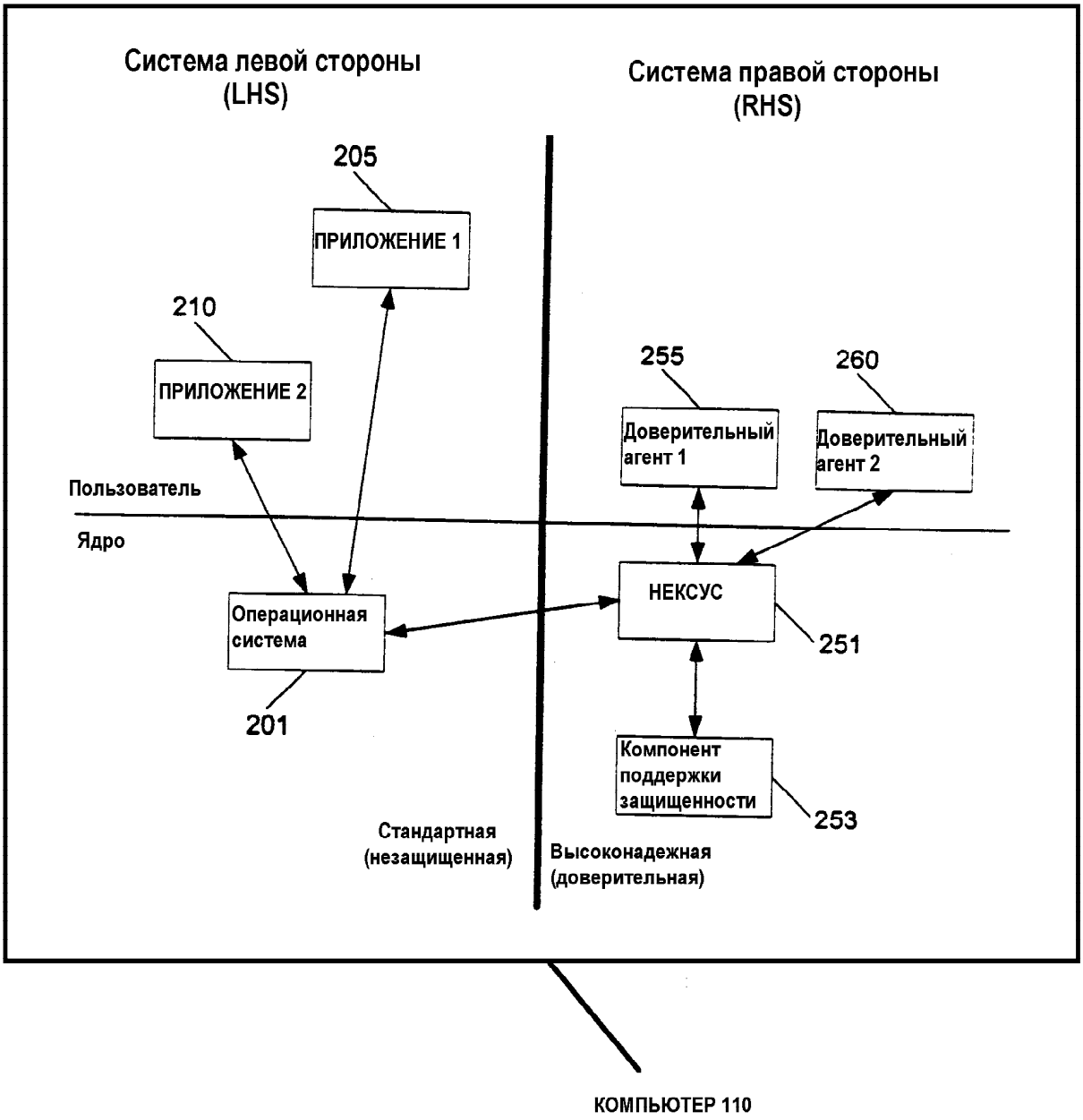
59. Система по п.51, дополнительно содержащая основной контролирующий агент, помещенный в высоконадежной среде, причем основной контролирующий агент является частью высоконадежной вычислительной базы для оценки защищенности.

60. Система по п.51, дополнительно содержащая основной контролирующий агент, первая часть основного контролирующего агента помещена в высоконадежной среде, а вторая часть основного контролирующего агента помещена на физически удаленной машине, причем первая и вторая части соединены защищенным каналом связи.

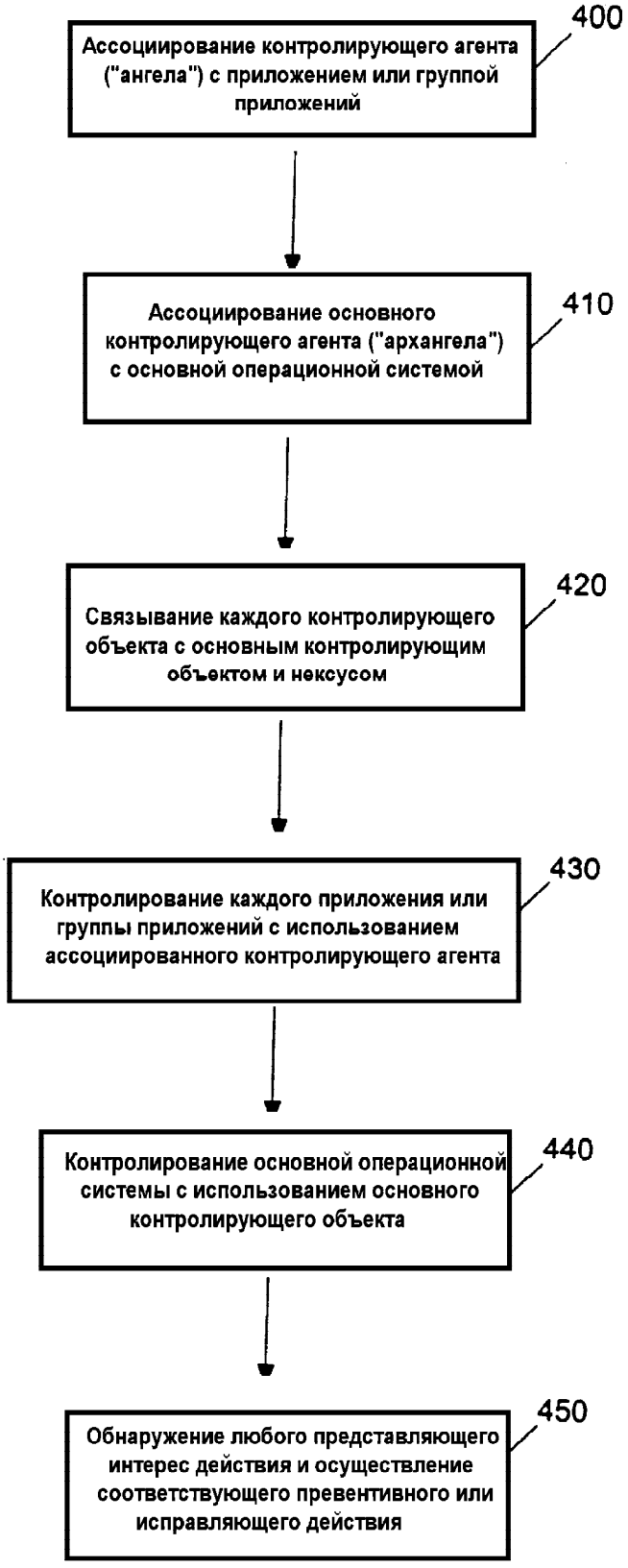
ВЫЧИСЛИТЕЛЬНАЯ СРЕДА



ФИГ.1


Система левой стороны
(LHS)
Система правой стороны
(RHS)
205
ПРИЛОЖЕНИЕ 1
210
ПРИЛОЖЕНИЕ 2
255
Доверительный агент 1
260
Доверительный агент 2
Пользователь
Ядро
Операционная система
201
НЕКСУС
251
Компонент поддержки защищенности
253
Стандартная
(незащищенная)
Высоконадежная
(доверительная)
КОМПЬЮТЕР 110

ФИГ.2


400
Ассоциирование контролирующего агента ("ангела") с приложением или группой приложений
410
Ассоциирование основного контролирующего агента ("архангела") с основной операционной системой
420
Связывание каждого контролирующего объекта с основным контролирующим объектом и нексусом
430
Контролирование каждого приложения или группы приложений с использованием ассоциированного контролирующего агента
440
Контролирование основной операционной системы с использованием основного контролирующего объекта
450
Обнаружение любого представляющего интерес действия и осуществление соответствующего превентивного или исправляющего действия

ФИГ.4