РОССИЙСКАЯ ФЕДЕРАЦИЯ

(19) **RU** (11) **2 439 665** (13) **C2**

(51) МПК
G06F 9/45 (2006.01)

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(21)(22) Заявка: **2009120207/08, 27.11.2007**

(24) Дата начала отсчета срока действия патента:
27.11.2007

Приоритет(ы):
(30) Конвенционный приоритет:
28.11.2006 US 11/564,249

(43) Дата публикации заявки: **10.12.2010** Бюл. № 34

(45) Опубликовано: **10.01.2012** Бюл. № 1

(56) Список документов, цитированных в отчете о поиске: **US 2004/0226007 A1, 11.11.2004. US 6978018 B2, 20.12.2005. US 2004/0255268 A1, 16.12.2004. US 2006/0123403 A1, 08.06.2006. RU 2286595 C2, 10.12.2004.**

(85) Дата начала рассмотрения заявки PCT на национальной фазе: **27.05.2009**

(86) Заявка PCT:
US 2007/085664 (27.11.2007)

(87) Публикация заявки PCT:
WO 2008/067329 (05.06.2008)

Адрес для переписки:
129090, Москва, ул. Б. Спасская, 25, стр.3, ООО "Юридическая фирма Городисский и Партнеры", А.В.Мицу

(72) Автор(ы):
РАЙТОН Дэвид Чарльз (US),
ЮНОКИ Роберт Садао (US)

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

RU 2439665 C2

(54) КОМПИЛЯЦИЯ ИСПОЛНЯЕМОГО КОДА В МЕНЕЕ ДОВЕРЯЕМОМ АДРЕСНОМ ПРОСТРАНСТВЕ

(57) Реферат:

Изобретение относится к средствам компиляции программного кода. Технический результат заключается в повышении безопасности операционной системы при компиляции небезопасного программного кода. Выполняют прикладную программу из первого адресного пространства, заданного с первым набором разрешений для доступа к совместно используемой динамически распределяемой памяти. Получают один или более запросов от прикладной программы на компиляцию одного или более множеств команд на промежуточном языке. Компилируют одно или более множеств команд на промежуточном языке в заново скомпилированный код с использованием JIT-компилятора, работающего во втором адресном пространстве, которое обладает вторым набором разрешений для доступа к совместно используемой динамически распределяемой памяти. Передают заново скомпилированный код в совместно используемую динамически распределяемую

память, в которой прикладная программа может извлечь и выполнить заново скомпилированный код из первого адресного пространства. 3 н. и 17 з.п. ф-лы, 4 ил.

Фиг.1А

RU 2439665 C2

RU 2439665 C2

RUSSIAN FEDERATION

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY

(19) **RU** (11) **2 439 665** (13) **C2**

(51) Int. Cl.
G06F 9/45 (2006.01)

(12) ABSTRACT OF INVENTION

(21)(22) Application: **2009120207/08, 27.11.2007**

(24) Effective date for property rights:
27.11.2007

Priority:
(30) Priority:
28.11.2006 US 11/564,249

(43) Application published: **10.12.2010** Bull. 34

(45) Date of publication: **10.01.2012 Bull. 1**

(85) Commencement of national phase: **27.05.2009**

(86) PCT application:
US 2007/085664 (27.11.2007)

(87) PCT publication:
WO 2008/067329 (05.06.2008)

Mail address:
129090, Moskva, ul. B. Spasskaja, 25, str.3, OOO "Juridicheskaja firma Gorodisskij i Partnery", A.V.Mitsu

(72) Inventor(s):
RAJTON Dehvid Charl'z (US),
JuNOKI Robert Sadao (US)

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

RU 2439665 C2

(54) COMPILATION OF EXECUTABLE CODE IN LESS TRUSTWORTHY ADDRESS SPACE

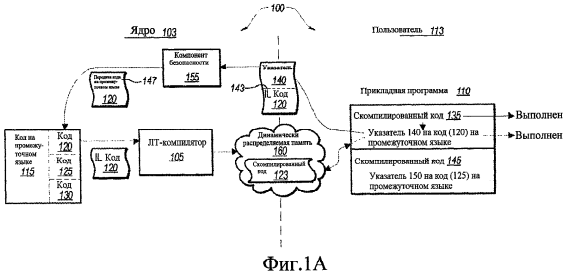
(57) Abstract:

FIELD: information technologies.

SUBSTANCE: applied programme is executed from the first address space specified with the first set of permits for access to a jointly used dynamically distributed memory. One or more inquiries are received from the applied programme to compile one or more multitudes of commands in an intermediate language. One or more command multitudes are complied in the intermediate language into a newly compiled code with application of an LT-compiler, operating in the second address space, which has the second set of permits for access to jointly used dynamically distributed memory. The newly compiled code is transferred into a jointly used dynamically distributed memory, in which the applied programme may extract and fill the newly compiled code from the first address space.

EFFECT: increased safety of operational system in compilation of unsafe programme code.

20 cl, 4 dwg



Фиг.1А

RU 2439665 C2

УРОВЕНЬ ТЕХНИКИ

Увеличилась популярность как компьютеризированных систем, так и различных прикладных программ, используемых на компьютеризированных системах. В частности, сейчас существует широкий диапазон прикладных программ, сконфигурированных для любого количества назначений, чтобы функционировать либо в качестве комплексных операционных систем, баз данных и так далее, либо в качестве простого вычислителя. Во многих случаях разработчики программного обеспечения будут писать новые прикладные программы, подразумевая конкретную операционную систему, с использованием любого количества подходящих языков. Как только программное обеспечение завершено, разработчик будет компилировать приложение в исполняемый компьютером код, который затем может устанавливаться на компьютерную систему с подходящей операционной системой.

Поэтому признаем, что имеется некоторое количество соображений, которое часто должно продумываться разработчиками операционных систем, а также отдельных прикладных программ. Многие из этих интересов могут быть даже конкурирующими. Например, многие разработчики прикладных программ могут иметь интересы, относящиеся к быстрой работе, тогда как многие разработчики операционных систем могут иметь интересы, относящиеся к безопасности и стабильности. В некоторых случаях требования безопасности и стабильности могут вызвать более медленное исполнение у некоторых прикладных программ и/или более низкую производительность.

Например, операционная система может быть сконфигурирована для работы прикладных программ на менее доверяемом "пользовательском" уровне, но другие системные компоненты работают на доверяемом уровне "ядра". В результате прикладная программа, работающая на пользовательском уровне, может иметь возможность выполнять только некоторые типы функций путем запроса заданной функции через промежуточный доверяемый компонент. Промежуточный компонент может затем проверить достоверность запроса и затем передать запрос на функцию компоненту уровня ядра, который может затем выполнить запрос.

Другими путями управления безопасностью являются ограничение различных приложений и компонентов конкретными доступными для чтения, записи и/или исполнения пространствами разрешений. Например, операционная система могла бы разрешить некоторым прикладным программам работать только в адресном пространстве чтения/выполнения. Это могло бы позволить прикладным программам выполнять любые существующие команды, но препятствовало бы приложению выполнять любые операции записи. В отличие от этого, операционная система могла бы разрешить другим чувствительным системным компонентам работать только в адресном пространстве чтения/записи. Это могло бы позволить чувствительным компонентам делать новые записи, но запретило бы выполнение тех записей.

В еще одних случаях операционная система могла бы разрешить только некоторым типам прикладных программ, соответствующим определенным стандартам кода, работать в пространстве, которое доступно для чтения, записи и выполнения. Например, операционная система могла бы разрешить только приложениям "с типовой безопасностью" работать в адресном пространстве чтения/записи/выполнения. Одним примером правила типовой безопасности могло бы быть требование сложения целочисленного значения только с другими целочисленными значениями, а не значениями с плавающей запятой. Тогда мог бы использоваться компилятор с типовой безопасностью, чтобы компилировать только

тот исполняемый программный код, который обладает типовой безопасностью, и соответственно является доверяемым операционной системой.

К сожалению, некоторые последние направления в разработке прикладных программ усложняют различные стороны вышеупомянутых подходов к управлению безопасностью. Например, широкий диапазон разработчиков приложений создают сейчас прикладные программы видеоигр с использованием "управляемого кода". Вообще, управляемый код включает в себя исполняемый программный код, а также код на промежуточном языке, который может компилироваться по мере необходимости. Например, разработчик прикладной программы мог бы включить одну или более ссылок (в скомпилированный исполняемый код) на промежуточный код. Таким образом, когда исполняемый код подходит к точке, где ему нужно использовать функцию, которая доступна только в коде на промежуточном языке, используется JIT-компилятор (Just-In-Time, динамический) для компилирования некоторого кода на промежуточном языке в исполняемые команды.

Поэтому можно признать, что операционные системы иногда будут ограничивать использование управляемого кода приложениями с типовой безопасностью. В частности, поскольку JIT-компилятору потребуется запись, и поскольку приложению потребуется выполнение, и еще потому что прикладной программе потребуется обращаться к скомпилированному коду, записанному JIT-компилятором, JIT-компилятор и исполняемая прикладная программа, как правило, будут работать в одном адресном пространстве, которое доступно для чтения, записи и выполнения. Таким образом, если код на промежуточном языке был небезопасным по типу (или соответствующим другим ограничениям программного кода), то злоумышленник мог обманом заставить JIT-компилятор сформировать вредоносные команды, которые выполняются.

К сожалению, ограничения программного кода, такие как типовая безопасность, часто считают противоречащими соображениям скорости и производительности. Это, в частности, может быть проблематичным для приложений видеоигр, где соображения скорости и производительности ставятся на первое место. Поэтому в некоторых случаях разработчики приложений видеоигр могут обнаружить, что лучше или эффективнее игнорировать определенные спецификации кода, например, типовую безопасность.

КРАТКАЯ СУЩНОСТЬ ИЗОБРЕТЕНИЯ

Реализации настоящего изобретения предоставляют системы, способы и компьютерные программные продукты, сконфигурированные для допущения использования управляемого кода в операционной системе, где управляемый код необязательно может соответствовать какому-нибудь отдельному стандарту кода. Например, в одной реализации операционная система обеспечивает доступ к ячейке памяти в двух разных адресных пространствах, и задает разрешения в адресных пространствах из условия, чтобы ячейка памяти была доступна с разными разрешениями из двух разных адресных пространств. В одной реализации JIT-компилятор, работающий в одном адресном пространстве, передает скомпилированный код в совместно используемую динамически распределяемую память. Исполняемый программный код, в свою очередь, обращается к скомпилированному коду из динамически распределяемой памяти и выполняет его в другом адресном пространстве в памяти.

Например, способ выполнения управляемого кода, чтобы недоверяемый программный код мог компилироваться и выполняться способом, который не

представляет опасности или иным образом подвергает риску безопасность системы, может включать в себя выполнение прикладной программы в первом адресном пространстве ячейки памяти. Способ также может включать в себя получение одного или более запросов от прикладной программы на компиляцию одного или более множеств команд на промежуточном языке. К тому же, способ может включать в себя компиляцию одного или более множеств команд на промежуточном языке в заново скомпилированный код с использованием JIT-компилятора, работающего во втором адресном пространстве ячейки памяти. Кроме того, способ может включать в себя передачу заново скомпилированного кода в совместно используемую динамически распределяемую память. Прикладная программа может затем извлечь заново скомпилированный код из совместно используемой динамически распределяемой памяти в первое адресное пространство.

Аналогичным образом, другой способ формирования исполняемого компьютером программного кода способом, который использует JIT-компиляцию наряду с предотвращением нарушений безопасности, может включать в себя получение кода прикладной программы, который включает в себя исполняемый код и код, который необходимо компилировать. Способ также может включать в себя выполнение исполняемого кода в режиме с меньшими привилегиями и в первом адресном пространстве. К тому же, способ может включать в себя идентификацию одного или более указателей в исполняемом коде по меньшей мере для некоторого кода, который необходимо компилировать. Кроме того, способ может включать в себя переключение в режим с большими привилегиями. Более того, способ может включать в себя компиляцию по меньшей мере некоторого кода в другом адресном пространстве, используя компилятор, работающий в режиме с большими привилегиями.

Данная сущность изобретения предоставляется, чтобы представить набор идей в упрощенном виде, которые дополнительно описываются ниже в подробном описании. Данная сущность изобретения не предназначена для определения ключевых признаков или существенных признаков заявленного предмета изобретения, и также не предназначена для использования в качестве содействия в определении объема заявленного предмета изобретения.

Дополнительные признаки и преимущества изобретения будут изложены в последующем описании и частично будут очевидны из этого описания, либо могут быть изучены при применении изобретения на практике. Признаки и преимущества изобретения могут быть реализованы и получены посредством инструментов и сочетаний, подробно указанных в прилагаемой формуле изобретения. Эти и другие признаки настоящего изобретения станут более очевидными из нижеследующего описания и прилагаемой формулы изобретения, либо могут быть изучены при применении изобретения на практике, как изложено далее.

КРАТКОЕ ОПИСАНИЕ ЧЕРТЕЖЕЙ

Чтобы описать способ, которым могут быть получены вышеупомянутые и другие преимущества и признаки изобретения, более конкретное описание изобретения, кратко описанного выше, будет представлено посредством ссылки на его конкретные варианты осуществления, которые иллюстрируются на прилагаемых чертежах. Предполагая, что эти чертежи изображают только типичные варианты осуществления изобретения и поэтому не должны рассматриваться как ограничивающие его объем, изобретение будет описываться и объясняться с помощью дополнительной специфики и подробностей посредством использования прилагаемых чертежей, на которых:

Фиг. 1A иллюстрирует обзорную принципиальную схему реализации в соответствии с настоящим изобретением, в которой прикладная программа, работающая в менее доверяемом режиме безопасности, вызывает управляемый код, который компилируется с помощью JIT-компилятора в доверяемом режиме безопасности;

Фиг. 1B иллюстрирует принципиальную схему, в которой ячейка памяти, управляемая операционной системой, является доступной с помощью компонентов в двух разных адресных пространствах, которые обладают разными разрешениями для доступа к ячейке памяти;

Фиг. 2 иллюстрирует блок-схему последовательности действий в соответствии с реализацией настоящего изобретения, в которой JIT-компилятор получает и обрабатывает один или более запросов для команд на промежуточном языке; и

Фиг. 3 иллюстрирует блок-схему общей последовательности действий, в которой операционная система получает прикладную программу, которая включает в себя одну или более ссылок на управляемый код, и выполняет прикладную программу в соответствии с одним или более механизмами безопасности.

ПОДРОБНОЕ ОПИСАНИЕ

Реализации настоящего изобретения распространяются на системы, способы и компьютерные программные продукты, сконфигурированные для допуска использования управляемого кода в операционной системе, где управляемый код необязательно может соответствовать какому-нибудь отдельному стандарту кода. Например, в одной реализации операционная система обеспечивает доступ к ячейке памяти в двух разных адресных пространствах, и задает разрешения в адресных пространствах из условия, чтобы ячейка памяти была доступна с разными разрешениями из двух разных адресных пространств. В одной реализации JIT-компилятор, работающий в одном адресном пространстве, передает скомпилированный код в совместно используемую динамически распределяемую память. Исполняемый программный код, в свою очередь, обращается к скомпилированному коду из динамически распределяемой памяти и выполняет его в другом адресном пространстве в памяти.

Как будет в полной мере понятно в этом документе, реализации настоящего изобретения могут предоставить надежную систему без обязательной необходимости в проверке того, что сформированный код не нарушает ограничения в безопасности у системы. Это может быть выполнено, по меньшей мере частично, путем "помещения в песочницу" скомпилированного кода, а также любого другого кода, который выполняется. В частности, реализации настоящего изобретения могут определить "песочницу", которая по существу является предопределенным множеством границ, в котором может выполняться любой тип кода. В частности, границы песочницы, описываемые в этом документе, приведут к тому, что злоумышленный запрос(ы), сделанный путем выполнения кода, будет либо отклонен операционной системой (как поступающий от компонента пользовательского режима), либо ограничен в действиях или функциями только в рамках предопределенных разрешений (например, запрет записи в адресное пространство чтения/выполнения).

В результате код, который компилируется JIT-компилятором (например, 105), или даже прикладная программа (например, 110), в конечном счете вызывающая JIT-компилятор, может выполняться в "песочнице" без необходимости быть "безопасной по типу" или соответствовать каким-нибудь другим соображениям безопасности. Признаем, что это может дать свободу данному разработчику в написании кода прикладной программы способом, который теоретически менее ограничен и

теоретически быстрее и производительнее, нежели ранее возможный.

В дополнение к обеспечению того, что код выполняется должным образом, реализации настоящего изобретения также предоставляют механизмы, которые гарантируют, что сам JIT-компилятор не может быть "захвачен", например, при получении и компиляции кода на промежуточном языке. В частности, реализации настоящего изобретения включают в себя JIT-компилятор, который конфигурируется для выполнения с типовой безопасностью вместо обязательной проверки входящего кода на типовую безопасность или компиляции только кода, безопасного по типу. По существу, JIT-компилятор в соответствии с реализациями настоящего изобретения может быть защищен от запросов, которые заставили бы сам JIT-компилятор нарушить определения безопасности (например, определения типовой безопасности).

Например, в одной реализации JIT-компилятор может быть сконфигурирован с определениями типовой безопасности, которые ограничивают JIT-компилятор от доступа за пределы его собственных структур данных, или структур данных, которые определены как часть периода выполнения системы 100. Например, JIT-компилятор может быть сконфигурирован для выполнения серий проверок, чтобы гарантировать, что выполняются только допустимые приведения, всякий раз при выполнении приведений из одного типа в другой. Аналогичным образом, JIT-компилятор может быть сконфигурирован так, что всякий раз при запросе чтения из массивов JIT-компилятор выполняет одну или более проверок границ, чтобы гарантировать, что JIT-компилятор находится в пределах границ массива. Относительно использования в рамках языка программирования C, например, JIT-компилятор также может конфигурировать для гарантии того, что всякий раз при использовании "объединения" JIT-компилятор считывает или записывает в надлежащую часть объединения. Кроме того, JIT-компилятор может быть сконфигурирован для гарантии того, что JIT-компилятор никогда не переполнит или опустошит во время чтения или записи стека типов (стека типов внутри JIT-компилятора).

Вообще, стек типов JIT-компилятора является внутренней структурой данных, которая обычно важна для поддержки правильности и т.д. Например, код на промежуточном языке, как правило, является системой на основе стека, в которой JIT-компилятор работает над объектами в стеке по порядку и помещает результаты обратно в стек по порядку. JIT-компилятор в соответствии с реализациями настоящего изобретения соответственно конфигурируется для имитации стека, чтобы обеспечить ожидаемую работу JIT-компилятора. Например, JIT-компилятор может выполнять имитацию стека при компиляции кода на промежуточном языке. Если имитируемый стек значительно отклоняется от того, что подается в JIT-компилятор, то JIT-компилятор может завершить компиляцию или сформировать ошибку. Это помогает JIT-компилятору гарантировать то, что он работает в предписанных границах и соответственно защищен от нарушения одного или более правил безопасности.

Фиг. 1A иллюстрирует обзорную принципиальную схему компьютеризированной системы 100 (например, операционной системы видеоигры), в которой выполняется прикладная программа (то есть 110). В одной реализации прикладная программа 110 является приложением видеоигр, однако примем во внимание, что эта прикладная программа 110 может быть любым типом исполняемого программного кода. В любом случае фиг. 1A также показывает, что прикладная программа 110 содержит одно или более множеств исполняемых команд, таких как скомпилированный код 135, который включает в себя указатель 140 на код 120 на промежуточном языке ("IL").

Аналогичным образом, фиг. 1A показывает, что прикладная программа 110 содержит скомпилированный код 145, который включает в себя указатель 150 на код 125 на промежуточном языке. Код 125 на промежуточном языке, в свою очередь, содержит несколько разных компонентов или модулей, таких как код 120, 125 и 130, которые нуждаются в дополнительной компиляции перед тем, как они могут быть выполнены.

Имеется любое количество разных способов, которыми прикладная программа 110 будет или может выполняться на компьютерной системе 100. Например, пользователь может загрузить устройство хранения в другое устройство, на котором установлена система 100. Устройство хранения может включать в себя двоичный исполняемый код для прикладной программы 110, а также управляемый код в виде кода 115 на промежуточном языке. Как исполняемый код, так и код на промежуточном языке в прикладной программе 110 затем могли бы быть загружены в компьютеризированную систему 100. В иных случаях пользователь, например, разработчик, может загрузить прикладную программу 110, включая код 115 на промежуточном языке, посредством сетевого соединения. В таком случае пользователь мог бы выполнять прикладную программу 110 для тестирования недавно разработанных прикладных программ (например, 110).

В любом случае фиг. 1A также иллюстрирует, что прикладная программа 110 выполняется в режиме с меньшими привилегиями (например, "пользовательском" режиме), тогда как JIT-компилятор 105 работает в режиме с большими привилегиями (например, режиме "ядра"). Например, фиг. 1A показывает, что прикладная программа 110 работает в режиме пользователя 113 с пользовательскими привилегиями, тогда как JIT-компилятор 105 работает в режиме ядра 103 с соответствующими привилегиями ядра. К тому же, фиг. 1A показывает, что к коду 115 на промежуточном языке обращается один или более компонентов с привилегиями уровня ядра 103. Наоборот, и как будет в полной мере понятно в этом документе, исполняемый код будет выполняться только с помощью компонентов, работающих с уровнями привилегий пользователя 113.

Соответственно, так как рабочий цикл для прикладной программы 110 выполняет каждую из скомпилированных команд 135, 145 в режиме пользователя 113, рабочий цикл столкнется с любым из одного или более указателей на код на промежуточном языке. Например, во время выполнения, рабочий цикл для прикладной программы 110 встретит указатель 140 на код 120 на промежуточном языке. Поскольку указатель 140 ссылается на код, к которому можно обратиться только в режиме ядра 103, рабочий цикл выйдет из режима пользователя, и система 100 переключится на режим ядра 103.

Затем будет обрабатываться запрос 143 с помощью компонента 155 безопасности, который работает в режиме ядра 103. Вообще, компонент 155 безопасности может содержать любое число или тип компонентов или модулей, сконфигурированных для получения запроса компонентов в режиме пользователя 113 (например, 143) и последующей проверки, является ли этот запрос подходящим. Это выполняется, потому что режим пользователя 113 является недоверяемым, и потому что прикладная программа 110 может представлять собой или не представлять (или иным образом включать в себя) опасный или вредоносный код.

Таким образом, для гарантии того, что запросы из выполнения в режиме 113 пользователя не повредят систему 100, компонент 155 безопасности может выполнять любое количество или любой тип функций проверки достоверности. Например, компонент 155 безопасности может проанализировать сообщение 143 на любое количество дескрипторов, маркеров или т.п. Кроме того, компонент 155 безопасности

может проанализировать запрос 143 на команды приложения, которые могли бы использоваться для компрометации системы 100, например, запросы к особым адресам в памяти или запросы, которые могли бы привести к переполнению буфера, и т.д. После проверки достоверности запроса 143 компонент 155 безопасности может запустить JIT-компилятор 105 в режиме ядра.

При работе в режиме ядра JIT-компилятору может подаваться запрошенный код (то есть 120), и он может начинать компиляцию. Например, фиг. 1A показывает, что компонент 155 безопасности выполняет один или более запросов 147, которые заставляют JIT-компилятор 105 получить и компилировать код 120 на промежуточном языке. После компиляции кода 120 в исполняемые двоичные команды (то есть скомпилированный код 123), фиг. 1A также показывает, что JIT-компилятор 105 может затем передать код 123 в динамически распределяемую память 160.

Как будет в полной мере понятно по отношению к фиг. 1B, динамически распределяемая память 160 охватывает границу между операциями режима пользователя 113 и режима ядра 103. На самом деле, динамически распределяемая память 160 действует как межразрешительное/межграничное хранилище, которое доступно компонентам, работающим в режиме ядра 103 и/или режиме 113 пользователя. Как только завершена компиляция, система 100 может переключиться обратно в режим пользователя и продолжить выполнение прикладной программы 110. В частности, приложение 110 - работающее в режиме пользователя - может извлечь скомпилированный код 123, как только он будет доступен, и начать его выполнение в режиме 113 пользователя. Поэтому примем во внимание, что динамически распределяемая память 160 может использоваться для помощи в обеспечении границ безопасности между двумя уровнями безопасности путем разрешения JIT-компилятору 105 и пользователю 113 функционировать независимо, в разных режимах привилегий без прямого взаимодействия.

Фиг. 1B иллюстрирует дополнительные подробности о том, как может достигаться или иным образом поддерживаться граница безопасности между JIT-компилятором 105 и прикладной программой 110. В частности, фиг. 1B иллюстрирует реализацию, в которой JIT-компилятор 105 и прикладная программа 110 работают относительно конкретной одинаковой ячейки памяти, хотя и с разными наборами разрешений. В частности, фиг. 1B иллюстрирует реализацию, в которой к одной и той же ячейке памяти можно обращаться с помощью компонентов в одном адресном пространстве с одним набором разрешений в одном адресном пространстве, и обращаться с помощью разных компонентов в другом адресном пространстве с другим набором разрешений. Например, фиг. 1B показывает, что ячейка 160 памяти доступна в адресном пространстве 170 с разрешениями чтения/записи, и в адресном пространстве 175 с разрешениями чтения/выполнения.

Вообще, один или более компонентов уровня ядра 103 в операционной системе 100 будут поддерживать таблицу 180 страниц памяти для любого заданного адресного расположения и соответствующих адресных пространств. Например, фиг. 1B показывает, что таблица 180 страниц памяти поддерживается на уровне ядра 103 (то есть одного или более компонентов в режиме ядра) в системе 100. Одной из причин того, что это поддерживается компонентом в режиме ядра 103, является гарантия, что недоверяемая прикладная программа (то есть работающая в режиме пользователя) не может обращаться или иным образом неправильно манипулировать таблицей страниц.

В любом случае фиг. 1B показывает, что таблица 180 страниц соотносит ячейки 160 и 165 памяти с адресными пространствами 170, 175, 190 и 195. Например, ячейка 160

памяти является совместно используемой динамически распределяемой памятью, тогда как ячейка 165 памяти является расположением, в которое загружается для выполнения прикладная программа 110. К тому же, таблица 180 страниц устанавливает соответствие разрешений доступа у ячеек 160 и 165 памяти, так что адресные пространства 170 и 190 обладают доступом "чтения/записи" к ячейкам 160 или 165 соответственно. Аналогичным образом, таблица 180 страниц устанавливает соответствие разрешений у ячеек 160 и 165 памяти для адресных пространств 175 или 195 в виде "чтения/выполнения" соответственно. Соответственно, когда компонент 155 безопасности (фиг. 1A) получает запрос (например, 143) от компонента в режиме пользователя 113, компонент 155 безопасности может сопоставить адресные пространства у компонента, создающего запрос (например, 143), с адресным пространством для выходных данных JIT-компилятора (например, 123).

Как упоминалось ранее, одним из способов, которыми система 100 может усилить границы уровня безопасности и разрешения, является динамически распределяемая память 160, которая охватывает описываемые границы безопасности/разрешения. Вообще, "динамически распределяемая память" содержит множество адресов памяти, заданных отдельно системой 100 во время или непосредственно перед рабочим циклом. В этом конкретном примере система 100 может выделить и сконфигурировать динамически распределяемую память 160, чтобы только компоненты уровня ядра (например, JIT-компилятор 105) могли записывать в динамически распределяемую память 160 (например, посредством таблицы 180 страниц), тогда как компоненты уровня пользователя могли только считывать из динамически распределяемой памяти 160. В результате прикладная программа 110 не может выполнить никакой скомпилированный код от JIT-компилятора 105 в динамически распределяемой памяти 160, а вместо этого должна делать это только в адресном пространстве 175.

Поэтому примем во внимание, что "песочница" может быть задана путем требования работы приложения только в режиме пользователя и путем требования, чтобы приложение и JIT-компилятор обращались к определенным компонентам или структурам данных из адреса памяти, ассоциированного с разными наборами разрешений. Соответственно, фиг. 1A-1B и соответствующий текст иллюстрируют некоторое количество разных архитектурных компонентов, которые могут использоваться для доступа и/или выполнения практически любого типа исполняемого кода, включая управляемый код, безопасным способом. В частности, фиг. 1A-1B и соответствующий текст иллюстрируют, как приложение может выполнить в режиме пользователя 113 и обратиться к динамически распределяемой памяти только с разрешениями чтения или чтения/выполнения для JIT-компилированного кода. К тому же чертежи и соответствующий текст иллюстрируют, как приложение может вызвать один или более компонентов уровня ядра в другом адресном пространстве 170, которое обладает разрешениями чтения/записи на динамически распределяемую память 160, и может соответственно скомпилировать и передать управляемый код в динамически распределяемую память 160, но не выполнить его.

Как упоминалось ранее, этот тип распределенной конфигурации адресного пространства может обеспечить некоторое количество различных преимуществ для выполнения и разработки программ. Вначале, например, разработчик прикладной программы может написать практически любой тип кода, не беспокоясь о соображениях безопасности (например, типовой безопасности). Кроме того, разработчику операционной системы не нужны тратящие скорость ресурсы при

разработке кода проверки в рабочем цикле, который заставит весь выполняющийся программный код стать безопасным (например, безопасным по типу).

В дополнение к вышеупомянутому реализации настоящего изобретения также могут описываться на основе блок-схем алгоритмов, имеющих одно или более действий в способе для достижения конкретного результата. В частности, фиг. 2 и 3 и соответствующий текст иллюстрируют блок-схемы алгоритмов одного или более действий для выполнения управляемого кода, чтобы безопасный и небезопасный код прикладной программы мог выполняться без угрозы или подвергания риску безопасности. Способы, проиллюстрированные на фиг. 2 и 3, описываются ниже со ссылкой на компоненты и схемы из фиг. 1A-1B.

Соответственно, фиг. 2 показывает, что способ из перспективной клиентской компьютерной системы может содержать действие 200 выполнения приложения в первом адресном пространстве. Действие 200 включает в себя выполнение прикладной программы в первом адресном пространстве ячейки памяти. Например, фиг. 1B показывает, что прикладная программа 110 выполняется из адресного пространства 175, которое обладает разрешениями чтения/выполнения для доступа к ячейке 160 памяти (то есть, где JIT-скомпилированный код будет размещаться и соответственно обозначаться как чтение/выполнение).

Фиг. 2 также показывает, что способ может содержать действие 210 получения запроса от приложения на команды на промежуточном языке. Действие 210 может включать в себя получение одного или более запросов от прикладной программы на компиляцию одного или более множеств команд на промежуточном языке. Например, рабочий цикл для прикладной программы 110 встречает указатель 140 на код 120 на промежуточном языке, к которому можно обращаться только в режиме ядра 103. По существу, рабочий цикл передает указатель 120 в виде сообщения 143 компоненту 155 безопасности, который обрабатывает запрос в режиме ядра.

К тому же фиг. 2 показывает, что способ может содержать действие 220 компиляции команд на промежуточном языке во втором адресном пространстве. Действие 220 включает в себя компиляцию одного или более множеств команд на промежуточном языке в заново скомпилированный код с использованием JIT-компилятора, работающего во втором адресном пространстве. Например, после проверки достоверности запроса 143 компонент 155 безопасности готовит и выполняет один или более запросов 147 для передачи запрошенного кода на промежуточном языке в JIT-компилятор 105. JIT-компилятор 105 затем компилирует код 120 на промежуточном языке во втором адресном пространстве 170, которое в этой иллюстрации снабжается разрешениями чтения/записи на совместно используемую динамически распределяемую память 160.

Кроме того, фиг. 2 показывает, что способ может содержать действие 230 передачи скомпилированного кода в совместно используемую динамически распределяемую память. Действие 230 включает в себя передачу заново скомпилированного кода в совместно используемую динамически распределяемую память, в которой прикладная программа может извлечь заново скомпилированный код в первое адресное пространство. Например, фиг. 1A и 1B показывают, что JIT-компилятор 105, а также прикладная программа 110, имеют доступ к динамически распределяемой памяти 160. В частности, JIT-компилятор 105 может записывать (но не выполнять) в динамически распределяемую память 160, тогда как прикладная программа 110 может только считывать и выполнять из динамически распределяемой памяти 160. Таким образом, когда JIT-компилятор 105 компилирует и создает код 123, рабочий цикл для

прикладной программы 110 может извлечь скомпилированный код 123 в адресное пространство 175 и выполнить код в режиме пользователя.

В дополнение к вышеупомянутому, фиг. 3 показывает, что способ в соответствии с реализацией настоящего изобретения, состоящий из формирования выполняемого компьютером программного кода для компьютерной системы способом, который использует JIT-компиляцию наряду с предотвращением нарушений безопасности, может содержать действие 300 получения исполняемого кода и кода, который необходимо компилировать. Действие 300 включает в себя получение программного кода, который включает в себя исполняемый код и код, который необходимо компилировать. Например, операционная система 100 получает один или более носителей информации и/или получает загрузку по сети прикладной программы 110. Прикладная программа 110 включает в себя исполняемый программный код, а также код 115 на промежуточном языке, к которому обращаются отдельно с помощью одного или более компонентов уровня ядра 103.

Фиг. 3 также показывает, что способ может содержать действие 310 выполнения исполняемого кода в режиме с меньшими привилегиями. Действие 310 включает в себя выполнение исполняемого кода в режиме с меньшими привилегиями и в первом адресном пространстве. Например, фиг. 1A показывает, что к исполняемой части прикладной программы 110 обращаются или иным образом выполняют только в режиме пользователя 113, тогда как к коду 115 на промежуточном языке обращаются только компоненты в режиме ядра.

К тому же, фиг. 3 показывает, что способ может содержать действие 310 получения указателя на код, который необходимо компилировать. Действие 310 включает в себя получение одного или более указателей в исполняемом коде по меньшей мере на некоторый код, который необходимо компилировать. Например, фиг. 1A-1B показывают прикладную программу 110, которая работает в режиме пользователя 113 и в/из адресного пространства 175, содержит скомпилированный код 135, указатель 140 на код 120 на промежуточном языке, скомпилированный код 145 и указатель 150 на код 125 на промежуточном языке. Несмотря на выполнение прикладной программы 110 в режиме пользователя, указатели 140 и/или 150 будут идентифицироваться по очереди.

Кроме того, фиг. 3 показывает, что способ может содержать действие 330 переключения в более привилегированный режим. Например, рабочий цикл для прикладной программы 110 идентифицирует указатель 140 во время выполнения и идентифицирует, что нужно будет запустить JIT-компилятор 105. Поскольку JIT-компилятору 105 необходимо работать в режиме ядра, система 100 мгновенно останавливает выполнение приложения 110, переключается из режима пользователя в режим ядра и затем запускает JIT-компилятор 105 в качестве компонента режима ядра 103. Сообщение 143, которое включает в себя указатель 140, передается затем компоненту 155 безопасности в режиме ядра 103. Компонент 155 безопасности, работающий в режиме ядра, затем оценивает запрос, чтобы удостовериться в том, что запрос 143 сформирован должным образом и/или включает в себя подходящие дескрипторы, идентификаторы безопасности и т.д.

Более того, фиг. 3 показывает, что способ может содержать действие 340 компиляции запрошенного кода в режиме с большими привилегиями. Действие 340 включает в себя компиляцию запрошенного кода в другом адресном пространстве, используя компилятор, работающий в режиме с большими привилегиями. Например, фиг. 1A и 1B показывают, что JIT-компилятор 105, который работает на уровне

ядра 103 с большими привилегиями, может компилировать код 120 в одном адресном пространстве (адресном пространстве 170) и затем передать скомпилированный код 123 в динамически распределяемую память 160, где JIT-компилятор обладает доступом чтения/записи. При переключении обратно в режим пользователя прикладная программа 110 может затем обратиться к скомпилированному коду 123 и выполнить этот код из другого адресного пространства (адресного пространства 175), которое обладает разрешениями чтения/выполнения на динамически распределяемую память 160.

По существу, фиг. 1A-2 и соответствующий текст предоставляют некоторое количество компонентов, модулей и механизмов, которые могут использоваться для выполнения недоверяемого кода, включая управляемый код, без принесения в жертву важных гарантий безопасности. Как описывалось ранее, это может быть по меньшей мере частично достигнуто путем разделения компиляции кода на промежуточном языке и выполнения двоичного кода в отдельных адресных пространствах для одной и той же программы. К тому же, это может быть достигнуто с помощью JIT-компилятора с типовой безопасностью, который компилирует промежуточный код и передает скомпилированный код в совместно используемую динамически распределяемую память. JIT-компилятор с типовой безопасностью конфигурируется таким образом, что несмотря на то, что он может принимать и компилировать код, который не обладает типовой безопасностью, сам JIT-компилятор ограничивается от работы вне определенных предписанных границ типовой безопасности. Более того, это может быть достигнуто путем гарантии, что к исполняемому коду обращаются только компоненты, работающие в режиме пользователя, и что к коду на промежуточном языке обращаются только компоненты, работающие в режиме ядра в адресном пространстве чтения/записи.

Варианты осуществления настоящего изобретения могут содержать специализированный или универсальный компьютер, включающий в себя различные аппаратные средства, которые обсуждаются подробнее ниже. Варианты осуществления в рамках объема настоящего изобретения также включают в себя компьютерночитаемые носители для передачи или хранения на них выполняемых компьютером команд или структур данных. Такие компьютерночитаемые носители могут быть любыми доступными носителями, к которым можно обращаться посредством универсального или специализированного компьютера.

В качестве примера, а не ограничения, такие компьютерночитаемые носители могут содержать RAM, ROM, EEPROM, компакт-диск или другой накопитель на оптических дисках, накопитель на магнитных дисках или другие магнитные устройства хранения, либо любой другой носитель, который может использоваться для перемещения или хранения необходимого средства программного кода в виде выполняемых компьютером команд или структур данных, и к которому можно обращаться посредством универсального или специализированного компьютера. Когда информация предается или предоставляется компьютеру по сети или другому соединению связи (одному из проводного, беспроводного или сочетания проводного или беспроводного), компьютер по существу рассматривает соединение как компьютерночитаемый носитель. Таким образом, любое такое соединение корректно называть компьютерночитаемым носителем. Сочетания вышеперечисленного также следует включить в область компьютерночитаемых носителей.

Выполняемые компьютером команды содержат, например, команды и данные, которые заставляют универсальный компьютер, специализированный компьютер или

специализированное устройство обработки выполнять определенную функцию или группу функций. Несмотря на то, что предмет изобретения описан на языке, характерном для структурных особенностей и/или методологических действий, необходимо понимать, что предмет изобретения, определенный в прилагаемой формуле, не обязательно ограничивается описанными выше конкретными особенностями или действиями. Скорее, описанные выше конкретные признаки и действия раскрываются в качестве примеров реализации формулы изобретения.

Настоящее изобретение может быть реализовано в других конкретных видах без отклонения от его сущности или неотъемлемых характеристик. Описанные варианты осуществления должны рассматриваться во всех отношениях только как пояснительные, а не ограничительные. Следовательно, объем изобретения указывается прилагаемой формулой изобретения, а не предшествующим описанием. Все изменения, которые подпадают под смысл и диапазон эквивалентности формулы изобретения, должны включаться в ее объем.

Формула изобретения

1. В компьютеризированной среде, содержащей память, а также JIT-компилятор и одну или более прикладных программ, загруженных в память, способ выполнения управляемого кода, чтобы недоверяемый программный код мог компилироваться и выполняться способом, который не представляет опасности или иным образом подвергает риску безопасность системы, содержащий этапы, на которых:

выполняют прикладную программу из первого адресного пространства, заданного с первым набором разрешений для доступа к совместно используемой динамически распределяемой памяти;

получают один или более запросов от прикладной программы на компиляцию одного или более множеств команд на промежуточном языке; компилируют одно или более множеств команд на промежуточном языке в заново скомпилированный код с использованием JIT-компилятора, работающего во втором адресном пространстве, которое обладает вторым набором разрешений для доступа к совместно используемой динамически распределяемой памяти; и

передают заново скомпилированный код в совместно используемую динамически распределяемую память, в которой прикладная программа может извлечь и выполнить заново скомпилированный код из первого адресного пространства.

2. Способ по п.1, дополнительно содержащий действие, в котором, при получении указания, что заново скомпилированный код передан в совместно используемую динамически распределяемую память, переключают уровень операций из режима ядра в режим пользователя.

3. Способ по п.2, дополнительно содержащий действие, в котором прикладная программа извлекает скомпилированный код и выполняет скомпилированный код из первого адресного пространства.

4. Способ по п.1, в котором первое адресное пространство конфигурируется с разрешениями чтения/выполнения относительно доступа к совместно используемой динамически распределяемой памяти, так что никакой компонент, работающий в первом адресном пространстве, не может записывать в совместно используемую динамически распределяемую память.

5. Способ по п.1, в котором второе адресное пространство конфигурируется для доступа к динамически распределяемой памяти с разрешениями чтения/записи, так что никакой компонент, работающий во втором адресном пространстве, не может

выполнять код в динамически распределяемой памяти.

6. Способ по п.1, в котором JIT-компилятор работает в режиме с большими привилегиями, а прикладная программа работает в режиме с меньшими привилегиями.

7. Способ по п.1, в котором JIT-компилятор ограничивается выполнением в рамках одного или более ограничений типовой безопасности, но конфигурируется для получения и компиляции кода на промежуточном языке, который не обладает типовой безопасностью.

8. Способ по п.7, в котором JIT-компилятор выполняет действия, в которых:

получают один или более запросов на выполнение функции, которая нарушает ограничение безопасности для JIT-компилятора; и

отклоняют один или более запросов на выполнение функции или прекращают компиляцию одного или более множеств команд на промежуточном языке.

9. Способ по п.1, дополнительно содержащий действие, в котором, при получении одного или более запросов от прикладной программы, активизируют уровень операций режима ядра.

10. Способ по п.9, в котором действие, при котором активизируют уровень операций режима ядра, включает в себя действие, при котором инициируют компонент безопасности в режиме ядра.

11. Способ по п.10, в котором один или более запросов от прикладной программы принимаются компонентом безопасности в режиме ядра.

12. Способ по п.11, дополнительно содержащий действие компонента безопасности в режиме ядра, при котором проверяют достоверность одного или более запросов от прикладной программы.

13. Способ по п.12, в котором действие, при котором проверяют достоверность одного или более запросов, содержит действие, при котором определяют, допустим ли дескриптор, включенный в один или более запросов.

14. В компьютеризированной среде, содержащей память, JIT-компилятор и одну или более прикладных программ, загруженных в память, способ формирования исполняемого компьютером программного кода способом, который использует JIT-компиляцию наряду с предотвращением нарушений безопасности, содержащий этапы, на которых:

получают код прикладной программы, который включает в себя исполняемый код и код, который необходимо компилировать;

выполняют исполняемый код в режиме с меньшими привилегиями и в первом адресном пространстве;

идентифицируют один или более указателей в исполняемом коде по меньшей мере на некоторый код, который необходимо компилировать;

переключаются в режим с большими привилегиями; и

компилируют по меньшей мере некоторый код в другом адресном пространстве, используя компилятор, работающий в режиме с большими привилегиями.

15. Способ по п.14, в котором код прикладной программы содержит часть приложения видеоигр, которое принимается из хранения в операционную систему видеоигр.

16. Способ по п.14, в котором компилятор является JIT-компилятором с типовой безопасностью, сконфигурированным для обработки только запросов с типовой безопасностью, но иным образом сконфигурированным для компиляции кода на промежуточном языке с типовой безопасностью или без типовой безопасности.

17. Способ по п.14, в котором режим с большими привилегиями является уровнем

операций режима ядра, и режим с меньшими привилегиями является пользовательским уровнем операций.

18. Способ по п.14, в котором первое адресное пространство конфигурируется для доступа к динамически распределяемой памяти с разрешениями чтения/выполнения, а второе адресное пространство конфигурируется для доступа к динамически распределяемой памяти с разрешениями чтения/записи.

19. Способ по п.14, дополнительно содержащий действия, в которых:

переключаются в режим с меньшими привилегиями после определения, что по меньшей мере некоторый код скомпилирован; и

выполняют скомпилированный по меньшей мере некоторый код в первом адресном пространстве.

20. В компьютеризированной среде, содержащей память, JIT-компилятор и одну или более прикладных программ, загруженных в память, изделие хранения компьютерной программы, имеющее записанные на нем исполняемые компьютером команды, которые при выполнении заставляют один или более процессоров выполнять способ, содержащий:

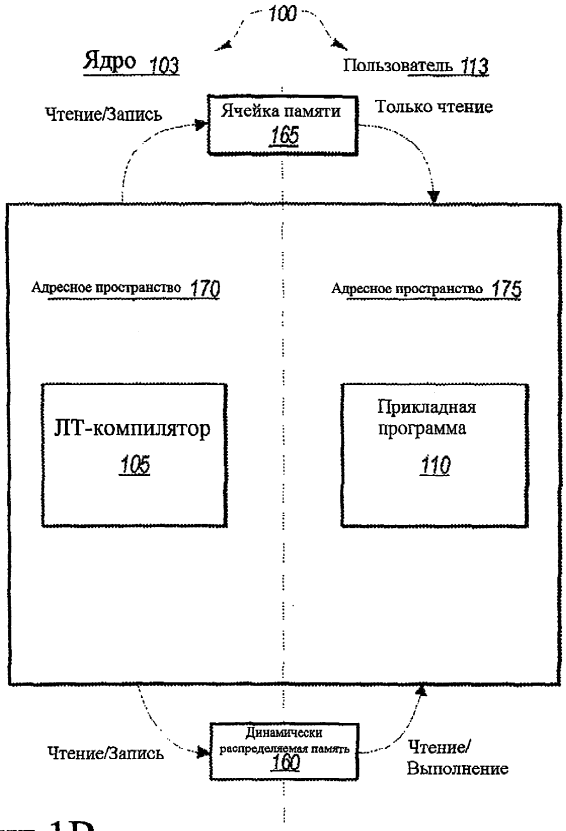
выполнение прикладной программы из первого адресного пространства, заданного с первым набором разрешений для доступа к совместно используемой динамически распределяемой памяти;

получение одного или более запросов от прикладной программы на компиляцию одного или более множеств команд на промежуточном языке;

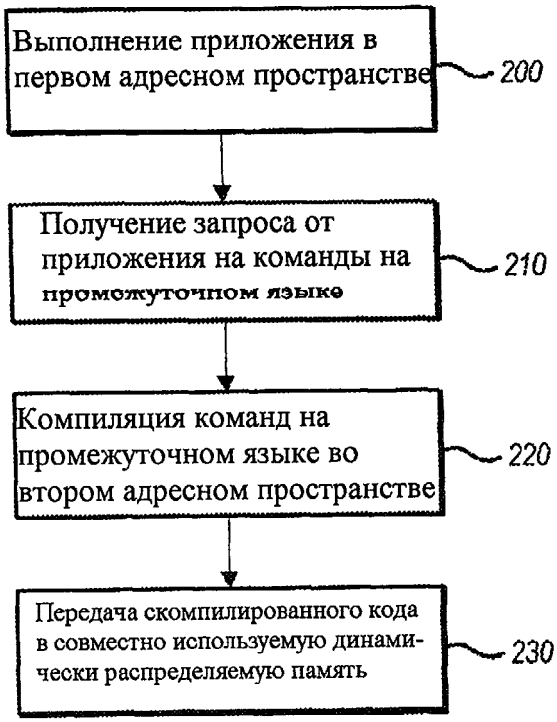
компиляцию одного или более множеств команд на промежуточном языке в заново скомпилированный код с использованием JIT-компилятора, работающего во втором адресном пространстве, которое обладает вторым набором разрешений для доступа к совместно используемой динамически распределяемой памяти; и

передачу заново скомпилированного кода в совместно используемую динамически распределяемую память, в которой прикладная программа может извлечь и выполнить заново скомпилированный код из первого адресного пространства.

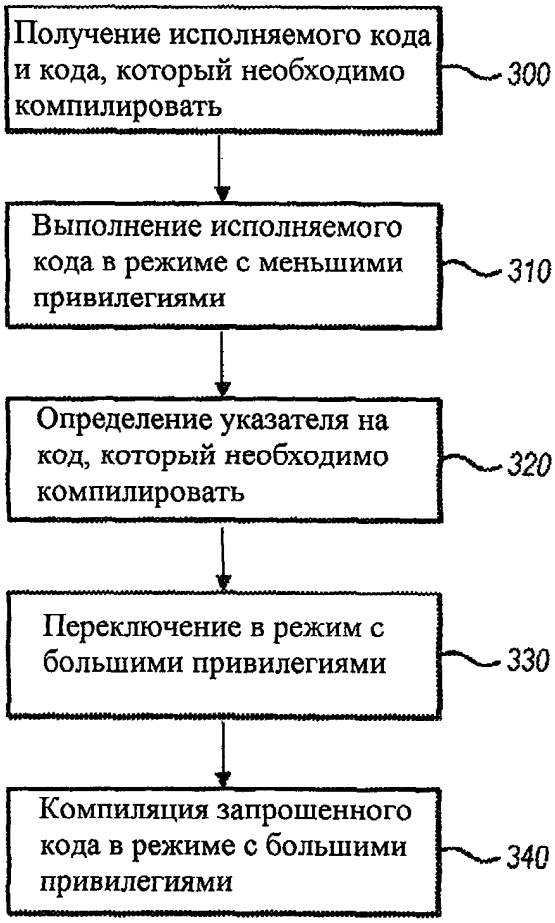
Таблица страниц 180				
Ячейка памяти	Адресное пространство 170	Адресное пространство 175	Адресное пространство 190	Адресное пространство 195
160	Чтение/Запись	Чтение/ Выполнение	Нет доступа	Нет доступа
165	Чтение/Запись	Только чтение	Чтение/Запись	Чтение/ Выполнение



Фиг.1В



Фиг.2


Получение исполняемого кода и кода, который необходимо компилировать
300
Выполнение исполняемого кода в режиме с меньшими привилегиями
310
Определение указателя на код, который необходимо компилировать
320
Переключение в режим с большими привилегиями
330
Компиляция запрошенного кода в режиме с большими привилегиями
340

Фиг.3