РОССИЙСКАЯ ФЕДЕРАЦИЯ

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(19) **RU** (11) **2 757 409**(13) **C1**

(51) МПК
G06F 21/56 (2013.01)
G06F 9/455 (2006.01)

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(52) СПК
G06F 9/455 (2021.02); G06F 21/56 (2021.02)

(21)(22) Заявка: **2020120434, 19.06.2020**

(24) Дата начала отсчета срока действия патента:
19.06.2020

Дата регистрации:
15.10.2021

Приоритет(ы):
(22) Дата подачи заявки: **19.06.2020**

(45) Опубликовано: **15.10.2021** Бюл. № **29**

Адрес для переписки:
125212, Москва, Ленинградское ш., 39а, стр. 3, АО "Лаборатория Касперского", Управление по интеллектуальной собственности, Московский Дмитрий Валерьевич

(72) Автор(ы):
Пинтийский Владислав Валерьевич (RU),
Аникин Денис Вячеславович (RU),
Кирсанов Дмитрий Александрович (RU),
Трофименко Сергей Владимирович (RU)

(73) Патентообладатель(и):
Акционерное общество "Лаборатория Касперского" (RU)

(56) Список документов, цитированных в отчете о поиске: **US 20110225655 A1, 15.09.2011. US 7603713 B1, 13.10.2009. CN 1242327 C, 15.02.2006. RU 2622627 C2, 16.06.2017. RU 2472215 C1, 10.01.2013.**

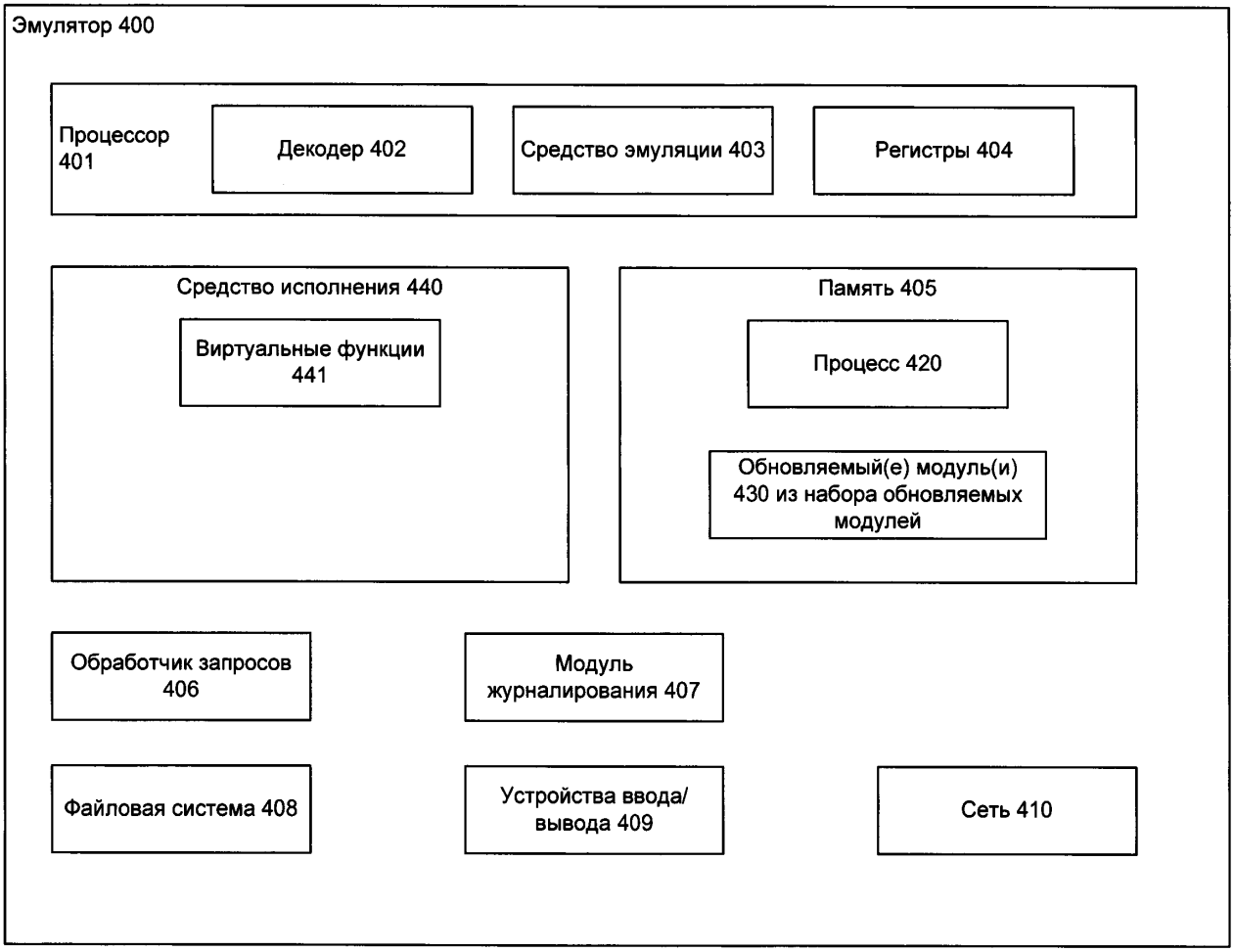
(54) Эмулятор и способ эмуляции

(57) Реферат:

Группа изобретений относится к области информационной безопасности. Техническим результатом является повышение достоверности эмуляции инструкций файла, повышение уровня обнаружения вредоносного кода, снижение времени реакции на новые угрозы. Устройство содержит средство эмуляции, предназначенное для эмуляции исполнения инструкций файла на виртуальном процессоре эмулятора; приостановления эмуляции исполнения инструкций на вызове API-функции; проверки наличия API-функции в наборе обновляемых модулей; эмуляции исполнения упомянутой API-функции по инструкциям согласно реализации из соответствующего обновляемого модуля, если API-функция найдена; передачи управления средству исполнения, если API-функция не найдена в наборе обновляемых модулей; продолжения эмуляции исполнения инструкций файла с инструкции по адресу возврата API-функции, используя результат исполнения API-функции; содержащийся в памяти набор обновляемых модулей из по меньшей мере одного обновляемого модуля, где каждый обновляемый модуль из упомянутого набора содержит реализацию по меньшей мере одной API-функции; средство исполнения, предназначенное для формирования результата исполнения упомянутой API-функции согласно реализации API-функции, содержащейся в средстве исполнения или созданной средством исполнения и последующей передачи средству эмуляции результата исполнения API-функции. 2 н. и 18 з.п. ф-лы, 6 ил.

RU 2757409 C1

RU 2757409 C1



Фиг. 4

RU 2757409 C1

RU 2757409 C1

RUSSIAN FEDERATION

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY

(19) **RU** (11) **2 757 409**(13) **C1**

(51) Int. Cl.
G06F 21/56 (2013.01)
G06F 9/455 (2006.01)

(12) ABSTRACT OF INVENTION

(52) CPC
G06F 9/455 (2021.02); G06F 21/56 (2021.02)

(21)(22) Application: **2020120434, 19.06.2020**

(24) Effective date for property rights:
19.06.2020

Registration date:
15.10.2021

Priority:
(22) Date of filing: **19.06.2020**

(45) Date of publication: **15.10.2021** Bull. № **29**

Mail address:
125212, Moskva, Leningradskoe sh., 39a, str. 3, AO "Laboratoriya Kasperskogo", Upravlenie po intellektualnoj sobstvennosti, Moskovskij Dmitrij Valerevich

(72) Inventor(s):
Pintijskij Vladislav Valerevich (RU),
Anikin Denis Vyacheslavovich (RU),
Kirsanov Dmitrij Aleksandrovich (RU),
Trofimenko Sergej Vladimirovich (RU)

(73) Proprietor(s):
Aktsionernoe obshchestvo "Laboratoriya Kasperskogo" (RU)

RU 2757409 C1

(54) EMULATOR AND METHOD FOR EMULATION

(57) Abstract:

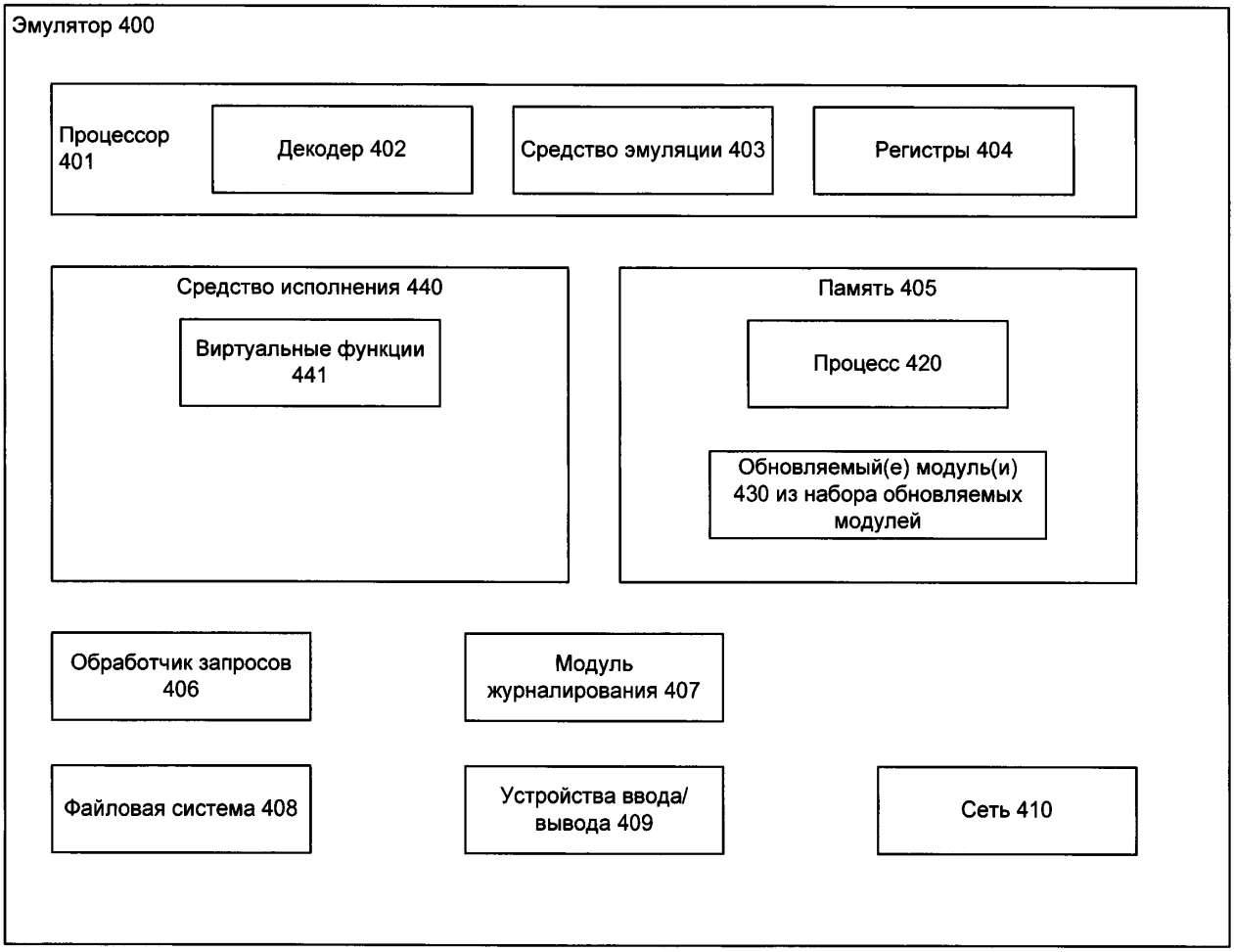
FIELD: information security.

SUBSTANCE: apparatus comprises an emulation tool intended for emulating execution of the instructions of a file on the virtual processor of the emulator; suspending the emulation of execution of the instructions on an API function call; confirming the presence of the API function in the set of updated modules; emulating the execution of said API function following the instructions according to the implementation from the corresponding updated module if the API function is found; transferring control to the execution tool if the API function is not found in the set of updated modules; continuing the emulation of execution of the instructions of the file from the instruction on the return address of the API function using the result of execution of the API function; a set of updated modules stored in the memory consisting of at least one updated module, wherein each updated module from said set comprises an implementation of at least one API function; an execution tool intended for forming the result of execution of said API function according to the implementation of the API function stored in the execution tool or created by the execution tool and further transferring the result of execution of the API function to the emulation tool.

EFFECT: increase in the accuracy of emulation of the instructions of a file, increase in the level of detection of malicious code, reduction in the time of response to new threats.

20 cl, 6 dwg

RU 2757409 C1



Фиг. 4

RU 2757409 C1

RU 2757409 C1

Область техники

Изобретение относится к области информационной безопасности, а более конкретно к эмуляторам и способам эмуляции.

Уровень техники

Код современных приложений, в том числе вредоносных, представляет собой сложный набор инструкций, содержащий переходы, вызовы, в том числе рекурсивные, циклы и т.д. Стоит отметить, что сложность исполняемых файлов постоянно увеличивается. Это связано с расширением функционала программного обеспечения (ПО), с использованием языков программирования высокого уровня, содержащих многочисленные библиотеки, а также с усложнением компьютерной техники и операционных систем (ОС). Это относится как к доверенным приложениям, так и к вредоносным. Вредоносные приложения могут совершать ряд характерных действий, таких как кража паролей и других конфиденциальных данных пользователя, включение компьютера в бот-сеть для проведения DDoS-атак или рассылки спама, блокирование корректного функционирования системы с целью вымогательства и другие отрицательные и нежелательные с точки зрения пользователя действия.

Одним из методов исследования потенциально вредоносного приложения является использование эмулятора, который применяется в антивирусном приложении при анализе поведения приложения. Существуют различные способы эмуляции. Одним из них является программно-аппаратная имитация аппаратных компонентов компьютера и различных структур этих компонентов - процессора, памяти и других устройств путем создания виртуальных копий регистров процессора, памяти и набора инструкций процессора. Кроме того, возможен гибридный подход к эмуляции, включающий аппаратную и программную эмуляции. В этом подходе происходит дополнительная имитация операционной системы компьютера путем создания виртуальных копий компонентов ОС, в рамках которых происходит эмуляция исполнения приложения. Такими компонентами являются, например, часть ядра ОС, отвечающая за необходимые механизмы ее работы, такие как подсистемы ввода-вывода, файловая система, механизмы создания и инициализации процессов, обработка прерываний и исключений, вызовов системных API-функций (англ. application programming interface, API), а также драйверы устройств, управление памятью и т.д. Таким образом, в эмуляторе инструкции приложения исполняются не на реальном процессоре, а на его виртуальной копии, а вызовы системных API-функций эмулируют и отправляют в ответ проэмулированный результат работы функции.

Во время эмуляции формируют журнал вызовов функций, к которому также имеет доступ антивирус. В журнале обычно сохраняют данные о вызовах API-функций, произведенных упомянутым процессом во время эмуляции исполнения, а также данные о возвратах из вызванных API-функций (передача управления по адресу возврата). Полнота такого журнала влияет, в частности, на качество обнаружения вредоносных файлов. Например, снижаются ложноположительные и ложноотрицательные ошибки, и соответственно повышается уровень обнаружения (от англ. detection rate). Для этого антивирус проверяет журнал на соответствие сигнатурам антивирусных баз - известных последовательностей вызовов функций, характерных для вредоносных файлов. Таким образом, в случае нахождения одной из сигнатур в журнале вывозов функций, антивирус определяет проверяемый файл вредоносным.

Многие современные вредоносные приложения используют различные антиэмуляционные приемы (англ. anti-emulation tricks), позволяющие определить факт исполнения приложения не на реальном компьютере, а в эмуляторе. В этом случае

вредоносное приложение не будет выполнять вредоносные действия в эмуляторе, так как в начале его исполнения сработают эти приемы и вредоносный функционал не будет вызван. Вследствие этого антивирусное приложение посчитает вредоносное приложение доверенным и разрешит его исполнение на реальном компьютере либо посчитает «серым» приложением, то есть требующим разрешения от пользователя на исполнение своего функционала, в зависимости от настроек антивирусного приложения. Антиэмуляционные приемы могут включать проверку наличия запущенных служб, установленных системных переменных и других элементов, присутствующих в реальной ОС. Кроме того, вредоносное приложение может проверять корректность реализации вызовов API-функций или выполнять обращение к редким API-функциям, реализация которых может отсутствовать в эмуляторе.

Поэтому, для улучшения качества обнаружения вредоносных файлов и обхода антиэмуляционных приемов антивирусный эмулятор должен максимально полно имитировать аппаратные и программные компоненты компьютера. Однако, полностью имитировать ОС и аппаратные компоненты практически невозможно, так как необходимо фактически переписать код всей ОС, что отразится на скорости выполнения задач. Многие вредоносные приложения проверяют время своего исполнения и, если время превышает ожидаемое значение, вредоносное приложение обнаруживает, что оно исполняется не в реальной среде, а в эмуляторе. Более того, чем полнее реализация эмулятора, тем больше ресурсов компьютера необходимо для эмуляции. Поэтому в эмуляторах обычно реализуют имитацию наиболее важных аппаратных и программных компонентов, а также наиболее часто используемых вызовов API-функций.

Вызовы редко используемых API-функций могут быть реализованы в виде так называемых «заглушек» (англ. stub - функция, не выполняющая никакого осмысленного действия, возвращающая пустой результат или входные данные в неизменном виде). Либо их реализация генерируется эмулятором на лету, возвращая сгенерированный результат. Сгенерированный результат может быть произвольным либо зависеть от типа функции - например, в ответ на вызов функции работы с файлом будет возвращен произвольный файловый описатель. Такой подход позволяет достичь оптимального баланса между производительностью и качеством обнаружения, но он также имеет очевидные недостатки, связанные с возможностью использования антиэмуляционных приемов вредоносными приложениями. Поэтому антивирусные компании постоянно совершенствуют и обновляют код эмулятора, включая реализацию новых вызовов API-функций, а также исправляя уже реализованные вызовы API-функций. Например, если было обнаружено новое вредоносное приложение, вызывающее редкую API-функцию, для которой в эмуляторе была лишь заглушка, будет написана более полная реализация этой API-функции и обновлен код эмулятора на компьютерах пользователей. Однако, проблема заключается в том, что для обновления реализации функций требуется выпустить обновление кода всего эмулятора. При этом процедура выпуска обновления достаточно сложная и занимает много времени. К тому же до обновления эмулятора качество эмуляции будет недостаточным для обнаружения новых вредоносных приложений.

Поэтому возникает техническая проблема, заключающаяся в недостаточном качестве эмуляции существующим функционалом для обнаружения новых вредоносных файлов.

Один из методов борьбы с антиэмуляционными приемами описан в патенте US 7603713, способ работы которого включает исполнение ряда инструкций на реальном процессоре, что существенно ускоряет процесс эмуляции неизвестных приложений для того, чтобы можно было добраться до самого вредоносного кода. Другой подход,

раскрытый в заявке US 20110225655, рассматривает вариант определения приложения как подозрительного, если оно противодействует эмуляции. Подобный подход основан на том, что вредоносное приложение или сразу завершает исполнение, или меняет вектор (логику) исполнения.

Однако описанные подходы не решают указанную техническую проблему, т.к. качество эмуляции в этих подходах снижается при использовании новых антиэмуляционных приемов до обновления кода эмулятора.

Анализ предшествующего уровня техники позволяет сделать вывод о неэффективности и в некоторых случаях о невозможности применения предшествующих технологий, недостатки которых решаются настоящим изобретением, а именно заявленным эмулятором и способом эмуляции.

Раскрытие сущности изобретения

Первый технический результат заключается в повышении достоверности эмуляции инструкций файла за счет использования реализаций API-функций, содержащихся в обновляемых модулях и в эмуляторе.

Второй технический результат заключается в повышении уровня обнаружения вредоносного кода с использованием эмулятора.

Третий технический результат заключается в снижении времени реакции на новые угрозы.

Согласно варианту реализации, используется эмулятор, содержащий средства для эмуляции исполнения инструкций файла: средство эмуляции, предназначенное для: эмуляции исполнения инструкций файла на процессоре эмулятора; приостановления эмуляции исполнения инструкций на вызове API-функции; проверки наличия API-функции в наборе обновляемых модулей; эмуляции исполнения упомянутой API-функции по инструкциям согласно реализации из соответствующего обновляемого модуля, если API-функция найдена; передачи управления средству исполнения, если API-функция не найдена в наборе обновляемых модулей; продолжения эмуляции исполнения инструкций файла с инструкции по адресу возврата API-функции, используя результат исполнения API-функции; набор обновляемых модулей из по меньшей мере одного обновляемого модуля, где каждый обновляемый модуль из упомянутого набора содержит реализацию по меньшей мере одной API-функции; средство исполнения, предназначенное для формирования результата исполнения упомянутой API-функции согласно реализации API-функции, содержащейся в средстве исполнения или созданной средством исполнения и последующей передачи средству эмуляции результата исполнения API-функции.

Согласно частному варианту реализации в зависимости от файла средство эмуляции служит для эмуляции исполнения инструкций упомянутого файла: в контексте процесса файла для исполняемого файла; в контексте заданного процесса эмулятора, вызывающего функции библиотеки для файла библиотеки; в контексте системного процесса для файла драйвера; в контексте процесса интерпретатора файла для файла скрипта, при этом дополнительно подают на вход упомянутому интерпретатору файл скрипта.

Согласно другому частному варианту реализации файлы библиотеки включают динамически подключаемые библиотеки для ОС Windows.

Согласно еще одному частному варианту реализации средство эмуляции служит для эмуляции исполнения инструкций файла в контексте процесса интерпретатора консольных команд для файла, содержащего консольные команды.

Согласно частному варианту реализации средство исполнения содержит виртуальный код реализации API-функции, при этом виртуальный код состоит из инструкций,

создающих результат исполнения упомянутой API-функции.

Согласно другому частному варианту реализации обновляемый модуль содержит предварительно скомпилированный нативный код реализации API-функции, состоящий из инструкций, полностью имитирующих исполнение упомянутой API-функции.

Согласно еще одному частному варианту реализации обновляемый модуль дополнительно содержит по меньшей мере одну API-функцию, реализованную с использованием виртуального кода.

Согласно частному варианту реализации по меньшей мере один обновляемый модуль содержит API-функции, соответствующей системной библиотеке компьютера.

Согласно другому частному варианту реализации средство исполнения служит для создания реализации API-функции путем создания виртуального кода API-функции, возвращающего результат исполнения в зависимости от параметров вызова и соглашения о вызове упомянутой API-функции.

Согласно еще одному частному варианту реализации средство эмуляции, средство исполнения и по меньшей мере один обновляемый модуль независимо друг от друга получают обновления от удаленного сервера, при этом упомянутые обновления содержат реализацию по меньшей мере одной API-функции.

Согласно частному варианту реализации результат исполнения API-функции включает информацию об изменении регистров процессора эмулятора или изменении стека процессора эмулятора, а средство эмуляции учитывает выявленные изменения при эмуляции исполнения последующих инструкций.

Согласно варианту реализации, используется способ эмуляции, выполняемый при помощи средств эмулятора, в котором: с помощью средства эмуляции производят эмуляцию исполнения инструкций файла на процессоре эмулятора, во время которого приостанавливают эмуляцию исполнения инструкций на вызове API-функции; проверяют наличие API-функции в наборе обновляемых модулей, при этом: если API-функция найдена, с помощью средства эмуляции эмулируют исполнение упомянутой API-функции по инструкциям согласно реализации из соответствующего обновляемого модуля; если API-функция не найдена, с помощью средства исполнения формируют результат исполнения упомянутой API-функции согласно реализации API-функции, содержащейся в средстве исполнения или созданной средством исполнения; с помощью средства эмуляции продолжают эмуляцию исполнения инструкций файла согласно инструкции по адресу возврата API-функции, используя результат исполнения API-функции.

Согласно частному варианту реализации в зависимости от файла с помощью средства эмуляции производят эмуляцию исполнения инструкций упомянутого файла: в контексте процесса файла для исполняемого файла; в контексте заданного процесса эмулятора, вызывающего функции библиотеки для файла библиотеки; в контексте системного процесса для файла драйвера; в контексте процесса интерпретатора файла для файла скрипта, при этом дополнительно подают на вход упомянутому интерпретатору файл скрипта.

Согласно другому частному варианту реализации файлы библиотеки включают динамически подключаемые библиотеки для ОС Windows.

Согласно еще одному частному варианту реализации производят эмуляцию исполнения инструкций файла в контексте процесса интерпретатора консольных команд для файла, содержащего консольные команды.

Согласно частному варианту реализации API-функции в средстве исполнения реализованы с использованием виртуального кода, состоящего из инструкций, создающих результат исполнения упомянутой API-функции.

Согласно другому частному варианту реализации API-функции в обновляемом модуле реализованы с использованием предварительно скомпилированного нативного кода API-функции, состоящего из инструкций, полностью имитирующих исполнение упомянутой API-функции.

Согласно еще одному частному варианту реализации дополнительно по меньшей мере одна API-функция обновляемого модуля реализована с использованием виртуального кода.

Согласно частному варианту реализации с помощью средство исполнения создают код API-функции, возвращающий результат исполнения в зависимости от параметров вызова и соглашения о вызове упомянутой API-функции.

Согласно другому частному варианту реализации результат исполнения API-функции включает информацию об изменении регистров процессора эмулятора или изменении стека процессора эмулятора, при этом эмуляция исполнения последующих инструкций процесса происходит с учетом упомянутых изменений.

Краткое описание чертежей

Дополнительные цели, признаки и преимущества настоящего изобретения будут очевидными из прочтения последующего описания осуществления изобретения со ссылкой на прилагаемые чертежи, на которых:

Фиг. 1 иллюстрирует пример работы эмулятора из уровня техники при обработке вызова API-функции.

На Фиг. 2 приведен возможный пример модулей средства защиты компьютера.

На Фиг. 3 представлена схема исполнения инструкций файла в эмуляторе.

На Фиг. 4 представлен пример эмулятора согласно настоящему изобретению.

На Фиг. 5 представлен вариант способа осуществления настоящего изобретения.

Фиг. 6 представляет пример компьютерной системы общего назначения.

Осуществление изобретения

Объекты и признаки настоящего изобретения, способы для достижения этих объектов и признаков станут очевидными посредством отсылки к примерным вариантам осуществления. Однако, настоящее изобретение не ограничивается примерными вариантами осуществления, раскрытыми ниже, оно может воплощаться в различных видах. Сущность, приведенная в описании, является ничем иным, как конкретными деталями, обеспеченными для помощи специалисту в области техники в исчерпывающем понимании изобретения, и настоящее изобретение определяется в объеме приложенной формулы.

Глоссарий

Эмуляция - комплекс программных, аппаратных средств или их сочетание, предназначенные для копирования функций одной вычислительной системы на другой, отличной от первой, вычислительной системе таким образом, чтобы эмулированное поведение как можно ближе соответствовало поведению оригинальной системы[1] ([1] ГОСТ Р 57721-3017).

Эмуляция исполнения файла - имитация исполнения файла на одном устройстве посредством другого устройства или устройств вычислительной машины.

Эмуляция исполнения инструкции - имитация исполнения инструкции на одном устройстве посредством другого устройства или устройств вычислительной машины. Под инструкцией понимают инструкцию исполняемого кода, которая, в частности, может содержаться в исполняемом файле, равно как и в образе исполняемого файла, а также в неисполняемых файлах, содержащих исполняемый код (например, библиотеке DLL).

Образ файла - представление файла в рамках процесса эмуляции его исполнения, а именно набор данных, описывающих файл по меньшей мере с полнотой, необходимой для его исполнения (а также и эмуляции исполнения). При эмуляции исполнения файла средство эмуляции эмулирует исполнение инструкций образа этого файла. Инструкции файла могут быть формализованы в разном виде: как в виде машинных инструкций, таки и в виде промежуточного кода (инструкции MSIL или P-Code - иногда упоминаются как P-код) или сценария (скрипта).

Под средствами системы эмуляции исполнения файлов в настоящем изобретении понимаются реальные устройства, системы, компоненты, группы компонентов, реализованные с использованием аппаратных средств, таких как интегральные микросхемы (англ. application-specific integrated circuit, ASIC) или программируемые вентильные матрицы (англ. field-programmable gate array, FPGA) или, например, в виде комбинации программных и аппаратных средств, таких как микропроцессорная система и набор программных инструкций, а также на нейроморфных чипах (англ. neurosynaptic chips). Функциональность указанных средств системы может быть реализована исключительно аппаратными средствами, а также в виде комбинации, где часть функциональности средств системы реализована программными средствами, а часть - аппаратными. В некоторых вариантах реализации часть средств или все средства могут быть исполнены на процессоре компьютера общего назначения (например, который изображен на Фиг. 6). При этом компоненты системы могут быть реализованы в рамках как одного вычислительного устройства, так и разнесены между несколькими связанными между собой вычислительными устройствами.

Исполняемый файл - файл, содержащий приложение в виде, в котором оно может быть исполнено на компьютере. Поэтому в изобретении приложение и исполняемый файл рассматриваются как синонимы.

Виртуальный код API-функции - в рамках данного изобретения это код, состоящий из инструкций процессора компьютера, формирующий результат исполнения упомянутой API-функции.

Нативный код API-функции - в рамках данного изобретения это предварительно скомпилированный машинный код API-функции (для виртуального процессора эмулятора), состоящий из инструкций и полностью имитирующий исполнение указанной API-функции. Таким образом, нативный код содержит все или большинство инструкций реальной API-функции, адаптированные для исполнения на виртуальном процессоре эмулятора.

Фиг. 1 иллюстрирует пример работы эмулятора из уровня техники при обработке вызова API-функции на примере эмуляции исполняемого файла. Стоит отметить, что API-функции не ограничиваются Windows API. API-функции могут быть функциями любого используемого API, например Linux Kernel API, OpenGL, Qt и других. Когда при работе приложения (исполняемого файла) в реальной ОС происходит вызов API-функции, ОС производит большое количество действий, что связано со сложной внутренней архитектурой ОС. Схематически вызов API-функции приводит к исполнению большого количества инструкций на процессоре, после чего приложению возвращается результат работы вызванной API-функции. При работе эмулятора вызов API-функции не приводит к такому же исполнению ряда инструкций, что и в реальной ОС, вместо этого приложению возвращается сэмулированный (имитированный) результат работы API-функции. Например, при попытке создать файл эмулятор вернет указатель на виртуальный файл. Однако несмотря на общий результат работы (например, возврат указателя на файл), результат работы вызванной API-функции может отличаться в ОС

и эмуляторе, что связано с тем, что при вызове API-функции могут быть изменены, например, некоторые из регистров процессора, что не будет полностью отражено при работе эмулятора. Данное расхождение может быть использовано для противодействия эмуляции и в первую очередь - вредоносными программами.

На Фиг. 2 приведен возможный пример модулей средства защиты компьютера. Средство защиты (антивирус) компьютера 20 может содержать модули, предназначенные для обеспечения безопасности компьютера 20: сканер по доступу, сканер по требованию, почтовый антивирус, веб антивирус, модуль проактивной защиты, модуль HIPS (англ. Host Intrusion Prevention System - система предотвращения вторжений), DLP-модуль (англ. data loss prevention - предотвращение утечки данных), сканер уязвимостей, эмулятор 400, сетевой экран и др. В частном варианте реализации указанные модули могут быть составной частью средства защиты. В еще одном варианте реализации данные модули могут быть реализованы в виде отдельных программных компонент.

Сканер по доступу содержит функционал обнаружения вредоносной активности всех открываемых, запускаемых и сохраняемых файлов на компьютерной системе пользователя. Сканер по требованию отличается от сканера по доступу тем, что сканирует заданные пользователем файлы и директории по требованию пользователя.

Почтовый антивирус необходим для контроля входящей и исходящей электронной почты на предмет содержания вредоносных объектов. Веб-антивирус служит для предотвращения исполнения вредоносного кода, который может содержаться на веб-сайтах при их посещении пользователем, а также для блокирования открытия веб-сайтов. Модуль HIPS служит для обнаружения нежелательной и вредоносной активности программ и блокирования ее в момент исполнения. DLP-модуль служит для обнаружения и предотвращения утечки конфиденциальных данных за пределы компьютера или сети. Сканер уязвимостей необходим для обнаружения уязвимостей на компьютере 20 (например, отключены некоторые компоненты средства защиты, не актуальные вирусные базы, закрыт сетевой порт и пр.). Сетевой экран осуществляет контроль и фильтрацию сетевого трафика в соответствии с заданными правилами. Работа эмулятора 400 заключается в имитации гостевой системы во время исполнения инструкций файла в эмуляторе 400 и будет подробно рассмотрена далее. Модуль проактивной защиты использует поведенческие сигнатуры для обнаружения поведения исполняемых файлов и их классификации по уровню доверия.

На Фиг. 3 представлена схема исполнения инструкций файла в эмуляторе. Эмулятор 400 установлен на компьютерном устройстве пользователя (далее - компьютер) 20. Эмулятор может являться компонентом средства защиты, которое представлено на Фиг. 2, либо самостоятельным приложением, либо аппаратным компонентом компьютера 20, либо частью ОС 35. В предпочтительном варианте реализации эмулятор 400 используется для обнаружения вредоносного кода в файле 320, но также может использоваться и для других целей, в частности для отладки приложений, таких как файл 320, если он является исполняемым файлом. Роль исполняющего устройства в эмуляторе 400 выполняет средство эмуляции 403, которое непосредственно эмулирует исполнение инструкций. Эмулятор 400 и средства эмулятора 400 будут описаны далее. Приложение, которое может иметь различный функционал, в том числе являться установщиком ПО, представлено на схеме в виде файла 320. В других вариантах реализации файл 320 может быть неисполняемым, и для его воспроизведения потребуется дополнительно компилятор или интерпретатор. Файл 320 хранится на диске 27 или на другой внешней памяти и содержит инструкции, которые могут быть исполнены на

процессоре. Таким образом, в различных вариантах реализации файл 320 может быть одним из:

• исполняемым файлом, например формата PE (англ. Portable Executable);

• файлом библиотеки, например формата динамически подключаемой библиотеки DLL для ОС Windows;

• драйвером;

• файлом скрипта, например формата JavaScript, который может быть исполнен интерпретатором JavaScript, или формата пакетного файла, который может быть исполнен интерпретатором консольных команд, таким как cmd.exe для ОС Windows.

Для простоты изложения в описании далее будет рассматриваться файл 320, являющийся исполняемым файлом. Примером исполняемого файла является файл формата Portable Executable (PE) в ОС Windows. PE-файл содержит следующие основные поля: заголовок, секцию кода, секцию данных и таблицу импорта, которая содержит ссылки на подключаемые библиотеки.

Тем не менее, как упоминалось ранее, файл 320 может быть любым файлом, содержащим инструкции, которые могут быть исполнены на процессоре.

В одном частном примере реализации, когда файл 320 является исполняемым файлом, эмуляция исполнения инструкций файла 320 средством эмуляции 403 происходит в контексте процесса упомянутого исполняемого файла. То есть эмуляция исполнения инструкций файла 320 начинается с точки входа упомянутого процесса.

В другом частном примере реализации, когда файл 320 является библиотекой (например, DLL), эмуляция исполнения инструкций файла библиотеки осуществляется в контексте заданного процесса. Упомянутый процесс - заданный эмулятором 400 процесс, специально созданный как хост-процесс для эмуляции инструкций файлов библиотек. Такой процесс содержит вызовы функций библиотеки, которые необходимо проэмулировать.

В еще одном частном примере реализации, когда файл 320 является драйвером, средство эмуляции 403 вызывает инструкции драйвера, используя системный процесс (в ОС Windows - процесс system), то есть эмулирует исполнение файла драйвера в контексте системного процесса.

В одном из частных примеров реализации, когда файл 320 является файлом скрипта (или библиотекой скриптового языка), средство эмуляции 403 эмулирует инструкции файла скрипта в контексте процесса интерпретатора файла 320, при этом дополнительно подает на вход интерпретатору файл 320. Для некоторых популярных скриптовых языков эмулятор 400 может дополнительно включать реализацию интерпретатора. Однако, обычно файл 320 распространяется вместе с файлом интерпретатора, в этом случае средство эмуляции 403 начнет эмулировать исполнение процесса интерпретатора, которому на вход будет загружен упомянутый файл 320, являющийся файлом скрипта. В одном примере, когда файл является файлом формата "bat", содержащим консольные команды, средство эмуляции 403 будет эмулировать исполнение файла 320 в контексте процесса интерпретатора консольных команд cmd.exe.

В представленном варианте реализации компьютерная система дополнена эмулятором 400, который выполнен в виде программы, исполняемой на уровне приложений. В других частных вариантах реализации эмулятор может являться частью операционной системы 35 или отдельным устройством. Эмулятор 400, загруженный в память 25 компьютерной системы 20 и исполняемый на процессоре 21, осуществляет разбор файла 320 и обработку его кода. Характерной особенностью является то, что анализируемое приложение загружается в адресное пространство эмулятора 400. Программная

реализация эмулятора 400 позволяет обрабатывать код и ресурсы приложения аналогично процессору 21 и операционной системе 35. Эмулятор 400 связан с сервером обновлений 340, который служит для обновления кода эмулятора 400, и отдельных модулей, и средств эмулятора 400.

На Фиг. 4 представлен пример эмулятора 400 согласно настоящему изобретению. В одном из вариантов реализации для построения эмулятора 400, способного исполнить инструкции файла 320, требуется воссоздание процессора 401 (также - виртуального процессора), регистров процессора 404 и памяти 405. Для преобразования инструкций кода во внутреннюю систему команд исполняющих устройств может использоваться используется декодер 402. Регистры процессора 404 располагаются непосредственно в ядре процессора 401. Дополнительно в эмулятор 400 включают файловую систему 408, устройства ввода/вывода 409 и сеть передачи данных 410. В некоторых вариантах реализации возможно воспроизведение функций операционной системы 35 и внешних сервисов. Процесс 420, в контексте которого будет исполняться файл 320, воспроизводится в адресном пространстве эмулятора 400, в котором происходит его размещение в воссозданной памяти 405. Для исполняемого файла процесс 420 является процессом исполняемого файла. Файлы 320 других типов будут исполняться в контексте процесса 420, частные варианты реализации которого приведены ранее. Для контроля программных инструкций, которые обрабатываются в эмуляторе 400 и при необходимости должны быть переданы на исполнение в реальную среду, выделяется отдельный функциональный блок - обработчик запросов 406. Ход исполнения программы в эмуляторе в виде последовательности выполняемых команд ведется модулем журналирования 407. Модуль журналирования 407 может располагаться в эмуляторе 400 и дублироваться на компьютере 20 или располагаться на компьютере 20. В этом случае антивирус будет иметь доступ к модулю журналирования 407 или его копии и оперативно осуществлять обнаружение сигнатур вредоносного кода. Стоит отметить, что указанные на Фиг. 4 средства 401-410 являются виртуальными копиями, имитирующими работу реально присутствующих на компьютере 20 средств, таких как процессор 21, память 25 и других аппаратных и программных средств компьютера 20.

Эмулятор 400 содержит средство эмуляции 403 и связанное с ним средство исполнения 440 вместе с виртуальными функциями 441. Основными элементами реального процессора 21 являются исполняющие устройства, которые непосредственно выполняют все инструкции. Среди них можно выделить две основные группы: арифметико-логические устройства (англ. arithmetic and logic unit, ALU) и блок вычислений с плавающей точкой (англ. floating point unit, FPU). Поэтому, в эмуляторе 400 роль исполняющего устройства выполняет средство эмуляции 403. Кроме того, в эмуляторе 400 содержится набор обновляемых модулей 430. Набор модулей состоит из по меньшей мере одного обновляемого модуля. Каждый обновляемый модуль содержит реализацию одной или нескольких нативных функций. При этом в предпочтительном варианте реализации обновляемый модуль содержит нативную реализацию нескольких API-функций, относящихся к одной библиотеке, реально присутствующей на компьютере. В частном варианте реализации для остальных функций библиотеки, для которых не написан нативный код, будет использоваться генерируемый код - созданный код, который может возвращать сгенерированный результат, который может быть произвольным, либо зависеть от типа функции, либо возвращать значение по умолчанию. Сгенерированный код может быть использован для функций, не выполняющих полезного функционала или редко используемых, например FlushFileBuffers, FreeConsole, LockFile, SetConsoleCursor. При этом сгенерированный код

будет реализован в виде виртуального кода и будет исполняться средством исполнения 440. То есть сгенерированный код также является виртуальным кодом. В еще одном частном варианте реализации для остальных функций библиотеки может вовсе отсутствовать их реализация, в этом случае управление будет передано средству исполнения 440.

Под нативным кодом в данном изобретении понимается предварительно скомпилированный машинный код API-функции, состоящий из инструкций, рассчитанных на исполнение под эмулятором, и полностью имитирующий исполнение указанной API-функции. Таким образом, нативный код содержит все или большинство инструкций реальной API-функции, адаптированные для исполнения на виртуальном процессоре 401 эмулятора 400. Стоит также отметить, что упомянутый нативный код может быть исполнен и на реальном процессоре компьютера. Исполнение нативного кода API-функций будет эмулироваться средством эмуляции 403 на виртуальном процессоре 401 эмулятора 400 так же, как и исполнение процесса 420, в контексте которого выполняются инструкции файла 320. При этом эмуляция исполнения виртуального кода API-функций 441 уже будет осуществлена средством исполнения 440 на реальном процессоре 21 компьютера 20, так как виртуальный код не выполняет подробную имитацию функции, а служит для формирования результата исполнения API-функции. Упомянутый сформированный результат содержит непосредственно результат исполнения API-функции, т.е. возвращаемое значение и изменение состояния регистров и/или стека процесса в зависимости от реализации виртуального кода упомянутой API-функции. Например, в ответ на вызов функции работы с файлом будет возвращен файловый описатель запрашиваемого файла.

В частном варианте реализации настоящего изобретения по меньшей мере один обновляемый модуль 430 содержит две реализации API-функций, написанных для процессоров с 32-битной и с 64-битной архитектурами. В другом частном варианте реализации для по меньшей мере одного обновляемого модуля 430, содержащего реализацию 32-битных API-функций, написан также дополнительный обновляемый модуль, содержащий реализацию тех же API-функций для 64-битной архитектуры. При этом используется единый код эмулятора 400 (средств 401-410, средства исполнения 440) для эмуляции 32-битных и 64-битных API-функций. Это позволит эмулировать исполнение процессов файлов, написанных как для 32-битной, так и для 64-битной архитектуры процессора. За счет этого в заявленном изобретении будет снижен объем дискового пространства, занимаемый основным кодом эмулятора 400, и увеличена скорость и производительность эмуляции. Стоит также отметить, что модули, содержащие 32-битную реализацию API-функций, и модули, содержащие 64-битную реализацию API-функций, могут обновляться сервером обновлений 340 независимо друг от друга.

Средство исполнения 440 содержит виртуальные функции 441 и их реализацию. Эмулятор 400 связан с сервером обновлений 340, который служит для обновления модулей 430, виртуальных функций 441, средства эмуляции 403 и средства исполнения 440. При этом обновление указанных средств может осуществляться независимо.

Следует отметить, что указанные средства эмулятора 400 и обновляемые модули эмулятора 430 могут быть реализованы различными способами. В предпочтительном варианте реализации каждый обновляемый модуль 430 хранится на компьютере 20 отдельно друг от друга (например, в разных файлах). Средство эмуляции 403, средство исполнения 440 и виртуальные функции 441 также могут храниться отдельно друг от друга. В другом примере один файл может включать несколько средств и/или модулей.

Например, средство исполнения 440 может содержать виртуальные функции 441.

В предпочтительном варианте реализации API-функции реализованы в единственном экземпляре - либо в одном из обновляемых модулей 430, либо в виде виртуальной функции 441. В частном варианте реализации возможно дублирование реализации функции в нескольких местах. Например, когда определенная функция была изначально реализована в виде виртуальной функции 441. Затем была разработана нативная реализация указанной функции в обновляемом модуле 430, который был обновлен и передан на компьютер сервером обновления 340. Ввиду того, что обновление средств 403, 440 происходит реже, чем обновление модулей 430, то для указанной функции в эмуляторе 400 будет две реализации - в обновляемом модуле 430 и в виде виртуальной функции 441. Выбор конкретной реализации функции, которая будет использоваться в процессе эмуляции, будет произведен либо на этапе инициализации эмулятора 400, либо в непосредственно в процессе эмуляции. Инициализация эмулятора 400 будет рассмотрена далее.

Ниже кратко изложен процесс загрузки операционной системы на примере ОС Windows. Вначале загружается Windows Boot Manager, который отвечает за поиск и выбор для загрузки установленных ОС Windows. Далее происходит загрузка базовых драйверов, которые отвечают, например, за возможность работы с разделом жесткого диска, на котором установлена выбранная ОС. После чего с диска считывается и загружается в память часть ядра ОС, например Ntosml.exe и hal.dll, производится инициализация реестра, менеджера памяти, менеджера объектов и т.д. Затем происходит загрузка менеджера сессий (smss.exe), который отвечает за загрузку системных переменных, подсистемы Win32 и дальнейшую загрузку winlogon.exe. После того как пользователь успешно пройдет аутентификацию, произойдет загрузка приложений и служб, прописанных с ключом автозапуска, после чего ОС будет полностью готова к взаимодействию с пользователем, ожидая запуска приложений и ввода данных.

Для инициализации и работы эмулятора 400 необязательно полностью эмулировать загрузку операционной системы 35. Например, могут быть оставлены только этапы загрузки в память 405 эмулятора 400 ядра ОС 35 и менеджера сессий в упрощенном виде. Т.е. достаточно будет проэмулировать самый необходимый для работы ОС 35 функционал, который позволит провести эмуляцию приложения. Для Win32-приложений нужно будет проэмулировать запуск smss.exe с последующим запуском csrss.exe, что инициализирует Windows-подсистему и позволит создавать процессы и потоки. Так как эмуляция потенциального вредоносного приложения требует создания более детальной работающей среды (например, эмуляции других запущенных процессов), то также потребуется проэмулировать запуск winlogon.exe с последующим «запуском» таких процессов, как explorer.exe и services.exe, при этом от последнего можно эмулировать «запуск» процессов svchost.exe. Под термином «запуск» в данном случае подразумевается воссоздание в эмуляторе тех же процессов, которые протекают и при создании процессов в рамках реальной ОС, пусть и в сильно упрощенном виде. Подобный подход позволяет воссоздать реальную ОС в достаточной степени, чтобы можно было запустить практически любое приложение, предназначенное для работы в данной ОС.

Кроме того, инициализация эмулятора 400 включает загрузку в память 405 средств 401-410, 440, 441 и части или всех обновляемых модулей 430. Процесс 420, в контексте которого будет исполняться файл 320, загружают в память 405 перед началом эмуляции файла 320.

В одном частном варианте реализации эмулятор 400 инициализирован перед эмуляцией исполнения процесса 420. В еще одном частном варианте реализации при

инициализации эмулятора 400 в память 405 загружают обновляемые модули 430 согласно таблице импорта файла 320, то есть те обновляемые модули, которые содержат реализацию функций из таблицы импорта файла 320. Таким образом, эмулятор 400, а именно средство эмуляции 403 и средство исполнения 440, после инициализации эмулятора 400 «знают», какие API-функции они будут эмулировать или выполнять. Соответственно, в описанном ранее способе средство эмуляции 403 передает управление средству исполнения 440, если API-функция не найдена в наборе обновляемых модулей, при этом такая операция не занимает много времени. В другом частном варианте реализации, если выполняется полная антивирусная проверка (например, по требованию или по расписанию), загружают в память 405 эмулятора 400 все обновляемые модули 430 из набора модулей. Под полной антивирусной проверкой понимается проверка всех файлов определенного диска, раздела диска или директории. В еще одном частном варианте реализации при исполнении антивирусной проверки по доступу загружают в память 405 заданное количество наиболее используемых обновляемых модулей из набора модулей. В еще одном частном варианте реализации эмулятор 400 инициализирован путем загрузки образа эмулятора (или состояние эмулятора, англ. snapshot) перед эмуляцией исполнения процесса 420. Примеры использования образов эмулятора описаны в патенте RU 2553056.

Стоит отметить, что в случае, если в обновляемом модуле 430 содержится реализация API-функции, дублирующая реализацию виртуальной API-функции 441, то в обновляемом модуле 430 также будет содержаться информация о том, какую реализацию API-функции считать приоритетной.

Такая информация может содержаться в виде таблицы с именами API-функций и их приоритетами либо датой реализации. Соответственно, во время инициализации эмулятора 400 в память 405 будут загружены приоритетные варианты реализации API-функций. Таким образом, для каждой API-функции в памяти будет единственная реализация, обеспечивающая лучшую эффективность эмулятора.

На Фиг. 5 представлен вариант способа осуществления настоящего изобретения. Способ эмуляции осуществляется инициализированным эмулятором 400. Процесс инициализации эмулятора 400 и частные варианты реализации инициализации эмулятора 400 были приведены ранее. После инициализации эмулятора 400 и перед началом эмуляции в память 405 эмулятора 400 загружается процесс 420, в контексте которого будут исполняться инструкции файла 320.

На шаге 501 с помощью средства эмуляции 403 производят эмуляцию исполнения инструкций файла 320 на виртуальном процессоре 401 эмулятора 400. С помощью средства эмуляции 403 приостанавливают эмуляцию исполнения процесса 420 на вызове API-функции. На шаге 502 проверяют, содержится ли API-функция в одном из обновляемых модулей 430 из набора модулей, указанных на Фиг. 4. Если API-функция найдена, на шаге 503 с помощью средства эмуляции 403 эмулируют исполнение API-функции по инструкции согласно реализации указанной API-функции из соответствующего обновляемого модуля 430. Если API-функция не найдена в наборе обновляемых модулей, на шаге 504 с помощью средства исполнения 440 формируют результат исполнения API-функции согласно реализации API-функции, которая содержится или создана с помощью средства исполнения 440. При этом эмуляция средством исполнения 440 будет происходить не на виртуальном процессоре 401 эмулятора 400, а на реальном процессоре 21 компьютера 20 в контексте процесса эмулятора 400. В итоге на шаге 505 с помощью средства эмуляции 403 продолжают эмуляцию исполнения процесса с инструкции процесса, расположенной по адресу

возврата API-функции, используя результат эмуляции исполнения API-функции.

Стоит также отметить, что в процессе эмуляции может осуществляться запись в журнал данных о вызовах API-функций и данных о возвратах из API-функций с помощью модуля журналирования 407. В процессе эмуляции или после завершения эмуляции антивирусом, связанным с эмулятором, осуществляется поиск вирусных сигнатур с использованием эвристических правил и, в случае нахождения сигнатуры, файл 320 признается вредоносным. После чего принимаются меры по устранению возникшей угрозы, такие как запрет запуска файла на реальном компьютере, попытка лечения или удаление файла и другие действия.

Эмуляция может продолжаться до наступления условия остановки эмуляции, в частности одного из следующих:

- проэмулировано заданное количество инструкций и/или API-функций;
- прошло время, выделенное на эмуляцию;
- сработала вирусная сигнатура;
- выполнение других условий.

В одном из частных примеров реализации, при отсутствии в памяти 405 обновляемого модуля 430, содержащего указанную API-функцию, загружают указанный обновляемый модуль 430 в память 405. После чего продолжают эмуляцию указанной API-функции средством эмуляции 403. В еще одном примере некоторые процессы не будут загружены в память, тем не менее в списке процессов эмулятора 400 будут отображаться имена и уникальные идентификаторы таких процессов. При этом при обращении инструкции исследуемого файла 320 к одному из таких процессов, процесс будет загружен в память по требованию для продолжения корректной эмуляции.

В еще одном частном примере реализации с помощью средства исполнения 440 создают код API-функции, возвращающий результат исполнения API-функции. Созданный код может быть заглушкой. Кроме того, созданный код может возвращать сгенерированный результат, который может быть произвольным либо зависеть от параметров вызова и соглашения о вызове упомянутой API-функции - например, в ответ на вызов функции работы с файлом будет возвращен произвольный файловый описатель, а для функции, принимающей целые числа, будет возвращено произвольное целое число, например 0. При этом эмуляция может быть остановлена, если неизвестны параметры вызова или соглашение о вызове. Стоит отметить, что созданный код может быть как виртуальным, так и нативным.

Раскрытое изобретение решает заявленную техническую проблему, так как повышает качество эмуляции инструкций файла за счет использования реализаций API-функций, содержащихся в обновляемых модулях и в эмуляторе. И, таким образом, достигаются заявленные технические результаты. А именно, будет повышена достоверность эмуляции инструкций файла за счет использования реализаций API-функций, содержащихся в обновляемых модулях и в эмуляторе. Достоверность эмуляции тем выше, чем больше функций реальной системы способен имитировать эмулятор. Кроме того, чем более точно имитируются функции реальной системы эмулятором, тем выше достоверность эмуляции. Кроме того, будет достигнут технический результат, заключающийся в повышении уровня обнаружения вредоносного кода с использованием эмулятора, за счет использования реализаций API-функций из обновляемых модулей, а в случае отсутствия в обновляемых модулях реализаций API-функций, будут использованы реализации API-функций, содержащиеся или созданные эмулятором. Также будет достигнут технический результат, заключающийся в снижении времени реакции на новые угрозы, за счет использования обновляемых модулей, содержащих реализацию

API-функций.

Фиг. 6 представляет пример компьютерной системы общего назначения, персональный компьютер или сервер 20, содержащий центральный процессор 21, системную память 22 и системную шину 23, которая содержит разные системные компоненты, в том числе память, связанную с центральным процессором 21. Системная шина 23 реализована, как любая известная из уровня техники шинная структура, содержащая в свою очередь память шины или контроллер памяти шины, периферийную шину и локальную шину, которая способна взаимодействовать с любой другой шинной архитектурой. Системная память содержит постоянное запоминающее устройство (ПЗУ) 24, память с произвольным доступом (ОЗУ) 25. Основная система ввода/вывода (BIOS) 26, содержит основные процедуры, которые обеспечивают передачу информации между элементами персонального компьютера 20, например, в момент загрузки операционной системы с использованием ПЗУ 24.

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

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

Компьютер 20 имеет файловую систему 36, где хранится записанная операционная система 35, а также дополнительные программные приложения 37, другие программные модули 38 и данные программ 39. Пользователь имеет возможность вводить команды и информацию в персональный компьютер 20 посредством устройств ввода (клавиатуры 40, манипулятора «мышь» 42). Могут использоваться другие устройства ввода (не отображены): микрофон, джойстик, игровая консоль, сканер и т.п.Подобные устройства ввода по своему обычаю подключают к компьютерной системе 20 через последовательный порт 46, который в свою очередь подсоединен к системной шине, но могут быть подключены иным способом, например, при помощи параллельного порта, игрового порта или универсальной последовательной шины (USB). Монитор 47 или иной тип устройства отображения также подсоединен к системной шине 23 через интерфейс, такой как видеоадаптер 48. В дополнение к монитору 47, персональный компьютер может быть оснащен другими периферийными устройствами вывода (не отображены), например, колонками, принтером и т.п.

Персональный компьютер 20 способен работать в сетевом окружении, при этом используется сетевое соединение с другим или несколькими удаленными компьютерами 49. Удаленный компьютер (или компьютеры) 49 являются такими же персональными компьютерами или серверами, которые имеют большинство или все упомянутые

элементы, отмеченные ранее при описании существа персонального компьютера 20, представленного на Фиг. 6. В вычислительной сети могут присутствовать также и другие устройства, например, маршрутизаторы, сетевые станции, пиринговые устройства или иные сетевые узлы.

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

В соответствии с описанием, компоненты, этапы исполнения, структура данных, описанные выше, могут быть выполнены, используя различные типы операционных систем, компьютерных платформ, программ.

В заключение следует отметить, что приведенные в описании сведения являются примерами, которые не ограничивают объем настоящего изобретения, определенного формулой.

(57) Формула изобретения

1. Эмулятор исполнения инструкций файла, реализуемый посредством компьютерной системы, включающей по меньшей мере память и функционально связанный с памятью процессор, выполненный с возможностью осуществлять следующие средства:

а) средство эмуляции, предназначенное для:

• эмуляции исполнения инструкций файла на виртуальном процессоре эмулятора;

• приостановления эмуляции исполнения инструкций на вызове API-функции;

• проверки наличия API-функции в наборе обновляемых модулей;

• эмуляции исполнения упомянутой API-функции по инструкциям согласно реализации из соответствующего обновляемого модуля, если API-функция найдена;

• передачи управления средству исполнения, если API-функция не найдена в наборе обновляемых модулей;

• продолжения эмуляции исполнения инструкций файла с инструкции по адресу возврата API-функции, используя результат исполнения API-функции;

б) содержащийся в памяти набор обновляемых модулей из по меньшей мере одного обновляемого модуля, где каждый обновляемый модуль из упомянутого набора содержит реализацию по меньшей мере одной API-функции;

в) средство исполнения, предназначенное для формирования результата исполнения упомянутой API-функции согласно реализации API-функции, содержащейся в средстве исполнения или созданной средством исполнения и последующей передачи средству эмуляции результата исполнения API-функции.

2. Эмулятор по п. 1, в котором в зависимости от файла средство эмуляции служит для эмуляции исполнения инструкций упомянутого файла:

а) в контексте процесса файла для исполняемого файла;

б) в контексте заданного процесса эмулятора, вызывающего функции библиотеки

для файла библиотеки;

в) в контексте системного процесса для файла драйвера;

г) в контексте процесса интерпретатора файла для файла скрипта, при этом дополнительно подают на вход упомянутому интерпретатору файл скрипта.

3. Эмулятор по п. 2, в котором файлы библиотеки включают динамически подключаемые библиотеки для ОС Windows.

4. Эмулятор по п. 2, в котором средство эмуляции служит для эмуляции исполнения инструкций файла в контексте процесса интерпретатора консольных команд для файла, содержащего консольные команды.

5. Эмулятор по п. 1, в котором средство исполнения содержит виртуальный код реализации API-функции, при этом виртуальный код состоит из инструкций, создающих результат исполнения упомянутой API-функции.

6. Эмулятор по п. 1, в котором обновляемый модуль содержит предварительно скомпилированный нативный код реализации API-функции, состоящий из инструкций, полностью имитирующих исполнение упомянутой API-функции.

7. Эмулятор по п. 6, в котором обновляемый модуль дополнительно содержит по меньшей мере одну API-функцию, реализованную с использованием виртуального кода.

8. Эмулятор по п. 1, в котором по меньшей мере один обновляемый модуль содержит API-функции, соответствующей системной библиотеке компьютера.

9. Эмулятор по п. 1, в котором средство исполнения служит для создания реализации API-функции путем создания виртуального кода API-функции, возвращающего результат исполнения в зависимости от параметров вызова и соглашения о вызове упомянутой API-функции.

10. Эмулятор по п. 1, в котором средство эмуляции, средство исполнения и по меньшей мере один обновляемый модуль независимо друг от друга получают обновления от удаленного сервера, при этом упомянутые обновления содержат реализацию по меньшей мере одной API-функции.

11. Эмулятор по п. 1, в котором результат исполнения API-функции включает информацию об изменении регистров виртуального процессора эмулятора или изменении стека виртуального процессора эмулятора, а средство эмуляции учитывает выявленные изменения при эмуляции исполнения последующих инструкций.

12. Способ эмуляции исполнения инструкций файла, выполняемый на процессоре компьютера при помощи средств эмулятора по п. 1, в котором:

а) с помощью средства эмуляции производят эмуляцию исполнения инструкций файла на виртуальном процессоре эмулятора, во время которого приостанавливают эмуляцию исполнения инструкций на вызове API-функции;

б) проверяют наличие API-функции в наборе обновляемых модулей, содержащихся в памяти, при этом:

• если API-функция найдена, с помощью средства эмуляции эмулируют исполнение упомянутой API-функции по инструкциям согласно реализации из соответствующего обновляемого модуля;

• если API-функция не найдена, с помощью средства исполнения формируют результат исполнения упомянутой API-функции согласно реализации API-функции, содержащейся в средстве исполнения или созданной средством исполнения;

в) с помощью средства эмуляции продолжают эмуляцию исполнения инструкций файла согласно инструкции по адресу возврата API-функции, используя результат исполнения API-функции.

13. Способ по п. 12, в котором в зависимости от файла с помощью средства эмуляции

производят эмуляцию исполнения инструкций упомянутого файла:

а) в контексте процесса файла для исполняемого файла;

б) в контексте заданного процесса эмулятора, вызывающего функции библиотеки для файла библиотеки;

в) в контексте системного процесса для файла драйвера;

г) в контексте процесса интерпретатора файла для файла скрипта, при этом дополнительно подают на вход упомянутому интерпретатору файл скрипта.

14. Способ по п. 12, в котором файлы библиотеки включают динамически подключаемые библиотеки для ОС Windows.

15. Способ по п. 12, в котором производят эмуляцию исполнения инструкций файла в контексте процесса интерпретатора консольных команд для файла, содержащего консольные команды.

16. Способ по п. 12, в котором API-функции в средстве исполнения реализованы с использованием виртуального кода, состоящего из инструкций, создающих результат исполнения упомянутой API-функции.

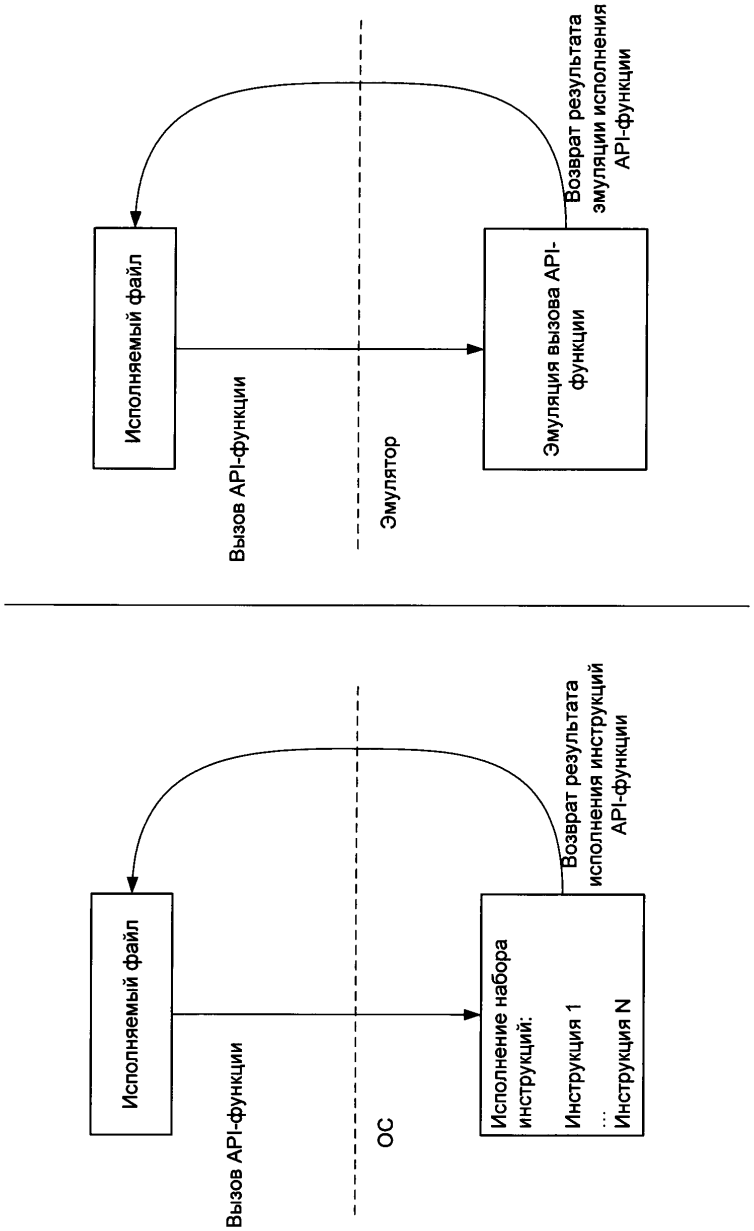
17. Способ по п. 12, в котором API-функции в обновляемом модуле реализованы с использованием предварительно скомпилированного нативного кода API-функции, состоящего из инструкций, полностью имитирующих исполнение упомянутой API-функции.

18. Способ по п. 17, в котором дополнительно по меньшей мере одна API-функция обновляемого модуля реализована с использованием виртуального кода.

19. Способ по п. 12, в котором с помощью средство исполнения создают код API-функции, возвращающий результат исполнения в зависимости от параметров вызова и соглашения о вызове упомянутой API-функции.

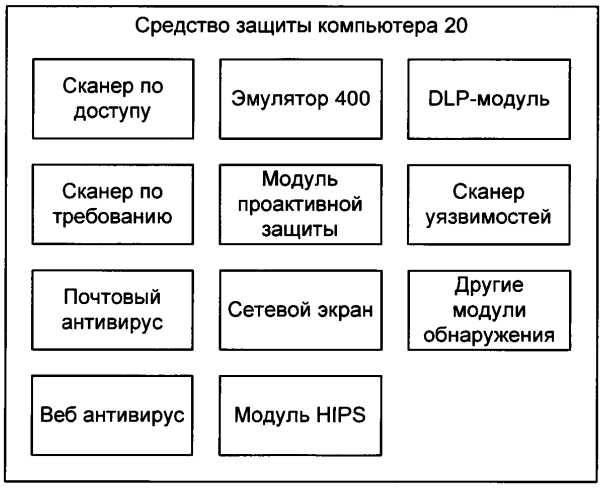
20. Способ по п. 12, в котором результат исполнения API-функции включает информацию об изменении регистров виртуального процессора эмулятора или изменении стека виртуального процессора эмулятора, при этом эмуляция исполнения последующих инструкций процесса происходит с учетом упомянутых изменений.

1


Исполняемый файл
Вызов API-функции
Эмулятор
Эмуляция вызова API-функции
Возврат результата эмуляции исполнения API-функции
Исполняемый файл
Вызов API-функции
ОС
Исполнение набора инструкций:
Инструкция 1
...
Инструкция N
Возврат результата исполнения инструкций API-функции

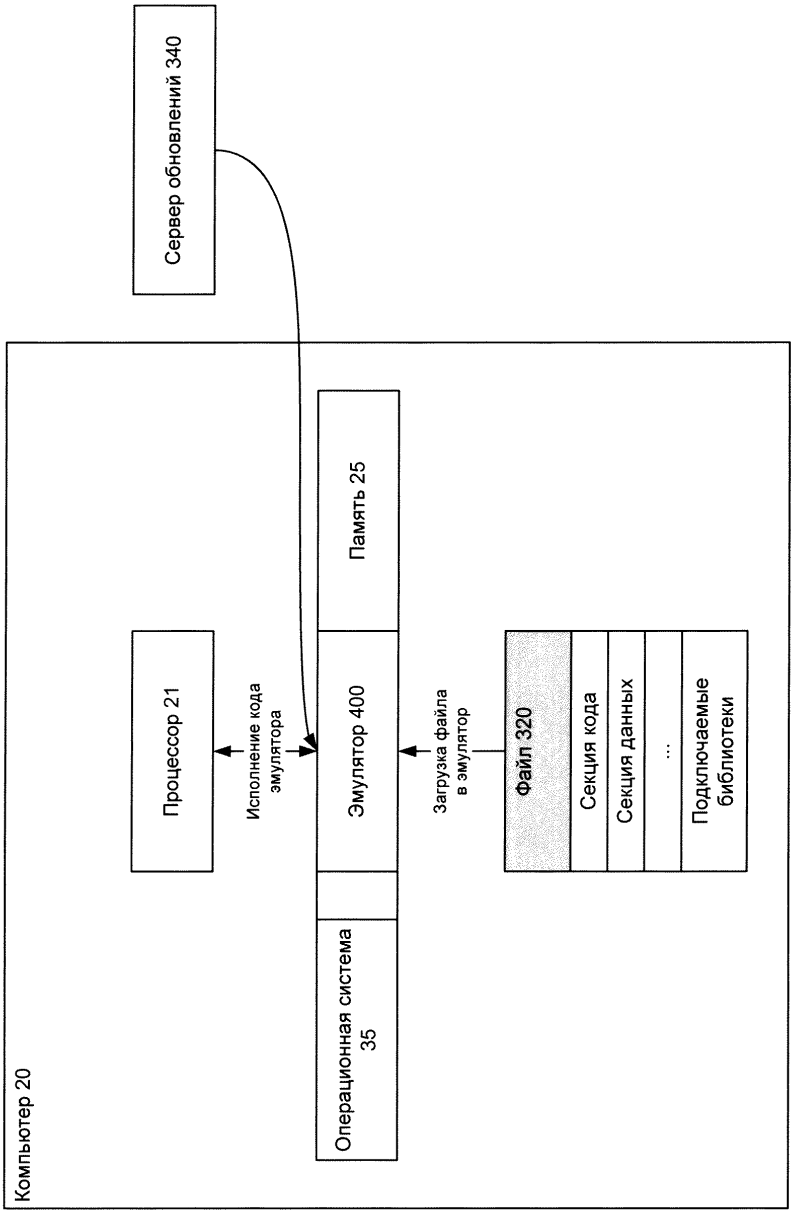
Фиг. 1

2


Средство защиты компьютера 20
Сканер по доступу
Эмулятор 400
DLP-модуль
Сканер по требованию
Модуль проактивной защиты
Сканер уязвимостей
Почтовый антивирус
Сетевой экран
Другие модули обнаружения
Веб антивирус
Модуль HIPS

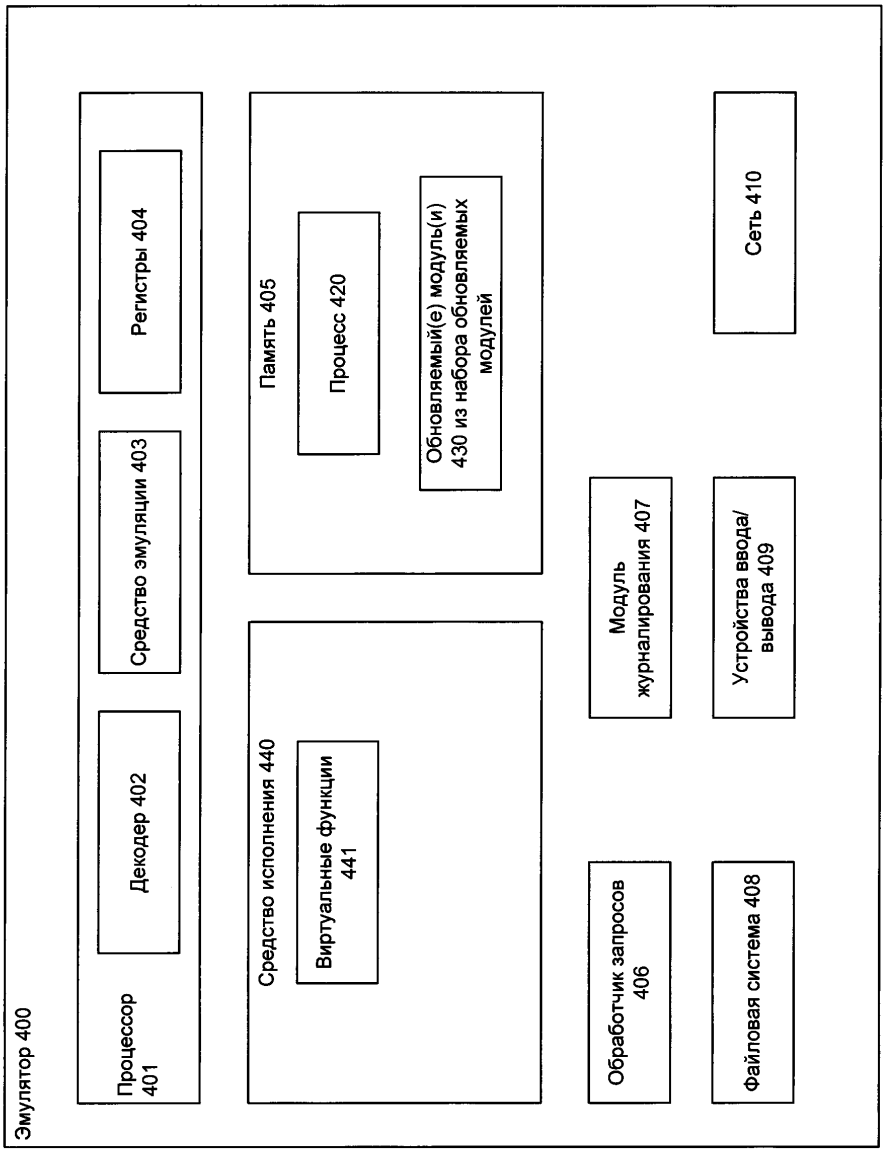
Фиг. 2

3

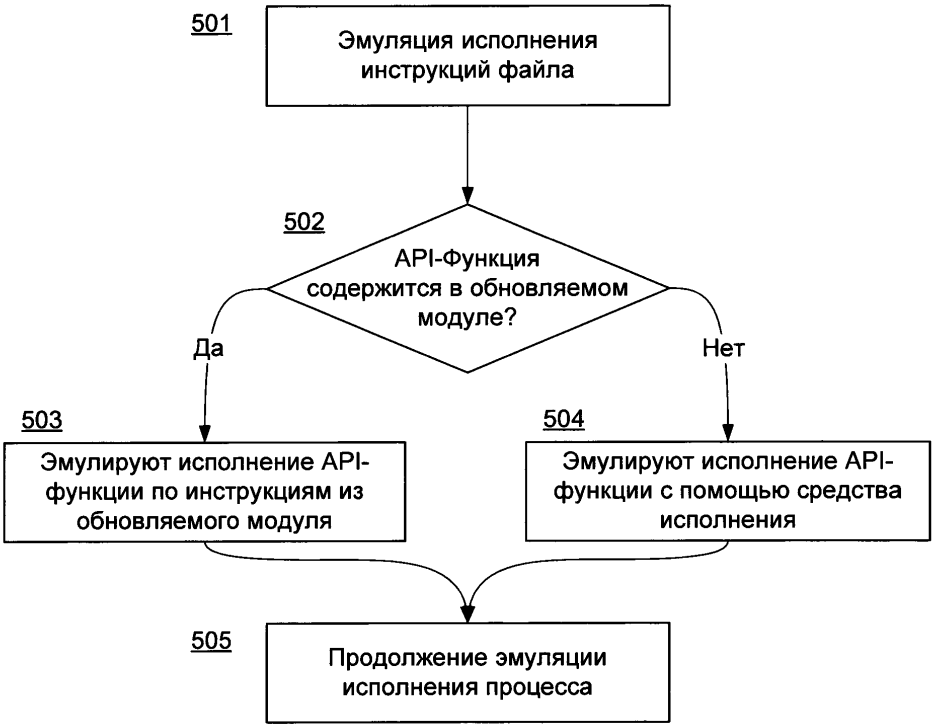

Сервер обновлений 340
Компьютер 20
Процессор 21
Исполнение кода эмулятора
Эмулятор 400
Память 25
Операционная система 35
Загрузка файла в эмулятор
Файл 320
Секция кода
Секция данных
...
Подключаемые библиотеки

Фиг. 3

4

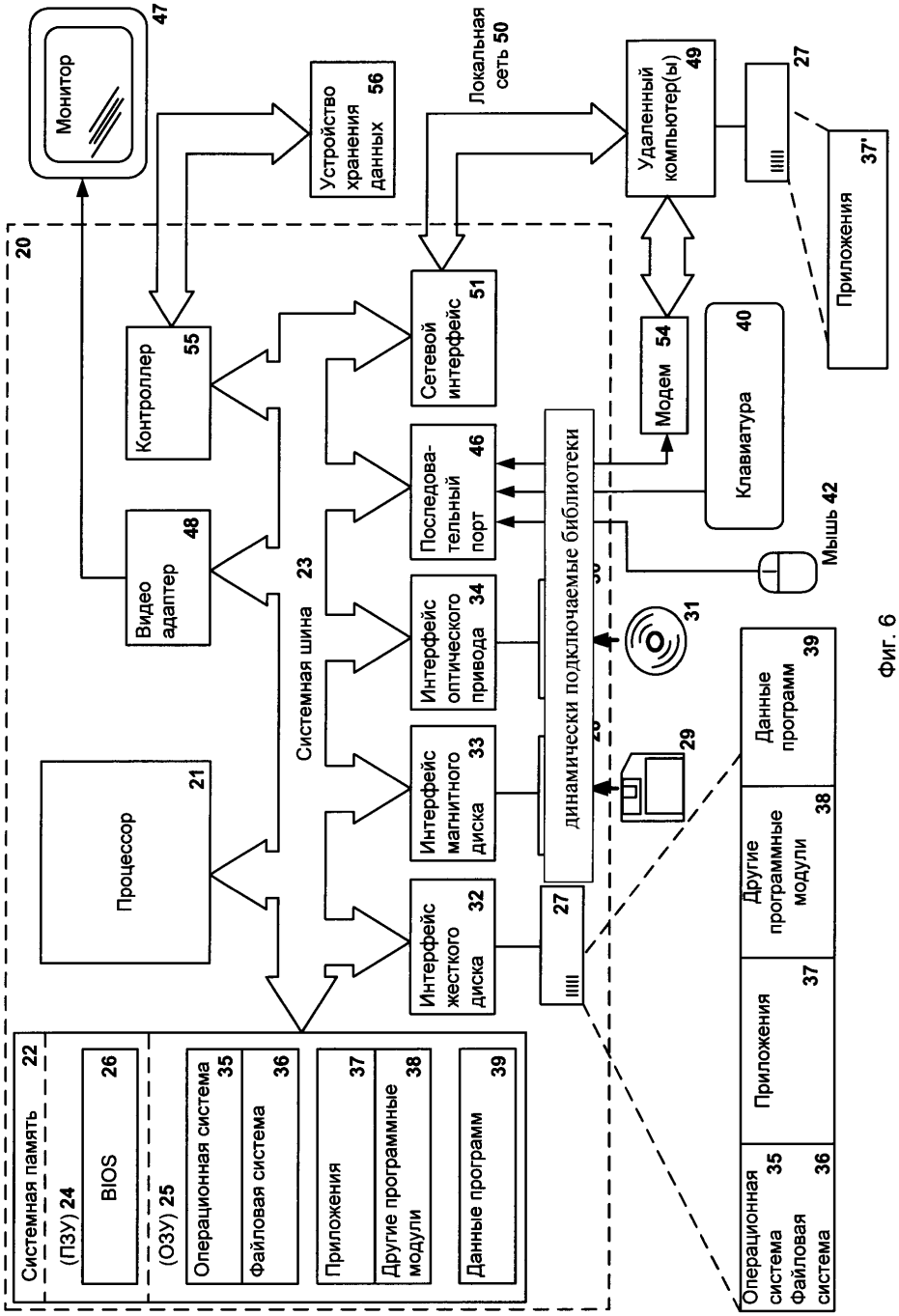

Эмулятор 400
Процессор 401
Декодер 402
Средство эмуляции 403
Регистры 404
Средство исполнения 440
Виртуальные функции 441
Память 405
Процесс 420
Обновляемый(е) модуль(и) 430 из набора обновляемых модулей
Обработчик запросов 406
Модуль журналирования 407
Файловая система 408
Устройства ввода/вывода 409
Сеть 410

Фиг. 4


501
Эмуляция исполнения инструкций файла
502
API-Функция содержится в обновляемом модуле?
Да
Нет
503
Эмулируют исполнение API-функции по инструкциям из обновляемого модуля
504
Эмулируют исполнение API-функции с помощью средства исполнения
505
Продолжение эмуляции исполнения процесса

Фиг. 5

6


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

Фиг. 6