РОССИЙСКАЯ ФЕДЕРАЦИЯ

(19) **RU** (11) **2 377 639** (13) **C2**

(51) МПК
*G06F 12/14* (2006.01)

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ,
ПАТЕНТАМ И ТОВАРНЫМ ЗНАКАМ

# (12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(21), (22) Заявка: **2004135454/09, 03.12.2004**

(24) Дата начала отсчета срока действия патента:
**03.12.2004**

(30) Конвенционный приоритет:
**05.12.2003 US 10/729,823**

(43) Дата публикации заявки: **10.05.2006**

(45) Опубликовано: **27.12.2009** Бюл. № **36**

(56) Список документов, цитированных в отчете о поиске: **RU 2077113 C1, 10.04.1997. RU 2148856 C1, 10.05.2000. US 2002112183 A, 15.08.2002. JP 2003140890 A, 16.05.2003.**

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

(72) Автор(ы):
**ТАУНСЕНД Стефен В. (US),**
**ФЭЙКС Томас Ф. (US)**

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

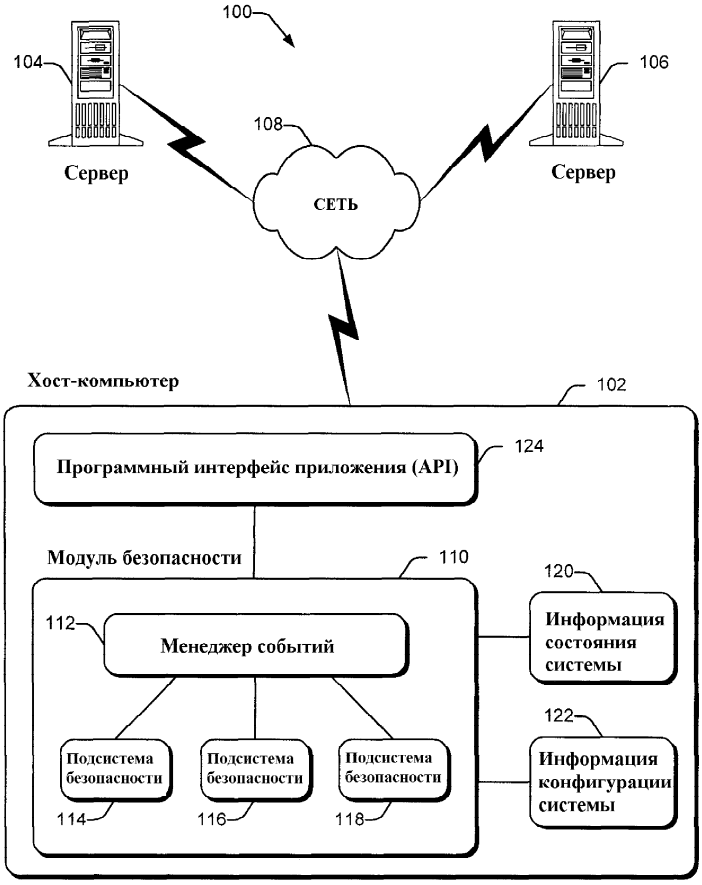
## (54) ПРОГРАММНЫЙ ИНТЕРФЕЙС, СВЯЗАННЫЙ С БЕЗОПАСНОСТЬЮ

(57) Реферат:

Изобретение относится к вычислительным системам, в частности к интерфейсу, связанному с обработкой событий, относящихся к безопасности. Техническим результатом является расширение функциональных возможностей. На компьютерно-считываемом носителе хранятся инструкции, которые при исполнении их на компьютере реализуют программный интерфейс, который включает в себя первую группу функций, относящихся к передаче новой политики безопасности множественным подсистемам безопасности. Каждая из компьютерных систем включает в себя множество подсистем безопасности и реализована с возможностью замены существующей политики безопасности новой политикой безопасности. Программный интерфейс также включает в себя вторую группу функций, относящихся к передаче указания о готовности каждой подсистемы безопасности к реализации новой политики безопасности. Способ осуществления политики безопасности описывает работу компьютерных систем. 4 н. и 28 з.п. ф-лы, 21 ил.

RU 2377639 C2

RU 2377639 C2




**Фиг. 1**

RU 2377639 C2

RU 2377639 C2

RUSSIAN FEDERATION

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY,
PATENTS AND TRADEMARKS

(19) **RU** (11) **2 377 639** (13) **C2**

(51) Int. Cl.
*G06F 12/14* (2006.01)

## (12) ABSTRACT OF INVENTION

(21), (22) Application: **2004135454/09, 03.12.2004**

(24) Effective date for property rights:
**03.12.2004**

(30) Priority:
**05.12.2003 US 10/729,823**

(43) Application published: **10.05.2006**

(45) Date of publication: **27.12.2009 Bull. 36**

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

(72) Inventor(s):
**TAUNSEND Stefen V. (US),**
**FEhJKS Tomas F. (US)**

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

## (54) SECURITY-RELATED PROGRAM INTERFACE

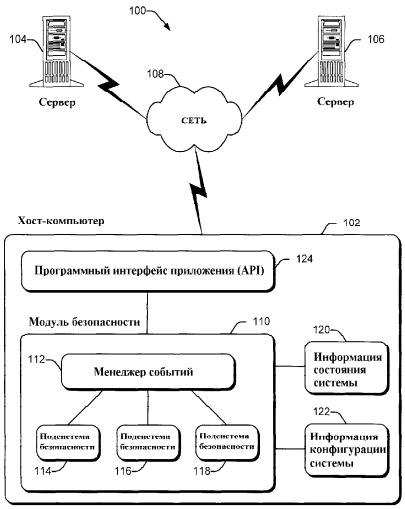
(57) Abstract:

FIELD: physics; computer engineering.

SUBSTANCE: invention relates to computer systems, specifically to an interface associated with processing security related events. Computer storage medium stores instructions which, when executed on a computer, realise a program interface which includes a first group of functions associated with transmitting a new security policy to several security subsystems. Each computer system includes several security subsystems and is made with possibility replacing the existing security policy with a new security policy. The program interface also includes a second group of functions associated with transmission an instruction on readiness of each security subsystem to implement the new security policy. The method of implementing the security policy describes operation of computer systems.

EFFECT: wider functional capabilities.

32 cl, 21 dwg




Фиг. 1

RU 2377639 C2

RU 2377639 C2

Область, к которой относится изобретение

Описанные здесь системы и способы относятся к вычислительным системам и, в частности, к интерфейсу, связанному с обработкой событий, например событий, относящихся к безопасности, и другой информации.

Уровень техники

Популярность компьютерных систем продолжает расти, и они часто бывают связаны с другими компьютерными системами посредством сетей, например локальных сетей (ЛС) и Интернета. Такие особенности, как электронная почта (email), мгновенный обмен сообщениями и онлайновые развлечения, побуждают использовать компьютерные системы, подключенные к сетям. Эти особенности позволяют пользователям, например, связываться с другими пользователями, извлекать аудио- и/или видеосодержимое и приобретать продукты или услуги из онлайновых источников.

Это усиление взаимосвязи компьютерных систем повышает вероятность атак на компьютерные системы со стороны злонамеренных пользователей. Эти атаки могут включать в себя установку вредоносных программ на компьютерах других пользователей (например, предназначенных для блокирования компьютеров других пользователей, для получения информации из компьютеров других пользователей, запуска атак против других компьютеров и т.п.). Атаки могут также включать в себя попытки заблокировать компьютер, чтобы значительно снизить его производительность (например, путем генерирования непрерывного потока запросов, направляемых компьютеру). Эти атаки могут досаждать пользователю компьютера и могут приводить к потере данных, повреждению данных, копированию с компьютера конфиденциальных данных или к потере работоспособности компьютера.

Чтобы минимизировать действенность таких атак, были разработаны различные защитные программы и услуги. Эти программы и услуги выполняются в компьютерной системе и защищают компьютерную систему от злонамеренных атак. Такие программы могут включать в себя, например, антивирусные программы и программы брандмауэра. Целью этих программ или услуг обычно является защита от атак определенного типа. Например, антивирусная программа защищает от загрузки и/или выполнения компьютерных вирусов, а программа брандмауэра защищает от несанкционированного доступа к компьютеру со стороны внешнего пользователя.

Эти различные программы обычно не имеют связи друг с другом. Например, антивирусная программа обычно не сообщает программе брандмауэра об обнаружении вируса. Таким образом, различные защитные программы в компьютерной системе могут не узнавать об определенных атаках в компьютерной системе. Было бы желательно обеспечить интерфейс, который допускает передачу политик безопасности и информации о событиях среди различных компонентов и защитных программ в компьютерной системе.

Раскрытие изобретения

Описанные здесь системы и способы обеспечивают интерфейс, связанный с обработкой событий и другой информации для повышения безопасности вычислительной системы. Согласно конкретному варианту осуществления программный интерфейс содержит первую группу функций, относящуюся к передаче новой политики безопасности множественным подсистемам безопасности. Каждая из множественных подсистем безопасности реализована с возможностью замены существующей политики безопасности новой политикой безопасности. Программный интерфейс также содержит вторую группу функций, связанных с передачей указания о

готовности каждой подсистемы безопасности к реализации новой политики безопасности.

Краткое описание чертежей

Аналогичные компоненты и/или признаки на чертежах обозначаются аналогичными позициями.

Фиг.1 - схема иллюстративной среды, в которой генерируются и обрабатываются различные события.

Фиг.2 - иллюстрация политики безопасности, содержащей данные и правила.

Фиг.3 - иллюстративная таблица, поддерживаемая модулем безопасности, в которой приведены данные, запрашиваемые различными подсистемами безопасности.

Фиг.4 - блок-схема варианта осуществления процедуры извлечения и распространения правил и данных политики безопасности.

Фиг.5 - блок-схема варианта осуществления процедуры применения обновленных данных политики безопасности.

Фиг.6 - блок-схема варианта осуществления процедуры применения распространения информации к одной или нескольким подсистемам безопасности.

Фиг.7 - блок-схема варианта осуществления процедуры обновления политики безопасности.

Фиг.8 - блок-схема другого варианта осуществления процедуры обновления политики безопасности.

Фиг.9 - схема вычислительной среды общего вида.

Фиг.10-21 иллюстрируют различные варианты осуществления программного интерфейса.

Осуществление изобретения

Рассмотренные здесь системы и способы предусматривают обработку различной информации, например события, генерируемые одной или несколькими программами или услугами. Кроме того, описан интерфейс, допускающий передачу информации, например информации, связанной с безопасностью, помимо других компонентов и программ в вычислительной системе. Вычислительная система содержит менеджер событий, который принимает события и другую информацию из множественных источников, например, подсистем безопасности и других вычислительных систем. Примерами подсистем безопасности служат антивирусные подсистемы, подсистемы брандмауэра и подсистемы обнаружения проникновения. Менеджер событий передает информацию событий, полученную от конкретного источника, одной или нескольким подсистемам безопасности, которые могут использовать эту информацию для повышения уровня безопасности вычислительной системы.

Хотя рассмотренные здесь конкретные примеры относятся к событиям, связанным с безопасностью, и другой информации, связанной с безопасностью, альтернативные варианты осуществления могут обрабатывать события или информацию любого типа. Эта информация включает в себя любую информацию, которую могут использовать компоненты, связанные с безопасностью, в хост-компьютере. Альтернативные варианты осуществления предусматривают прием, обработку и распространение информации, которая не обязательно относится к безопасности хост-компьютера. Термины «интерфейс», «программный интерфейс» и «программный интерфейс приложения» (API) используются здесь на равных основаниях.

Обработка событий

На фиг.1 показана иллюстративная среда 100, в которой генерируются и обрабатываются различные события. События включают в себя, например,

обнаружение компьютерных вирусов, обнаружение попытки доступа к конфиденциальным данным, извещение об уничтожении компьютерного вируса, извещение об остановке или запрете выполнения той или иной прикладной программы, изменения информации о состоянии системы и т.д. Хост-компьютер 102 подключен к множественным серверам 104 и 106 посредством сети 108. Хост-компьютер 102 и серверы 104 и 106 могут представлять собой вычислительные устройства любого типа, например вычислительное устройство, рассмотренное ниже со ссылкой на фиг.9. Сеть 108 может представлять собой сеть передачи данных любого типа, например локальную сеть (ЛС), глобальную сеть (ГС), Интернет и т.п. Хотя на фиг.1 показано, что хост-компьютер подключен к двум серверам 104 и 106, хост-компьютер 102 может быть подключен к любому количеству серверов или других устройств, способных связываться с хост-компьютером.

Среда 100 может представлять собой любую из различных установок, например сети дома, на предприятиях, в образовательных, научно-исследовательских учреждениях и т.д. Например, сервер 104 может представлять собой серверное устройство в корпоративной ЛС, а хост-компьютер 102 может представлять собой настольное или портативное вычислительное устройство в корпоративной ЛС. В другом примере сервер 104 может представлять собой вычислительное устройство в Интернете, а хост-компьютер 102 может представлять собой настольное вычислительное устройство в доме пользователя.

Хост-компьютер 102 содержит модуль 110 безопасности, который осуществляет различные функции, относящиеся к безопасности, например мониторинг, обнаружение и реагирование на атаки на хост-компьютер 102. Модуль 110 безопасности содержит менеджер 112 событий, подключенный к трем подсистемам 114, 116 и 118 безопасности. Подсистема безопасности может представлять собой любую услугу, которая помогает защититься от злонамеренных пользователей и/или вредоносных программ. Подсистемы 114-118 безопасности могут быть реализованы программными аппаратными средствами или теми и другими в совокупности. Конкретные подсистемы безопасности представляют собой прикладные программы, связанные с безопасностью, например антивирусные программы и программы обнаружения проникновения. Подсистемы 114-118 безопасности также можно именовать «услугами». Конкретный модуль 110 безопасности может включать в себя любое количество подсистем безопасности, подключенных к менеджеру 112 безопасности. Модуль 110 безопасности может также включать в себя другие модули, компоненты или прикладные программы (не показаны), например считыватель политики, связанный с безопасностью, или иной механизм применения политики.

Модуль 110 безопасности также имеет доступ к информации 120 состояния системы и информации 122 конфигурации системы. Информация 120 состояния системы содержит информацию, относящуюся к текущему рабочему состоянию и/или режиму работы хост-компьютера 102. Информация 122 конфигурации системы содержит информацию, относящуюся к конфигурации хост-компьютера 102. Информация 120 состояния системы и информация 122 конфигурации системы может храниться в энергонезависимом запоминающем устройстве, например устройстве памяти или на жестком диске. Согласно одному варианту осуществления менеджер 112 безопасности и подсистемы 114-118 безопасности способны получать информацию 120 состояния системы и информацию 122 конфигурации системы.

Хост-компьютер 102 также содержит программный интерфейс приложений (API) 124, допускающий передачу политик безопасности и информацию событий среди

различных компонентов и программ на хост-компьютере 102 или других устройствах. Например, API 124 позволяет компонентам или программам связываться с подсистемами 114-118 безопасности или менеджером 112 безопасности, чтобы передавать или принимать информацию, связанную с безопасности. API 124 также помогает, например, загружать новые подсистемы безопасности, выгружать существующие подсистемы безопасности, посылать политики безопасности в подсистемы безопасности, передавать изменения в данных подсистемам безопасности, пользовательскому взаимодействию с подсистемами безопасности и централизованному управлению конфигурацией подсистем безопасности. Ниже рассмотрены дополнительные подробности, относящиеся к API 124.

Хотя это и не показано на фиг.1, дополнительные источники данных или поставщики данных могут передавать информацию и события на модуль 110 безопасности и менеджер 112 событий. Эти дополнительные данные включают в себя, например, информацию конфигурации, относящуюся к «Информационной службе Интернета» (США), данные, предоставляемые приложением управления системой, данные, содержащиеся в системном реестре, и информацию, предоставляемую пользователем или администратором системы.

Каждая подсистема 114-118 безопасности осуществляет определенные функции, связанные с безопасностью, способствующие защите хост-компьютера 102 от злонамеренных пользователей или прикладных программ. Эти злонамеренные пользователи или прикладные программы могут пытаться заблокировать хост-компьютер 102 или заблокировать функции хост-компьютера 102, получить данные от хост-компьютера 102 (например, пароли или другую конфиденциальную информацию) или использовать хост-компьютер 102 (например, для помощи в атаке на другие компьютерные системы). Например, подсистема 114 безопасности обнаруживает компьютерные вирусы, подсистема 116 безопасности обеспечивает защиту посредством брандмауэра, и подсистема 118 безопасности блокирует выполнение конкретных прикладных программ на основании одной или нескольких привилегий или характеристик пользователей. В этом примере подсистема 114 безопасности защищает хост-компьютер 102 от заражения компьютерными вирусами, червями, троянскими программами и т.п. Кроме того, защита посредством брандмауэра включает в себя защиту хост-компьютера 102 от доступа через сетевое соединение со стороны других устройств. Блокирование выполнения конкретных прикладных программ включает в себя препятствование выполнению прикладных программ на хост-компьютере 102 пользователем, который не имеет соответствующих привилегий. Дополнительно, выполнение прикладной программы можно блокировать в случае обнаружения неправильного поведения, например неправильного сетевого доступа или неправильного доступа к запоминающему устройству.

Согласно другим вариантам осуществления одна или несколько подсистем безопасности может осуществлять обнаружение проникновения или анализ уязвимости. Обнаружение проникновения включает в себя, например, выявление доступа злонамеренной прикладной программы и/или пользователя к хост-компьютеру 102 и совершение соответствующего действия по извещению пользователя или администратора, попытку заблокировать вредоносную прикладную программу или прекратить доступ злонамеренного пользователя. Анализ уязвимости включает в себя, например, попытку обнаружения слабых мест в хост-компьютере 102, обусловленных подсистемами безопасности или другими компонентами, которые

были неправильно установлены или обновлены, подсистемами безопасности или другими компонентами, которые были неправильно настроены, исправлениями или оперативными коррекциями, которые не были установлены, паролями, которые не соответствуют нужным длинам или нужным символам, и т.п. Конкретной подсистеме 114-118 безопасности может быть не известно о существовании и функциональных возможностях других подсистем безопасности, подключенных к менеджеру 112 событий.

Каждая подсистема безопасности 114-118 передает события (например, обнаружение компьютерного вируса, обнаружение попытки извлечения данных из хост-компьютера 102 или предотвращение выполнения пользователем прикладной программы) менеджеру 112 событий. Эти события включают в себя информацию, собранную подсистемой безопасности, действия, предпринятые подсистемой безопасности, данные, собранные менеджером безопасности из одного или нескольких источников данных, и т.п. Примером информации является список всех виртуальных серверов, заданных в конкретной установке. Менеджер 112 событий обрабатывает эти события и передает информацию, содержащуюся в конкретных событиях, другим подсистемам 114-118 безопасности, которым эта информация может быть полезна.

Модуль 110 безопасности также принимает политики, связанные с безопасностью, которые содержат одно или несколько правил и различные данные. Менеджер 112 событий распространяет правила на соответствующие подсистемы безопасности 114-118 и, по мере необходимости, предоставляет данные подсистемам безопасности. Каждая подсистема безопасности 114-118 сохраняет эти правила и данные, полученные от менеджера 112 безопасности. Ниже более подробно рассмотрена работа модуля 110 безопасности, менеджера 112 событий и подсистем 114-118 безопасности.

На фиг.2 показана иллюстративная политика безопасности 200, содержащая данные и правила. Согласно одному варианту осуществления политика безопасности 200 хранится в модуле 110 безопасности. Конкретный модуль безопасности может принимать и сохранять любое количество различных политик безопасности 200, полученных от любого количества различных источников данных. Альтернативно, политика безопасности 200 может храниться в другом модуле или компоненте хост-компьютера 102. В примере, приведенном на фиг.2, раздел 202 данных политики безопасности 200 содержит один или несколько элементов данных. Согласно фиг.2, эти элементы данных включают в себя значения, присвоенные переменным (например, значение «1» присвоено переменной «A», и значение «4» присвоено переменной «B»). Согласно альтернативным вариантам осуществления, другие типы данных могут быть включены вместо данных, показанных на фиг.2, или в добавление к ним. Данные, содержащиеся в политике безопасности 200, используются, например, одним или несколькими правилами, содержащимися в политике безопасности 200 или содержащимися в одной или нескольких других политик безопасности.

Политика безопасности 200 также содержит раздел 204 правил, содержащий множественные правила. Правила в политике безопасности 200 могут быть связаны с одной или несколькими подсистемами безопасности. Например, определенные правила могут применяться только конкретными подсистемами безопасности. Правила могут размещаться в политике безопасности 200 в зависимости от того, с какой подсистемой безопасности связаны эти правила. Альтернативно, идентификатор, связанный с каждым правилом, может указывать подсистемы

безопасности, способные применять правило. В конкретных вариантах осуществления правило может быть связано с любым количеством подсистем безопасности. В других вариантах осуществления хост-компьютер может не содержать подсистему безопасности, которая применяет конкретное правило. В этом случае правило не связано ни с какой подсистемой безопасности.

В примере, показанном на фиг.2, правила определены с использованием конструкции «ЕСЛИ-ТО». Альтернативно, набор правил может принимать различные формы. С использованием конструкции ЕСЛИ-ТО, показанной на фиг.2, правило определяет конкретное(ые) условие(я) и соответствующее(ие) действие(ия) или результат(ы). В ходе применения правила при обнаружении данного конкретного условия выполняются определенные действия или получаются определенные результаты. Правило может указывать различные условия и соответствующие действия или результаты. Примеры условий включают в себя попытки доступа к ресурсу (например, ячейкам памяти, сетевым адресам или портам, другим программам или файлам, хранящимся в запоминающем устройстве), попытки записи данных в конкретные места (например, конкретные ячейки памяти или конкретные места в запоминающем устройстве), попытки запуска определенных программ и различные аспекты текущего рабочего состояния хост-компьютера 102. Примеры результатов включают в себя препятствование доступу к ресурсу, препятствование записи данных в определенные места, препятствование выполнению программы или генерацию извещения об обнаружении наступления условия в правиле (например, регистрацию этого наступления в журнале или отправку сообщения пользователю или другому компьютеру). Конкретные результаты могут также быть разрешительными, а не только запретительными. Например, результаты могут указывать, что доступ к конкретному ресурсу или месту разрешается только, если хост-компьютер 102 удовлетворяет условию в правиле, или что запуск конкретной программы разрешается только, если хост-компьютер 102 удовлетворяет условию в правиле.

Дополнительные примеры правил включают в себя разрешение определенным прикладным программам или услугам обновлять файлы данных в конкретной директории или папке, разрешение приема трафика на порте 21, если действует протокол передачи файлов (FTP), и генерацию предупреждающего сообщения о вирусе, в случае обнаружения сигнатуры того или иного вируса. Другие примеры включают в себя генерацию события, если конкретная прикладная программа не была обновлена до конкретного уровня версии, препятствование доступу в сеть, если прикладная программа не была обновлена до минимального уровня версии, и препятствование хост-компьютеру в получении данных через сетевой порт 35.

На фиг.3 представлена таблица 300, поддерживаемая модулем безопасности, относящаяся к данным, запрашиваемым различными подсистемами безопасности. Согласно одному варианту осуществления таблица 300 может храниться в модуле 110 безопасности. Альтернативно, таблица 300 может храниться в другом модуле или компоненте хост-компьютера 102. Всякий раз, когда подсистема безопасности запрашивает данные у модуля безопасности, модуль безопасности обновляет таблицу (если необходимо) для включения этого запроса данных. Первый столбец 302 таблицы 300 указывает конкретный элемент данных, например переменную или другой идентификатор или информацию. Второй столбец 304 таблицы 300 указывает любые подсистемы безопасности, которые раньше запрашивали соответствующий элемент данных. Например, таблица 300 указывает, что элемент данных «A» ранее запрашивался подсистемой безопасности «1». Аналогично, элемент данных «D» ранее

запрашивался подсистемами безопасности «1», «4» и «6». Согласно рассмотренному более подробно ниже информация, содержащаяся в таблице 300, используется модулем безопасности для определения, какие подсистемы безопасности должны получать обновленные данные.

На фиг.4 показана блок-схема варианта осуществления процедуры 400 извлечения и распространения правил и данных политики безопасности. Процедура 400 может осуществляться, например, после инициализации хост-компьютера. Сначала модуль безопасности извлекает политики безопасности для хост-компьютера (блок 402). Менеджер событий идентифицирует правила в политиках безопасности, относящиеся к каждой подсистеме безопасности (блок 404). Затем менеджер событий передает правила в соответствующие подсистемы безопасности (блок 406).

Каждая подсистема безопасности идентифицирует данные, необходимые для применения связанных с нею правил (блок 408), например, путем идентификации элементов данных, содержащихся в правилах, которые будет применять подсистема безопасности. Затем каждая подсистема безопасности запрашивает идентифицированные ею данные у менеджера событий (блок 410). Получив запрос данных от подсистемы безопасности, менеджер событий заносит запрашиваемые данные в таблицу (например, таблицу 300 на фиг.3) или в другую структуру данных для последующего обращения (блок 412). Наконец, менеджер событий определяет местоположение запрашиваемых данных и предоставляет эти данные подсистеме безопасности (блок 414). Таким образом, вместо того, чтобы предоставлять все данные всем подсистемам безопасности, менеджер событий предоставляет запрашиваемые данные запрашивающей подсистеме безопасности.

На фиг.5 показан вариант осуществления процедуры 500 применения обновленных данных политики безопасности. Сначала модуль безопасности извлекает обновленные данные (блок 502). Например, обновленные данные могут включать в себя обновленные значения существующих переменных. Модуль безопасности идентифицирует одну или несколько подсистем безопасности, которые раньше запрашивали данные, которые были обновлены (блок 504). Согласно одному варианту осуществления модуль безопасности идентифицирует эти подсистемы безопасности с помощью таблицы, например таблицы 300, показанной на фиг.3. Идентифицировав соответствующие подсистемы безопасности, модуль безопасности предоставляет обновленные данные каждой идентифицированной подсистеме безопасности (блок 506). Наконец, подсистемы безопасности обновляют свои элементы данных обновленными данными. Процедура 500 повторяется каждый раз, когда модуль безопасности принимает обновленные данные. Если данные были обновлены, то модуль безопасности извлекает обновленные данные и распространяет данные согласно процедуре 500.

Согласно одному варианту осуществления при обновлении правила модуль безопасности идентифицирует подсистемы безопасности, связанные с правилом, и распространяет обновленное правило на идентифицированные подсистемы безопасности. Получив новое правило, модуль безопасности идентифицирует подсистемы безопасности, которые могут использовать новое правило, и распространяет новое правило на соответствующие подсистемы безопасности. Аналогично, обнаружив существующее правило, модуль безопасности удаляет правило из всех подсистем безопасности, связанных с правилом. Согласно другому варианту осуществления при обновлении правила модуль безопасности создает новый набор правил (содержащий обновленное правило) и распространяет новый набор

правил на подсистемы безопасности, тем самым заменяя существующие правила, содержащиеся в подсистемах безопасности.

На фиг.6 показана блок-схема варианта осуществления процедуры 600 применения распространения информации, например информации событий или информации состояния системы, на одну или несколько подсистем безопасности. Сначала менеджер событий принимает событие от подсистемы безопасности (блок 602). Затем менеджер событий идентифицирует информацию, содержащуюся в событии (блок 604), например тип события или характер атаки, генерировавшей событие. Менеджер событий также идентифицирует другие подсистемы безопасности, которые могут использовать информацию, содержащуюся в событии (блок 606). Взаимоотношения между различными подсистемами безопасности заданы, например, в политике безопасности, полученной хост-компьютером. Эти взаимоотношения могут быть заданы полностью или частично системным администратором или системным оператором при создании политики безопасности.

Затем менеджер событий предоставляет информацию, содержащуюся в событии, идентифицированным подсистемам безопасности (блок 608). Затем идентифицированные подсистемы безопасности применяют полученную информацию (блок 610). Это совместное использование (или сопоставление) информации событий повышает уровень безопасности, обеспечиваемый хост-компьютером, против злонамеренных атак. Совместным использованием информации событий распоряжается менеджер событий, в результате чего отдельным подсистемам безопасности не требуется знать о других подсистемах безопасности, содержащихся в хост-компьютере. Рассматриваемая здесь информация, связанная с безопасностью, может храниться в определенном месте, что позволяет другим устройствам, компонентам и прикладным программам осуществлять доступ к информации. Например, другие подсистемы безопасности или вычислительные системы могут осуществлять доступ к сохраненной информации, относящейся к безопасности.

В одном примере процедуры 600 антивирусная подсистема безопасности обнаруживает повторяющиеся попытки доступа к сети через конкретный порт.Антивирусная подсистема безопасности сообщает эту информацию (например, даты и времена попыток доступа и порт, на котором производятся попытки доступа) менеджеру событий. В этом примере антивирусная подсистема безопасности не отвечает за реакцию на такие попытки доступа. Менеджер событий принимает информацию от антивирусной подсистемы безопасности и определяет, что подсистема безопасности для обнаружения проникновения и подсистема безопасности для брандмауэра могут использовать такую информацию. Получив информацию, подсистема безопасности для обнаружения проникновения и подсистема безопасности для брандмауэра могут отрегулировать свою работу на основании полученной информации. Например, подсистема безопасности для обнаружения проникновения может повысить частоту, с которой она проверяет наличие проникающих. Кроме того, подсистема безопасности для брандмауэра может временно заблокировать порт, на котором была попытка доступа. Таким образом, общая безопасность хост-компьютера против атак повышается благодаря тому, что подсистемы безопасности могут регулировать свою работу на основании совместно используемой информации, касающейся событий, относящейся к безопасности.

В другом примере процедуры 600 подсистема безопасности уязвимости обнаруживает, установлено ли конкретное исправление на хост-компьютере. Если исправление не установлено, то подсистема безопасности, связанная с уязвимостями,

генерирует событие, указывающее, что исправление не установлено. Подсистема безопасности брандмауэра хоста и подсистема безопасности поведенческой блокировки зарегистрировались с помощью менеджера событий для извещения, если исправление не установлено. Когда подсистема безопасности брандмауэра хоста и подсистема безопасности поведенческой блокировки получают извещение о том, что исправление не установлено, подсистемы безопасности применяют правила, ограничивающие функциональные возможности (или препятствующие выполнению) прикладной программы, в которой не было установлено исправление.

Согласно еще одному варианту осуществления информация состояния совместно используется различными компонентами (например, менеджером событий и множественными подсистемами безопасности) в модуле безопасности. Информация состояния системы может обеспечиваться различными источниками данных. Иллюстративная информация состояния системы включает в себя текущее состояние сети, является ли сетевое соединение проводным или беспроводным, осуществляет ли хост-компьютер доступ к корпоративной сети или к неизвестной сети, и информация конфигурации хост-компьтера. Таким образом, если подсистема безопасности идентифицирует конкретную информацию состояния системы, то эта идентифицированная информация может совместно использоваться другими подсистемами безопасности и другими компонентами или модулями в хост-компьютере.

В конкретном варианте осуществления информация состояния системы, собранная различными компонентами, хранится в центральном пункте, что обеспечивает доступ к информации со стороны других устройств, компонентов и прикладных программ. Например, информация состояния системы, собранная одной подсистемой безопасности, доступна другим подсистемам безопасности, модулям безопасности и вычислительным системам.

Обновления политики безопасности

Согласно рассмотренному выше политику безопасности можно использовать для описания правил, подлежащих применению, например, подсистемами безопасности или провайдерами услуг безопасности. При внесении изменений в политику безопасности обновленные правила направляются различным подсистемам безопасности, и эти различные подсистемы безопасности перенастраиваются для работы с использованием обновленных правил, по существу, в одно и то же время.

Помимо компонентов, программ и модулей, рассмотренных выше в отношении фиг.1, хост-компьютер 102 принимает политики безопасности от одного или нескольких устройств-источников, например серверов 104 и 106. Политики безопасности описывают, как должны действовать различные подсистемы безопасности на хост-компьютере 102. Хотя на фиг.1 показан только один хост-компьютер 102, очевидно, что множественные хост-компьютеры 102 могут получать политики безопасности от одного и того же устройства-источника.

Иллюстративные устройства-источники включают в себя вычислительные устройства настольного типа или типа рабочей станции, серверные вычислительные устройства, портативные или переносные вычислительные устройства, игровые приставки, сетевые приборы, сотовые телефоны, карманные персональные компьютеры (КПК), сетевые устройства (например, маршрутизаторы, шлюзы, брандмауэры, точки беспроводного доступа и т.д.) и т.д.

Хост-компьютер 102 может также включать в себя модуль считывания политики, модуль генератора набора правил и хранилище динамических данных для правил.

Следует понимать, что один или несколько таких модулей могут быть объединены в единый модуль и/или один или несколько этих модулей могут быть разделены на два или более модулей.

В общем случае, для обновления политики безопасности, применяемой подсистемами безопасности 114-118, считыватель политики получает политику безопасности от устройства-источника. Генератор набора правил использует вновь полученную политику безопасности, чтобы генерировать для каждой из различных подсистем безопасности набор из одного или нескольких правил и соответствующих данных. Эти наборы и правила передаются различным подсистемам безопасности, а соответствующие данные сохраняются в хранилище динамических данных для правил. Соответствующие данные могут также передаваться подсистемам безопасности. Получив набор из одного или нескольких правил, каждая подсистема безопасности обрабатывает новый набор правил, готовясь начать использовать новый набор правил. Однако каждая подсистема безопасности продолжает использовать свой текущий набор правил, пока не получит команду перейти к новому набору. Менеджер правил предписывает всем подсистемам безопасности перейти к новому набору правил после того, как менеджер правил получит указание от каждой из подсистем безопасности, что она готова перейти к новому набору правил.

В некоторых вариантах осуществления менеджер правил координирует обновление политик безопасности в хост-компьютере 102. Менеджер правил получает указания от различных подсистем безопасности, которые указывают готовность подсистем безопасности к переходу к новому набору правил, и дает указание подсистемам безопасности, когда они должны начать использовать новый набор правил.

Модуль считывания политики получает новую политику безопасности от устройства-источника. Модуль считывания политики может быть настроен на проверку наличия у источника новой политики безопасности с регулярными или нерегулярными интервалами или, альтернативно, может принимать указание от некоторого другого компонента (например, менеджера правил, устройства-источника или какого-либо другого источника, не показанного на фиг.1), что ему следует получить новую политику безопасности из источника (или проверить наличие у источника новой политики безопасности). Считыватель политики может указывать источнику конкретную политику безопасности, которую считыватель политики желает получить, или, альтернативно, может лишь запрашивать у источника самую последнюю политику безопасности для хост-компьютера. Сравнение между текущей политикой безопасности, используемой хост-компьютером, и самой последней политикой безопасности может производиться с целью определения, применяется ли уже на хост-компьютере самая последняя политика безопасности. Такое сравнение может производиться источником, считывателем политики или, альтернативно, каким-либо другим компонентом.

Когда от источника получена новая политика безопасности, генератор набора правил генерирует набор правил для каждой из различных подсистем безопасности 114-118. Различные подсистемы безопасности могут использовать разные правила, применяя политику безопасности на хост-компьютере 102. Например, одна подсистема безопасности 114 может представлять собой брандмауэр, а другая подсистема безопасности 116 может быть антивирусным компонентом. Политика безопасности может указывать правила, относящиеся к антивирусной подсистеме (и, таким образом, подсистеме брандмауэра не нужно к ним обращаться), и может также указывать правила, относящиеся к подсистеме брандмауэра (и, таким образом,

антивирусной подсистеме не нужно к ним обращаться).

В некоторых вариантах осуществления, политика безопасности сама по себе является списком правил и соответствующих данных. Политика безопасности может также включать в себя указание, какие правила и данные относятся к каким подсистемам безопасности, или, альтернативно, не включать в себя подобных указаний (например, предоставляя хост-компьютеру право определять, какие правила относятся к каким подсистемам безопасности). Политика безопасности позволяет разработчикам иметь единую запись или файл всех правил, используемых для защиты хост-компьютера, без необходимости распределять правила по разным записям или файлам для разных подсистем безопасности.

Кроме того, используя описанные здесь подходы, разработчики могут подготавливать новые политики, позволяющие передавать ответственность за защиту от конкретных атак с одной подсистемы безопасности на другую. Например, в рамках одной политики безопасности, защиту от конкретного типа атаки может осуществлять антивирусная программа, но в новой политике безопасности эта обязанность переходит к программе брандмауэра. Используя описанные здесь подходы, разработчики могут быть уверены, что эта передача ответственности будет происходить практически одновременно, что снижает уязвимость хост-компьютера к атакам в ходе передачи.

Генератор набора правил указывает на основании политики безопасности, какие правила и соответствующие данные (если таковые имеются) используются какими подсистемами безопасности. Для каждой подсистемы безопасности генератор набора правил генерирует набор правил для этой подсистемы безопасности и делает этот генерированный набор правил доступным этой подсистеме безопасности (например, набор правил можно передавать или отправлять подсистеме безопасности, подсистему безопасности можно информировать о том, в каком месте памяти можно получить генерированный набор правил, и т.д.). Эту генерацию можно производить разными способами. Например, генератор набора правил может генерировать новый набор правил безотносительно к текущим правилам, применяемым подсистемами безопасности. В другом примере текущий набор правил можно модифицировать или изменять, включая какие-либо различия между текущим и новым наборами правил. Кроме того, генератор набора правил может просто копировать правила из политик безопасности или, альтернативно, генератор набора правил может генерировать правила на основании информации в политике безопасности, где описаны правила.

В некоторых вариантах осуществления политика безопасности указывает, какие правила подлежат распространению на какие подсистемы безопасности. Например, каждое правило можно связать с конкретным ярлыком или идентификатором (например, «подсистема безопасности 1» или «антивирусная подсистема» и т.п.). Генератор набора правил может использовать эти идентификаторы при генерации наборов правил для различных подсистем безопасности. В альтернативных вариантах осуществления генератор набора правил может определять, какие правила нужно распространять на какие подсистемы безопасности. В других вариантах осуществления можно использовать комбинацию этих подходов (например, для некоторых правил, политика безопасности может указывать, какой подсистеме безопасности они должны быть назначены, а для других правил генератор политики безопасности может определять, какой подсистеме безопасности они должны быть назначены).

Набор правил, генерируемый генератором набора правил, может принимать

различные формы. В некоторых вариантах осуществления правила заданы конструкцией «если-то», как описано выше. С использованием этой структуры правила определяют конкретное(ые) условие(я) и соответствующее(ие) конкретное(ые) действие(я) или результат(ы). В ходе применения правила при обнаружении конкретного(ых) условия(ий) выполняется(ются) соответствующее(ие) конкретное(ые) действие(я) или результат(ы). Правило может указывать любые из разнообразных условий и соответствующих результатов. Примеры конкретных условий включают в себя: попытки доступа к конкретным ресурсам (например, ячейкам памяти, сетевым адресам или портам, другим программам, файлам в запоминающем устройстве и т.п.), попытки записи данных в конкретных местах (например, в конкретных ячейках памяти, в конкретных местах запоминающего устройства и т.п.), попытки запуска конкретных программ, различные аспекты текущего рабочего состояния хост-компьютера (например, имеющиеся ресурсы, действующие программы и т.п.) и т.д. Примеры результатов включают в себя препятствование доступу к ресурсу, препятствование записи данных в определенные места, препятствование выполнению программы или генерацию извещения об обнаружении наступления условия по данному правилу (например, регистрацию этого наступления в журнале или отправку сообщения пользователю или другому компьютеру). Конкретные результаты могут также быть разрешительными, а не только запретительными. Например, результаты могут указывать, что доступ к конкретному ресурсу или месту разрешается только, если хост-компьютер удовлетворяет условию в правиле, или что запуск конкретной программы разрешается только, если хост-компьютер удовлетворяет условию в правиле.

В определенных вариантах осуществления хост-компьютер 102 содержит хранилище динамических данных для правил, которое представляет собой данные, связанные с различными правилами, применяемыми подсистемами безопасности. В некоторых вариантах осуществления хранилище динамических данных для правил может включать в себя два набора данных: один набор для текущих правил, применяемых подсистемами безопасности, и другой набор - для новых правил, для применения которых подсистемы безопасности обновляются. При получении новой политики безопасности генератор набора правил обновляет хранилище динамических данных для правил данными, связанными с наборами новых правил, переданных подсистемам безопасности.

Каждая подсистема безопасности содержит модуль изменения правил, который получает набор из одного или нескольких правил от генератора набора правил. Данные, связанные с правилами, могут поступать от генератора набора правил совместно с правилами или, альтернативно, модуль изменения правил может получать данные, по желанию, из динамических данных для правил. Кроме того, заметим, что хотя согласно рассмотренному выше, генератор набора правил генерирует набор правил для каждой подсистемы безопасности на основании политики безопасности, альтернативно, каждая подсистема безопасности может получать всю политику безопасности (или большую часть политики безопасности) и генерировать свой собственный набор правил вместо того, чтобы принимать набор от генератора набора правил.

Модуль изменения правил обрабатывает новый набор правил по мере необходимости для генерации новых внутренних правил, которые соответствуют новой политике. Обработка нового набора правил для генерации новых внутренних правил означает любые действия, которые должна выполнять подсистема

безопасности, чтобы привести новый набор правил к виду, в котором они смогут выполняться подсистемой безопасности. Например, эта обработка может включать в себя преобразование нового набора правил во внутренний формат, сохранение правил в конкретных ячейках памяти, организацию правил в конкретное размещение или конкретный порядок и т.д. Модуль изменения правил может генерировать новые правила любым из разнообразных способов; модуль изменения правил может оставить правила в том же формате, в каком они были получены от генератора набора правил или, альтернативно, преобразовать правила во внутренний формат, используемый подсистемой безопасности.

Независимо от того, как генерируются новые правила, каждая подсистема безопасности поддерживает текущий набор правил, которые реализуют предыдущую политику безопасности для хост-компьютера (политику безопасности, которая подлежит обновлению). При генерации новых правил и даже после того, как новые правила сгенерированы, подсистема безопасности продолжает применять текущие правила. Подсистема безопасности не начинает применять новые правила, пока не получит предписание делать это (например, со стороны менеджера правил).

Закончив генерировать новые правила, модуль изменения правил указывает менеджеру правил, что он закончил и готов переключиться на использование новых правил (и, таким образом, начать применять новую политику безопасности). Получив такое указание от всех подсистем безопасности, менеджер правил предписывает каждой подсистеме безопасности начать использовать новые правила. Менеджер правил ждет, чтобы предписать каждой из подсистем безопасности начать использовать новые правила, пока не получит указание от всех подсистем безопасности. Получив предписание делать это, каждая подсистема безопасности начинает использовать новые правила. Как только подсистема безопасности начинает использовать новые правила, она может удалить ранее использовавшиеся правила.

В некоторых случаях подсистеме безопасности может не удаться обработать новые правила. В этом случае, подсистема безопасности возвращает указание о такой неудаче менеджеру правил. Альтернативно, менеджер правил может установить временной предел на ответы подсистем безопасности. Если все подсистемы безопасности не ответили указанием готовности к началу использования новых правил до истечения временного предела, то менеджер правил может предположить, что одной или нескольким подсистемам безопасности не удалось обработать новые правила.

Определив, что одной или нескольким подсистемам безопасности не удалось обработать новые правила, менеджер правил не предписывает ни одной из подсистем безопасности начать использовать новые правила. Вместо этого менеджер правил посылает указание прервать переход к новым правилам (это также можно называть откатом). Такое указание прерывания или отката информирует каждую из подсистем безопасности, что нужно игнорировать новые правила, полученные от генератора набора правил, а также любые новые правила, полученные в результате из обработки, и продолжать использовать текущие правила. В некоторых вариантах осуществления, в ответ на такое указание прерывания или отката, подсистемы безопасности могут безопасно удалять новые правила, генерированные ими (или находящиеся на стадии генерации).

В некоторых вариантах осуществления каждая подсистема безопасности ожидает, пока не будет достаточно уверена в том, что она может начать использовать новые правила, прежде чем сообщить менеджеру правило своей готовности к началу

использования новых правил. Иными словами, подсистема безопасности задерживает информирование менеджера правил о своей готовности использовать новые правила, пока, в процессе обработки новых правил, не удостоверится, что, начав применять эти правила после получения предписания делать это, она практически наверняка не потерпит неудачу. В некоторых вариантах осуществления для этого подсистема безопасности генерирует новый набор правил и поддерживает указатель на то, какой из наборов правил (старые правила или новые правила) подлежит использованию. Сгенерировав новый набор правил, подсистема безопасности указывает менеджеру правил о своей готовности начать использовать новый набор правил. Затем, получив предписание начать использовать новый набор правил, подсистема безопасности может просто изменить свой указатель со старого набора правил на новый набор правил. Подсистема безопасности может быть почти уверена в том, что она может изменить свой указатель и начать использовать новые правила. Очевидно, что «почти уверена» не означает 100%-ной гарантии, что неудача абсолютна невозможна. Существует возможность возникновения некоторых обстоятельств, которые могут привести к неудаче (например, отключение питания или вирусная атака, которая запрещает смену указателя). Однако следует также понимать, что вероятность неудачи весьма мала.

Менеджер правил может предписать подсистемам безопасности начать использовать новый набор правил (что также можно называть переключением на новый набор правил) любым из совокупности разнообразных способов. Однако любой способ должен предусматривать практически одновременное информирование всех подсистем безопасности о том, что все подсистемы безопасности могут начать использовать свои новые наборы правил, по существу, в одно и то же время (что здесь также называется практически одновременно). Благодаря тому, что все подсистемы безопасности начинают использовать свои новые наборы правил, по существу, в одно и то же время, можно исключить любую уязвимость хост-компьютера, обусловленную сменой правил. В целом, чем меньше разброс по времени начала использования подсистемами безопасности своих новых наборов правил, тем меньше уязвимость при переходе к новому набору правил. Ниже приведены два примера способа, каким менеджер правил может предписывать подсистемам безопасности, по существу, в одно и то же время, начать использовать свои новые наборы правил.

Один способ, каким менеджер правил может предписывать подсистемам безопасности начать использовать новый набор правил, состоит в использовании объекта события, которое может генерироваться сразу для всех подсистем безопасности. Например, каждая подсистема безопасности, получив новые правила от генератора набора правил, устанавливает внутренний флаг, чтобы начинать опрашивать конкретное событие всякий раз при обращении к правилам (при нормальной работе по защите хост-компьютера). Затем менеджер правил может предписать подсистемам безопасности начать использовать свои новые наборы правил, инициируя событие (то же, которое опрашивают подсистемы безопасности). Поэтому после генерации события любой последующий опрос события будет отражать тот факт, что событие было генерировано, и тем самым информировать опрашивающую подсистему безопасности о необходимости использовать новый набор правил. Например, в случае обнаружения генерируемого события, указатель в подсистеме безопасности можно изменить, чтобы он указывал на новый набор правил.

Помимо опроса события, подсистема безопасности, ожидающая события, может также запустить поток. При генерации события поток обнаруживает генерацию,

вследствие чего подсистема безопасности узнает, что нужно использовать новый набор правил. Например, в результате обнаружения потоком генерации события, указатель в подсистеме безопасности может измениться, чтобы указывать на новый набор правил.

Когда событие сгенерировано и используется новый набор правил, подсистема безопасности может прекратить опрос события. Кроме того, если подсистема безопасности запустила поток, ожидающий событие, этот поток можно завершить.

Другой способ, каким менеджер правил может предписывать подсистемам безопасности начать использовать новый набор правил, состоит в вызове функции, объявленной каждой подсистеме безопасности (например, функции «переключение»). Вызов такой функции подсистемы безопасности предписывает этой подсистеме безопасности начать использовать новый набор правил. Например, в результате вызова такой функции в подсистеме безопасности подсистема безопасности меняет свой указатель, чтобы он указывал на новый набор правил.

Еще один способ, каким менеджер правил может предписывать подсистемам безопасности начать использовать новый набор правил, состоит в извещении каждой из подсистем безопасности о совместно используемой структуре данных, к которой может обращаться каждая подсистема безопасности. Менеджер правил может информировать каждую подсистему безопасности о совместно используемой структуре данных в разные моменты, например, вызывая в каждой подсистеме безопасности функцию (например, функцию «идентификация структуры данных») или идентифицируя совместно используемую структуру данных при передаче подсистеме безопасности новых правил. Совместно используемая структура данных может принимать любую из различных форм, например, ячейки памяти (например, оперативной памяти (ОЗУ) или энергонезависимой памяти, в частности флэш-памяти), файла, хранящегося в локальном или удаленном запоминающем устройстве, и т.п.

Каждая подсистема безопасности проверяет эту совместно используемую структуру данных (например, всякий раз при обращении к правилам (при нормальной работе по защите хост-компьютера)), чтобы определить ее значение. Менеджер правил может предписать каждой из подсистем безопасности начать использовать новый набор правил, изменив значение(я), хранящиеся в совместно используемой структуре данных. Например, в совместно используемой структуре данных может первоначально храниться значение «предыдущий» или значение 0, указывающее, что нужно использовать текущий набор правил, а когда приходит время переключения, чтобы начать использовать новый набор правил, менеджер правил может записать в совместно используемую структуру данных значение «новый» или «переключение» или значение 1, указывающее, что нужно использовать новый набор правил.

Согласно рассмотренному выше в хранилище динамических данных для правил хранятся данные, связанные с различными правилами, применяемыми подсистемами безопасности. При этом при обновлении хост-компьютера, чтобы начать применять новую политику, данные, используемые подсистемой безопасности, также могут меняться. Эти данные могут меняться в ходе работы хост-компьютера (например, подсистема безопасности может позже запросить данные из хранилища динамических данных для правил или записать их туда). Чтобы подсистемам безопасности были доступны правильные данные, при обновлении политики безопасности хранилище динамических данных для правил может действовать таким же образом, как и подсистема безопасности. Это значит, что должны поддерживаться два набора данных для правил: первый набор - для использования до переключения, и второй

набор - для использования после переключения. Новые данные будут сохраняться в хранилище динамических данных для правил, и хранилище динамических данных для правил будет возвращать менеджеру правил указание своей готовности начать использовать новый набор данных. Затем хранилище динамических данных для правил продолжает использовать предыдущий набор данных, пока не получит от менеджера правил предписание начать использовать новый набор данных.

Заметим, что различные компоненты хост-компьютера можно реализовать в одном и том же прикладном процессе, выполняющемся на хост-компьютере. Например, считыватель политики, менеджер правил, динамические данные для правил, генератор набора правил и подсистемы безопасности могут входить в состав одного и того же прикладного процесса.

Альтернативно, разные компоненты хост-компьютера можно реализовать в двух или более прикладных процессах, выполняющихся на хост-компьютере. Например, в процессе может действовать одна или несколько подсистем безопасности, отделенная от других подсистем безопасности, а также отделенная от считывателя политики, менеджера правил, динамических данных для правил и генератора набора правил. Возможность для различных компонентов действовать в разных прикладных процессах позволяет, например, разным разработчикам разрабатывать различные сменные компоненты (например, сменные подсистемы безопасности) для улучшения защиты хост-компьютера. Эти дополнительные сменные компоненты будут обновляться для применения новых политик таким же образом, как и другие, несменные, компоненты.

При разделении компонентов между множественными прикладными процессами используется механизм, позволяющий различным компонентам связываться друг с другом. Эта связь позволяет, например, передавать наборы новых правил и данных подсистемам безопасности в других процессах, передавать данные от подсистем безопасности в разных процессах в динамические данные для правил, передавать предписания начать использование новых наборов правил подсистемам безопасности в разных процессах и т.д. Например, рассмотренные здесь компоненты можно реализовать как компоненты модели компонентных объектов (COM). Дополнительную информацию по архитектуре модели компонентных объектов можно получить в фирме Майкрософт Корпорейшн, Редмонд, Вашингтон.

Заметим, что в настоящем описании, подсистеме безопасности предписывается начать использовать ее новый набор правил менеджером правил. Альтернативно, это предписание можно реализовать другими способами, которые тем не менее позволяют всем подсистемам безопасности начинать использовать их новые наборы правила практически одновременно. Например, вместо того, чтобы использовать менеджер правил, по различным подсистемам безопасности можно распределить механизм управления, предписывающий каждой подсистеме безопасности начать использовать ее новый набор правил. Для этого, например, каждая из подсистем безопасности может извещать все остальные подсистемы безопасности о своей готовности начать использование нового набора правил, при этом ни одна подсистема безопасности не должна начинать использовать свой новый набор правил, пока все подсистемы безопасности не известят все остальные подсистемы безопасности о своей готовности начать использование нового набора правил.

На фиг.7 показана логическая блок-схема, иллюстрирующая процесс 700 обновления политики безопасности. Процесс 700 реализуется компонентом(ами), координирующим обновление политики безопасности на хост-компьютере, например,

рассмотренным здесь менеджером правил. Процесс 700 может осуществляться программными, аппаратными, программно-аппаратными средствами или их комбинацией.

Сначала получают (блок 702) новую политику, подлежащую применению на устройстве. Политику можно получить «активным» способом, когда хост-компьютер инициирует доступ к источнику новой политики, чтобы проверить, доступна ли от источника новая политика. Альтернативно, политику можно получить «пассивным» способом, когда хост-компьютер информируется (посредством передачи сообщения или другого указания) о наличии новой политики безопасности или о самой новой политике безопасности.

Независимо от способа получения новой политики, получив новую политику, каждой из подсистем безопасности предоставляют новый набор правил и/или данных, связанных с правилами для новой политики (блок 704). Согласно рассмотренному выше можно генерировать разные наборы правил и/или данных в зависимости от новой политики для каждой подсистемы безопасности.

Затем принимают от подсистем безопасности (блок 706) возвращаемые значения. В некоторых вариантах осуществления такая подсистема безопасности возвращает в компонент, реализующий процесс 700, значение, обозначающее «успех» или значение, обозначающее «неудачу». Когда подсистема безопасности возвращает значение, обозначающее «успех», она указывает, что подсистема безопасности обработала набор правил и/или данных, который она получила, и готова начать использовать новый набор правил и/или данных. Это также можно рассматривать как готовность подсистемы безопасности реализовать новый набор правил и/или данных. Например, все, что остается сделать подсистеме безопасности, это изменить свой указатель, чтобы указать на новый набор правил вместо предыдущего набора правил. Однако, когда подсистема безопасности возвращает значение, обозначающее «неудачу», она указывает, что подсистема безопасности не может обработать (или не обработала) набор правил и/или данных и что подсистема безопасности не способна начать использовать новый набор правил и/или данных. Кроме того, согласно рассмотренному выше на ответы подсистем безопасности можно наложить временной предел (также именуемый значением таймаута или пороговым временем), если подсистема безопасности не отвечает значением, обозначающим «успех» или «неудачу» до этого временного предела, то компонент, реализующий процесс 700, обрабатывает подсистему безопасности, как если бы она возвратила значение, обозначающее «неудачу».

Очевидно, что отправка правил и получение ответов (блоки 704 и 706) являются асинхронными процессами. Разным подсистемам безопасности может потребоваться разное время для обработки правил и/или данных, которые они получают, и процесс 700 просто ждет, пока все подсистемы безопасности не закончат соответствующую обработку (вплоть до любого необязательного наложенного временного предела).

Процесс 700 разветвляется в зависимости от того, все ли подсистемы безопасности возвратили значение, обозначающее «успех» (блок 708). Если все подсистемы безопасности возвратили значение, обозначающее «успех», значит все подсистемы безопасности готовы начать использовать новый набор правил, поэтому всем подсистемам безопасности предписывается использовать новый набор правил (блок 710).

Если же, хотя бы одна из подсистем безопасности не возвращает значение,

обозначающее «успех», то в каждую подсистему безопасности поступает вызов отката (блок 712). Этот вызов отката, по существу, прерывает процесс обновления, поэтому ни одна из подсистем безопасности не начнет использовать новый набор правил (даже те подсистемы безопасности, которые возвратили значение, обозначающее «успех»).

На фиг.8 показана блок-схема, иллюстрирующая другой процесс 800 обновления политики безопасности. Процесс 800 реализуется подсистемой безопасности на устройстве, например, подсистемой безопасности 114-118 на хост-компьютере 102, показанном на фиг.1. Процесс 800 может осуществляться программными, аппаратными, программно-аппаратными средствами или их комбинациями.

Сначала получают новый набор правил и/или данных для новой политики, подлежащей применению (блок 802). Согласно рассмотренному выше правила или данные могут быть приняты практически одновременно или, альтернативно, подсистема безопасности может, при необходимости, получать данные из хранилища данных (например, рассмотренного здесь хранилища динамических данных для правил). Затем новые правила и/или данные обрабатываются (блок 804). Обработка новых правил и/или данных создает внутренний набор правил для подсистемы безопасности (например, во внутреннем формате подсистемы безопасности), которым она должна следовать, применяя новую политику безопасности.

Процесс 800 разветвляется в зависимости от того, увенчалась ли успехом обработка правил (блок 806). Если подсистема безопасности закончила обработку набора правил и/или данных, которые она получила, и готова начать использовать новый набор правил и/или данных (например, все, что остается сделать подсистеме безопасности, это изменить свой указатель, чтобы указать на новый набор правил вместо предыдущего набора правил), значит обработка прошла успешно. В противном случае, обработка не прошла успешно. Если обработка прошла успешно, то возвращается значение, обозначающее «успех» (блок 808). Если же обработка не прошла успешно, то возвращается значение, обозначающее «неудачу» (блок 810). Возвращаемые значения в блоках 808 и 810 возвращаются компоненту(ам), координирующему(им) обновление политики безопасности на хост-компьютере (например, рассмотренному здесь менеджеру правил).

Независимо от возвращенного значения, подсистема безопасности продолжает использовать предыдущий или старый набор правил, пока не получит предписание к откату или к началу использования новых правил (блок 812). В случае предписания к началу использования новых правил, подсистема безопасности начинает использовать новые правила и/или данные (блок 814), например, сменив указатель со своего предыдущего набора правил на свой новый набор правил. Предписание начать использование нового набора правил может быть получено подсистемой безопасности любым из разнообразных способов, рассмотренных выше.

Если же предписан откат, подсистема безопасности отменяет все результаты обработки новых правил и/или данных (блок 816). Эта отмена может осуществляться независимо от того, закончила ли подсистема безопасности обработку нового набора правил, который она получила.

Таким образом, из фиг.8 явствует, что подсистема безопасности продолжает использовать свой предыдущий набор правил, пока не получит указание к переключению на новый набор правил. К моменту получения такого указания подсистема безопасности готова начать использовать новые правила и для осуществления переключения требуется очень мало времени. Например, подсистеме

безопасности может быть необходимо просто переключить указатель, чтобы он указывал на ее новый набор правил вместо ее старого набора правил.

Программные интерфейсы приложений (API)

API, например API 124, рассмотренный выше в отношении фиг.1, позволяет передавать политики безопасности и информацию о событиях среди различных компонентов и программ (например, подсистем безопасности) в хост-компьютере. Согласно одному варианту осуществления API задан с использованием модели компонентных объектов (COM). API поддерживает методы загрузки и выгрузки подсистем безопасности, отправку политик безопасности подсистемам безопасности, передачу изменений в данных политики безопасности нужным подсистемам безопасности, что позволяет пользователю хоста взаимодействовать с подсистемой безопасности во время принятия решения, чтобы активировать или деактивировать определенные действия, предусмотренные политикой, и централизованное управление конфигурацией для подсистем безопасности.

Рассмотренные здесь системы и процедуры допускают централизованное управление защитой вычислительной системы со стороны политик безопасности, ориентированных на конкретную вычислительную систему или группу вычислительных систем. Кроме того, эти системы и процедуры собирают и сопоставляют события и другую информацию, например события, связанные с безопасностью, генерируемые этими вычислительными системами или другими источниками данных.

Согласно одному варианту осуществления интерфейс поддерживает клиентский доступ к политикам безопасности и базам данных событий посредством защищенных, аутентифицированных протоколов. Интерфейс допускает связь между различными компонентами или прикладными программами и одной или несколькими подсистемами безопасности. Интерфейс также определяет, как подсистемы безопасности связываются друг с другом и с другими устройствами, компонентами, услугами или программами.

Согласно одному варианту осуществления интерфейс задан как интерфейс COM с использованием специализированного загрузчика для снижения вероятности того, что атакующий переключит подсистемы безопасности на собственный код атакующего, изменив значения реестра COM.

Согласно иллюстративному варианту осуществления API поддерживает следующие функциональные вызовы:

Функциональные вызовы от агента к подсистеме безопасности

(Инициализация (Initialize)

(Отключение (Shutdown)

(Подготовка политики (PreparePolicy)

(Фиксация политики (CommitPolicy)

(Откат политики (RollbackPolicy)

(Запись данных (WriteData)

(Запись конфигурации (WriteConfig)

Эти семь функциональных вызовов называются «интерфейс ISecurityEngine».

Функциональные вызовы от подсистемы безопасности к агенту

(Чтение и регистрация данных извещения (ReadAndRegisterNotifyData)

(Запись данных подсистемы безопасности (WriteSEData)

(Отмена регистрации данных извещения (UnRegisterNotifyData)

(Получение атрибута данных (GetDataAttribute)

(Чтение и регистрация конфигурации извещения (ReadAndRegisterNotifyConfig)

(Отмена регистрации конфигурации извещения (UnRegisterNotifyConfig)

(Запрос пользователя (QueryUser)

(Завершение (Complete)

Первые семь вышеперечисленных функциональных вызовов называются «интерфейс ISecurityAgent», а последний функциональный вызов (Complete) называется «интерфейс IAgentCallback».

Функциональный вызов можно также называть просто «вызов», «функция» или «услуга». Ниже приведены подробности, касающиеся этих функциональных вызовов. Альтернативные варианты осуществления могут предусматривать использование дополнительных функциональных вызовов и/или исключение одного или нескольких рассмотренных здесь функциональных вызовов.

Согласно одному варианту осуществления агент, например менеджер событий или агент безопасности, связывается с подсистемами безопасности посредством API. Агент можно также называть «менеджер». В конкретных вариантах осуществления агент не вызывает конкретную подсистему безопасности, когда вызов API уже ожидает выполнения. Имеются исключения из этого правила для вызовов асинхронного API. В этих случаях разрешенные действия агента заданы ниже в таблицах состояний.

Функциональный вызов Initialize

Этот метод вызывается один раз для каждой подсистемы безопасности, известной агенту. Метод вызывается при запуске агента. Функциональный вызов Initialize загружает подсистему безопасности и позволяет ей осуществлять операции инициализации.

Агент вызывает этот метод асинхронно, по очереди для каждой подсистемы безопасности, и обратные вызовы обрабатываются по мере их поступления. Агент ожидает завершения всех обратных вызовов прежде, чем продолжить.

Этот метод задан следующим образом:

```
HRESULT Initialize(
[in] ISecurityAgent *pAgent,
[in] IAgentCallback *pCallback);
```

pAgent представляет собой интерфейс COM, который могут использовать подсистемы безопасности для обратного вызова к агенту.

pCallback представляет собой объект обратного вызова, заданный ниже.

В случае неудачи функционального вызова Initialize, или если он не возвращает значение в разумный промежуток времени, вызывается функция Shutdown. В силу возможных состояний гонок, подсистемы безопасности обрабатывают Shutdown прежде, чем Initialize возвратит значение.

Функциональный вызов Shutdown

Этот метод вызывается один раз для каждой подсистемы безопасности, для которой агент вызвал функцию Initialize. Метод позволяет подсистеме безопасности начать обработку своего отключения. Даже если Initialize оканчивается неудачей, агент вызывает Shutdown, чтобы подсистема безопасности могла закрыть любые выделенные ресурсы. Например, этот метод позволяет подсистеме безопасности осуществлять комплексное отключение, которое невозможно произвести в ходе выполнения DLL_PROCESS_DETACH.

Поскольку для выполнения этого метода может потребоваться значительное время, она использует объект обратного вызова, чтобы указать, что она завершила обработку отключения. Когда этот метод вызывается в результате отключения

системы в ходе процесса, время, доступное для завершения обработки, ограничивается, и агент может быть остановлен системой до завершения обратного вызова.

Этот метод задан следующим образом:

```
typedef enum tagSHUTDOWN_TYPE
{
SHUTDOWN_NORMAL=0,
SHUTDOWN_SYSTEM
} SHUTDOWN_TYPE;
HRESULT Shutdown(
[in] SHUTDOWN_TYPE eType,
[in] IAgentCallback *pCallback);
```

eTYPE является перечислением либо SHUTDOWN_NORMAL, либо SHUTDOWN_SYSTEM.

pCallback представляет собой объект обратного вызова, заданный ниже.

Выгрузка DLL происходит после осуществления обратного вызова Shutdown (или после наступления таймаута). Поскольку обратный вызов можно производить асинхронно, агенту может понадобиться ожидать поток, который выдает результат обратного вызова, прежде чем продолжить выгрузку DLL подсистемы безопасности. Это позволяет стековому фрейму обратного вызова разворачиваться до точки вне выгружаемой DLL и избегать исключения в процессе. Если обратный вызов производится в ходе вызова Shutdown, этот дополнительный этап не требуется, поскольку потоки подсистемы безопасности предполагаются отключенными.

Ошибки регистрируются как рабочие события, но в противном случае игнорируются, поскольку агент все равно близок к отключению.

Функциональный вызов PreparePolicy

Агент вызывает этот метод, когда получает обновленную политику. Результирующие политики сливаются, и каждый набор правил (RuleSet) строится для передачи надлежащей подсистеме безопасности. Данные XML передаются как объект IStream, который можно использоваться MSXML (XML Майкрософт) - либо DOM (объектной моделью документов), либо SAX (простой API для XML) - для чтения данных XML. Этот метод задан следующим образом:

```
HRESULT PreparePolicy(
[in] IStream *pstreamRuleset,
[in] IAgentCallback *pCallback);
```

pstreamRuleset представляет собой интерфейс COM к объекту Stream, который позволяет читать набор правил XML. Можно предположить, что указанный IStream является локальным по отношению к подсистеме и не обращается к данным по сети.

pCallback представляет собой объект обратного вызова, заданный ниже.

Если вызов возвращает ошибку, то предполагается, что подсистема безопасности продолжает выполнять ранее применявшуюся политику (которая может быть политикой, установленной во время загрузки). Агент вызывает RollbackPolicy для всех подсистем безопасности, на которых вызов PreparePolicy увенчался успехом, но ни для одной из подсистем безопасности, на которых этот вызов закончился неудачей. Этот процесс начинается, как только подсистема безопасности возвращает ошибку. Кроме того, если обратный вызов PreparePolicy не приходит вовремя, агент расценивает это как неудачу со стороны этой подсистемы безопасности. Поэтому подсистемы безопасности допускают, что агент может вызвать RollbackPolicy прежде, чем

вызов PreparePolicy возвратит значение.

Функциональный вызов CommitPolicy

Агент вызывает этот метод, когда все подсистемы безопасности возвратили успех в ответ на вызов PreparePolicy. Этот вызов инициирует их переключение на новую политику. Этот метод задан следующим образом:

HRESULT CommitPolicy(void);

Этот вызов является синхронным, и агент будет ждать завершения одного вызова, прежде чем перейти к следующему вызову. Согласно одному варианту осуществления ожидается, что вся работа, которая может привести к неудаче в обновлении политики, производится в вызове PreparePolicy, и этот вызов является простым переключателем со структур данных старой политики на структуры данных новой политики.

Метод CommitPolicy возвращает фатальные неудачи, например сбой в связи между пользователем и элементами ядра подсистемы безопасности. Когда этот вызов возвращает ошибку, агент пытается перезагрузить предыдущую политику и повторно применить эту политику. Вследствие ошибки, это может не сработать, и применение политики останется в неизвестном состоянии. Агент регистрирует рабочую ошибку, если CommitPolicy возвращает неудачу.

Функциональный вызов RollbackPolicy

Агент вызывает этот метод, когда подсистема безопасности возвращает ошибку в ответ на свой вызов PreparePolicy. Этот вызов принуждает агента предписать всем остальным подсистемам безопасности прервать обновление и вернуться к действующей политике. Этот метод задан следующим образом:

HRESULT RollbackPolicy(void);

Этот вызов является асинхронным, поскольку ожидается, что обработка отката подсистемами безопасности является экстенсивной, примерно повторяющей в обратном порядке действия, выполняемые по вызову PreparePolicy. После выполнения подсистемой безопасности обработки этого вызова подсистема безопасности вызывает Complete, чтобы указать состояние отката.

В случае отката политики, агент отменяет регистрацию любых данных, зарегистрированных согласно PreparePolicy, - система возвращается к предыдущему набору данных, подписанных каждой подсистемой безопасности. По этой причине, подсистемы безопасности не удаляют свои локальные копии зависших данных, пока не получат от агента вызов CommitPolicy. Агент отвечает за применение периода хронирования, в течение которого вызовы ReadAndRegisterData поступают от подсистем безопасности в ходе отката политики.

Обратные вызовы RollbackPolicy и Complete могут возвращать фатальные неудачи. Предполагается, что подсистемы безопасности реализуют PreparePolicy с возможностью поддержки отката. Агент регистрирует рабочую ошибку, когда RollbackPolicy возвращает неудачу. После этого о состоянии применения политики этой подсистемы безопасности и любой действующей с ней совместно подсистемой безопасности нельзя делать никаких предположений. Дальнейшие обновления политики будут продолжать отправляться на подсистему безопасности.

Функциональный вызов WriteData

Агент вызывает этот метод, когда изменяется фрагмент данных, в отношении которого подсистема безопасности ранее вызвала ReadAndRegisterNotifyData. Параметры аналогичны вызову ReadAndRegisterNotifyData за исключением того, что памятью распоряжается агент, поэтому подсистема безопасности не должна удалять элемент после его обработки.

WriteData не вызывается, когда подсистема безопасности находится в процессе получения новой политики от агента; т.е. между вызовами PreparePolicy и CommitPolicy/RollbackPolicy. Любые изменения данных, обнаруженные агентом в это время, группируются и направляются заинтересованным подсистемам безопасности, как только новая политика зафиксирована или выполнит ее откат. Агент оптимизирует очередь незавершенных обновлений во избежание, насколько возможно, передачи множественных последовательных изменений в одном и том же фрагменте данных. Метод WriteData задан следующим образом:

```
#define DF_DYNAMIC 0x1
#define DF_COLLECTION 0x2
#define DF_BOOLEAN 0x4
#define DF_PERSISTED 0x8
HRESULT WriteData(
[in] REFGUID guidDataID,
[in] DWORD dwFlags,
[in] DWORD dwDataSize,
[in] VARIANT varData,
[in] DWORD dwKeySize,
[in] byte *pbKeyValue).
```

Параметры для передачи Key (dwKeySize, pbKeyValue) используются при передаче контекста, связанного с предыдущим запросом, обратно подсистеме безопасности. Подсистема безопасности использует этот контекст для сопоставления результата запроса с предыдущим вызовом QueryUser, направляемым агенту. Эти дополнительные данные необходимы, поскольку данный запрос может происходить неоднократно для разных контекстов в одном и том же правиле, например спрашивая пользователя, разрешено ли приложению X изменить значение реестра, а затем задавая тот же вопрос относительно приложения Y.

Ошибки регистрируются как рабочие события, но в других отношениях игнорируются. Сбойная подсистема безопасности будет по-прежнему извещаться о дальнейших обновлениях того же фрагмента данных. Если подсистема безопасности желает препятствовать этому, она может вызвать UnRegisterNotifyData для этого фрагмента данных.

Функциональный вызов WriteConfig

Этот метод позволяет агенту распространять данные конфигурации на нужные подсистемы безопасности. После того, как подсистема безопасности прочитает данные конфигурации с использованием метода ReadAndRegisterNotifyConfig, агент, вызывающий этот метод, информирует ее об изменениях в этих данных. Метод задан следующим образом:

```
HRESULT WriteConfig(
[in] WCHAR *wszDataName,
[in] VARIANT varData);
```

wszDataName представляет собой имя типа Text записываемого элемента данных конфигурации, и это имя используется в реестре для этих данных.

varData представляет собой структуру типа VARIANT, содержащую единичный элемент данных, представляющий это имя. Эти данные могут относиться к разным типам в зависимости от того, какой тип данных имеется в реестре. Агент не проверяет тип данных - предполагается, что подсистема безопасности сама, по мере необходимости, проверяет тип данных, в соответствии с контекстом.

Ошибки регистрируются как рабочие события, но в других отношениях игнорируются. Сбойная подсистема безопасности будет по-прежнему извещаться о дальнейших обновлениях тех же данных параметра конфигурации. Если подсистема безопасности желает препятствовать этому, она может вызвать UnRegisterNotifyData для этого фрагмента данных.

Конкретная подсистема безопасности обычно не вызывает агент в то время, как вызов API от этого агента уже ожидает выполнения. Агент рассматривает это как ошибку и игнорирует второй вызов.

Функциональный вызов ReadAndRegisterNotifyData

Этот метод позволяет подсистеме безопасности считывать данные из подсистемы динамических данных для правил, чтобы использовать их при обработке своих правил. После того, как подсистема безопасности считает данные, агент, вызывающий метод WriteData интерфейса ISecurityEngine, информирует ее об изменениях в этих данных. Метод задан следующим образом:

```
HRESULT ReadAndRegisterNotifyData(
[in] REFGUID guidDataID,
[out] DWORD *pdwFlags,
[out] DWORD *pdwDataSize,
[out] VARIANT *pvarData);
```

guidDataID представляет собой GUID элемента данных, подлежащего извлечению.

pdwFlags представляет собой набор флагов, описывающих элемент данных. Иллюстративными значениями могут быть DYNAMIC или STATIC, а также COLLECTION или BOOLEAN.

pdwDataSize представляет собой полный размер элементов данных в массиве, возвращаемом в данных типа VARIANT.

pvarData представляет собой структуру типа VARIANT, содержащую ссылку на массив элементов данных, или значение элемента данных для данных булева типа. Вариант на входе является пустым.

Если подсистема безопасности обращается к данным, которых уже нет в политике, это является ошибкой. Агент генерирует рабочую ошибку по любой ошибке. В этом случае нет никакой гарантии, что подсистема безопасности и агент имеют согласованную точку зрения на поврежденные данные.

Функциональный вызов WriteSEData

Подсистема безопасности вызывает этот метод, когда изменяется фрагмент данных, которым подсистема безопасности владеет и который она публикует (для сохранности или для использования другими подсистемами безопасности). Параметры аналогичны вызову WriteData, за исключением того, что памятью распоряжается подсистема безопасности, поэтому агент не удаляет элемент после его обработки. Метод задан следующим образом:

```
HRESULT WriteSEData(
[in] REFGUID guidDataID,
[in] DWORD dwDataSize,
[in] VARIANT varData).
```

Этот метод можно вызывать в любое время, в том числе, когда другой вызов WriteSEData все еще ожидает обработки, на любом потоке. В случае необходимости, гарантировать сериализацию призван агент.

Владелец элемента данных идентифицируется в определении коллекции посредством GUID, который задает владельца. Это может быть GUID подсистемы

безопасности или идентификатор агента или, возможно, идентификатор другого потребителя.

Если подсистема безопасности определяет коллекцию, которой она владеет, предполагается, что данные будут опубликованы для агента через этот API.

Агент регистрирует любую ошибку как рабочее событие. Подсистема безопасности может решать, продолжать ли обеспечение обновлений после ошибки. Нет гарантии, что после ошибки агентская версия данных согласуется с точкой зрения подсистемы безопасности.

Функциональный вызов UnRegisterNotifyData

Этот метод позволяет подсистеме безопасности останавливать получение извещений WriteData для элементов данных, которыми она больше не интересуется. Метод задан следующим образом:

```
HRESULT UnRegisterNotifyData(
[in] REFGUID guidDataID);
```

guidDataID представляет собой GUID, идентифицирующий элемент данных, в изменении извещений для которого подсистема безопасности больше не нуждается. Подсистема безопасности может указать, что она желает отменить регистрацию всех текущих извещений, передав GUID со значением Null {00000000-0000-0000-000000000000}.

Агент регистрирует любую ошибку как рабочее событие. В том числе, это происходит, когда данные неизвестны агенту, чтобы было легче диагностировать проблемы управления политикой.

Функциональный вызов GetDataAttribute

Этот метод позволяет подсистеме безопасности извлекать конкретный атрибут, связанный с элементом данных. Имя атрибута такое же, как имя, присутствующее в XML политики, включая случай текста. Значения атрибутов могут изменяться только при смене политики, поэтому никакой системы извещения для этих данных не требуется. Метод задан следующим образом:

```
HRESULT GetDataAttribute(
[in] REFGUID guidDataID,
[in] WCHAR *wszAttributeName,
[out] VARIANT *pvarAttributeValue);
```

Этот метод можно вызывать в любое время.

guidDataID представляет собой GUID, идентифицирующий элемент данных, для которого извлекается атрибут.

wszAttributeName представляет собой имя атрибута, точно такое же, как в документе политики.

pvarAttributeValue представляет собой значение атрибута в виде Variant. Применяются нормальные правила выделения ресурсов для выходных параметров. Агент выделяет новый Variant с информацией, и в обязанность вызывающего входит его дальнейшее освобождение.

Функциональный вызов ReadAndRegisterNotifyConfig

Этот метод позволяет подсистеме безопасности считывать данные конфигурации из агента. После того, как подсистема безопасности считает данные конфигурации, агент, вызывающий метод WriteData интерфейса ISecurityEngine, информирует ее об изменениях в этих данных.

Данные конфигурации для агента и его собственных подсистем безопасности могут размещаться в общем корневом каталоге. Метод задан следующим образом:

HRESULT ReadAndRegisterNotifyConfig(
[in] WCHAR *wszDataName,
[out] VARIANT *pvarData);

wszDataName представляет собой имя типа Text элемента данных конфигурации, подлежащего извлечению, и это имя используется в реестре для этих данных. Оно идентифицирует отдельный элемент относительно общего корневого каталога агента. Символ «\» спереди не требуется. Значение нечувствительно к регистру, но символы пробела являются значимыми.

pvarData представляет собой структуру типа VARIANT, содержащую единичный элемент данных, представляющий имя. Эти данные могут относиться к разным типам в зависимости от того, какой тип данных имеется в реестре. Агент не проверяет тип данных - предполагается, что подсистема безопасности сама, по мере необходимости, проверяет тип данных в соответствии с контекстом.

Любую ошибку агент регистрирует как рабочее событие.

Функциональный вызов UnRegisterNotifyConfig

Этот метод позволяет подсистеме безопасности остановить получение извещений WriteConfig для элементов данных, которыми она больше не интересуется. Метод задан следующим образом:

HRESULT UnRegisterNotifyConfig(
[in] WCHAR *wszDataName);

wszDataName представляет собой имя типа Text, идентифицирующее элемент данных конфигурации, в извещениях об изменении которого подсистема безопасности больше не нуждается.

Агент регистрирует любую ошибку как рабочее событие. В том числе, это происходит, когда данные неизвестны агенту, чтобы было легче диагностировать проблемы управления политикой.

Функциональный вызов QueryUser

Это метод позволяет подсистеме безопасности запросить агент отобразить пользователю то или иное сообщение, возвращая ответ, выбранный пользователем. Агент может также кэшировать этот ответ и сохранять это значение при перезапуске агента. Вопрос, задаваемый пользователю, может содержать конкретную информацию о том, почему пользователю задан этот вопрос. Эта информация может обеспечиваться подсистемой безопасности и может быть разной каждый раз при вызове этого метода. Как агент решает, был ли этот вопрос задан раньше, и каков ответ, определяется информацией ключа, которую обеспечивает подсистема безопасности.

Вызов немедленно возвращает значение подсистеме безопасности. Затем подсистема безопасности приостанавливает действие сеанса/потока, который инициировал этот запрос, пока не получит извещение об ответе. Это происходит, когда пользователь набирает ответ на клавиатуре, или по истечении времени запроса. Обработка таймаута осуществляется агентом. При этом агент обновляет соответствующий элемент данных ответом, набранным на клавиатуре или принятым по умолчанию, и извещает подсистему безопасности о результате с соответствующим контекстом.

Поскольку при получении ответа на такие запросы время играет решающую роль, подсистема безопасности, применяющая правило, требующее выдачи запроса, может вызывать этот API в любое время. Метод задан следующим образом:

HRESULT QueryUser(

[in] REFGUID guidQueryItem,
[in] DWORD dwKeySize,
[in] byte *pbKeyValue,
[in] SAFEARRAY(VARIANT) pvarQueryParams);

guidQueryItem представляет собой GUID элемента данных, который содержит базовые строки (Basic string), используемые для задания вопроса пользователю и предоставления возможных ответов на этот вопрос.

dwKeySize представляет собой длину значения ключа в байтах.

pbKeyValue представляет собой набор байтов, которые задают уникальный ключ для этого запроса.

pvarQueryParams представляет собой тип данных Safearray переменных типа Variant, содержащий параметры запроса, подлежащие подстановке в текст запроса, отображаемый пользователю. Порядок и синтаксис параметров определяется типом правила, с которым связано действие QueryUser.

Агент возвращает ошибку, если элемент данных не поддается идентификации. Ошибки при выполнении запроса регистрируются как рабочие события. В этом случае подсистеме безопасности возвращается действие, принятое по умолчанию.

Функциональный вызов Complete

Этот метод извещает агента о том, что подсистема безопасности завершила обработку, связанную с предыдущим асинхронным вызовом от агента на эту подсистему безопасности. Хотя конкретная подсистема безопасности может, потенциально, иметь от агента более одного асинхронного вызова, ожидающего обработки, агент управляет внутренним состоянием каждой подсистемы безопасности так, чтобы контекст конкретного обратного вызова Complete был однозначным. Метод задан следующим образом:

HRESUET Complete(
[in] HRESUET hrCompletionCode);

hrCompletionCode представляет собой возвращаемый код для асинхронного вызова, который агент ранее произвел для этой подсистемы безопасности.

Использование интерфейса

Ниже описан пример ограничений, налагаемых на использование API для взаимодействия с одной или несколькими подсистемами безопасности.

В конкретный момент времени подсистема безопасности находится в определенном состоянии относительно ее взаимодействия с агентом. В нижеприведенном списке указаны возможные состояния подсистемы безопасности.

| Состояние | Определение |
|---|---|
| Ожидание инициализации | DLL подсистемы состояния загружены, но пока не получено ни одного вызова API. Состояние политики в этот момент зависит от подсистемы безопасности - NSE имеет политику времени загрузки, поведенческая блокировка ничего не имеет, пока агент не даст правила. |
| Инициализация | Initialize был вызван, но не завершен |
| Выполнение | Подсистема безопасности сделала обратный вызов агенту, чтобы сообщить об успешной инициализации, и применяет либо (первоначально) время загрузки, либо (после последующего CommitPolicy) политику, выданную агентом. |
| Подготовка политики | PreparePolicy вызван, но ни одного обратного вызова не было |
| Политика подготовлена | Обратный вызов PreparePolicy завершился возвращением кода «успех», ожидание вызова CommitPolicy |
| Откат политики | Подсистема безопасности, получившая вызов RollbackPolicy, обрабатывает запрос отката |
| Отключение | Shutdown вызван, но не завершен |
| Ожидание окончания | Shutdown завершен - ожидание окончания процесса |

Разрешенные взаимодействия между агентом и подсистемами безопасности можно формализовать в виде набора таблиц, задающих API, которые каждая сущность может вызывать, когда подсистема безопасности находится в конкретном состоянии, и какое изменение состояния или иное действие необходимо осуществить подсистеме безопасности в результате. Рабочее состояние агента предполагается несущественным - подсистемы безопасности могут предположить, что он все время остается в нормальном рабочем режиме, в то время как подсистемы безопасности загружаются в память.

Таблицы состояния охватывают следующие фазы рабочего цикла подсистемы безопасности:

(Инициализация

(Обновление политики от агента

(Отключение

Любую комбинацию вызова API и состояния подсистемы безопасности, не охваченную этими таблицами, следует считать неправильным использованием API. Ответственность за избежание этого неправильного использования лежит на стороне, вызывающей API.

В нижеследующей таблице заданы разрешенные последовательности API в ходе инициализации подсистемы безопасности и изменения состояния подсистемы безопасности в соответствии с вводами от агента. Вызов любого API, не указанный в качестве разрешенного ввода для списка состояний, связанного с инициализацией подсистемы безопасности, означает ошибку протокола со стороны вызывающей сущности.

| Состояние подсистемы безопасности<br>Вызовы API агента | Ожидание инициализации | Инициализация |
|---|---|---|
| Initialize | Инициализация | Ошибка |
| Shutdown | Ошибка | Ожидание окончания |
| WriteConfig | Ошибка | Успех |
| Вызовы API подсистемы безопасности | | |
| Complete(успех) | Ошибка | Выполнение (нет политики) |
| Complete(неудача) | Ошибка | Ожидание окончания |
| ReadAndRegisterNotify Config | Ошибка | Успех |

В следующей таблице состояний заданы разрешенные последовательности API в ходе обновления политики и соответствующие изменения состояния подсистемы безопасности. Вызов любого API, не указанный в качестве разрешенного ввода для списка состояний, связанного с обновлением политики, означает ошибку протокола со стороны вызывающей сущности.

| Состояние подсистемы безопасности<br>Вызовы API агента | Выполнение | Подготовка политики |
|---|---|---|
| PreparePolicy | Подготовка политики | Ошибка |
| WriteConfig | Успех | Успех |
| Вызовы API подсистемы безопасности | | |
| Complete(OK) | Ошибка | Политика подготовлена |
| Complete(FAIL) | Ошибка | Выполнение (старая политика) |
| ReadAndRegisterNotifyConfig | Успех | Успех |
| ReadAndRegisterNotifyData | Успех | Успех |

| Состояние подсистемы безопасности<br>Вызовы API агента | Политика подготовлена |
|---|---|
| CommitPolicy | Выполнение (новая политика) |
| RollbackPolicy | Откат политики (старая политика) |
| WriteConfig | Успех |
| Вызовы API подсистемы безопасности | |
| Complete(успех) | Ошибка |
| Complete(неудача) | Ошибка |
| ReadAndRegisterConfig | Успех |
| ReadAndRegisterData | Успех |

Ниже описан пример последовательности обновления политики в целом, с учетом множественных собственных подсистем безопасности.

1. Вызывается PreparePolicy каждой подсистемы безопасности.

2. Агент ожидает от каждой подсистемы безопасности вызова Complete с успехом или неудачей.

3. Если какая-либо подсистема безопасности сообщает о неудаче, все остальные подсистемы безопасности осуществляют вызов своих методов RollbackPolicy.

4. Если ни одна подсистема безопасности не сообщает о неудаче, то для каждой подсистемы безопасности вызывается метод CommitPolicy.

5. Если обнаружена другая неудача или если необходим Shutdown до того, как вызваны какие-либо методы CommitPolicy, то для каждой подсистемы безопасности вызывается метод RollbackPolicy.

В нижеследующей таблице состояний заданы разрешенные последовательности API в ходе отключения подсистемы безопасности и изменения состояния подсистемы безопасности в соответствии с вводами агента. Вызов любого API, не указанный в качестве разрешенного ввода для списка состояний, связанного с отключением подсистемы состояний, означает ошибку протокола со стороны вызывающей сущности.

| Состояние подсистемы безопасности<br>Вызовы API агента | Инициализация, Выполнение, Подготовка политики, Политика подготовлена, Откат политики | Отключение |
|---|---|---|
| Shutdown | Отключение | Ошибка |
| Вызовы API МБ | | |
| Complete | Ошибка | Ожидание окончания |

Ниже перечислены иллюстративные типы коллекций, поддерживаемые агентом, и описания того, как каждая коллекция передается в качестве динамических данных посредством вызовов метода ReadAndRegisterNotifyData и WriteData.

Многими из рассмотренных ниже элементов данных можно манипулировать, передавая единичную строку BSTR, или упаковывая беззнаковые целые в тип LONG или a LONGLONG variant. Те элементы, которые не легко подогнать под эту модель, представляют собой DirectorySet, ProtocolSet и IPv4AddressSet. Для каждого из этих типов предусмотрена система упаковки, которая упаковывает данные в строку BSTR, чтобы их можно было легко преобразовать в SafeArray.

FileSet

Данные, передаваемые для каждого элемента:

Имя файла - строка

Реализация:

BSTR

DirectorySet

Данные, передаваемые для каждого элемента:
Имя директории - строка
Рекурсивный - флаг
Реализация:
Упакованная BSTR - «Рекурсивный флаг:строка»
Рекурсивный флаг является единичным символом -
«R» - рекурсивный
«F» - линейный

RegistrySet

Данные, передаваемые для каждого элемента:
Имя ключа реестра - строка
Реализация:
Упакованная BSTR - «Рекурсивный флаг:строка»
Рекурсивный флаг является единичным символом -
«R» - рекурсивный
«F» - линейный

Протокол

Данные, передаваемые для каждого элемента:
Первичный /вторичный - строка или перечисление
Тип IP - строка или перечисление
Направление - строка или перечисление
Порт или диапазон портов - одно или два целых (16-битовые беззнаковые целые)
Реализация:
Упакованный LONGLONG:
1 байт - Первичный /вторичный
1 байт - тип IP TCP/UDP
1 байт - направление вход/выход/оба
1 байт - не используется
2 байта - конец диапазона портов (или нуль)
2 байта - начало диапазона портов (или порт)

ProcessSet

Данные, передаваемые для каждого элемента:
Имя процесса или путь - строка
Реализация:
BSTR

NetworkPortSet

Данные, передаваемые для каждого элемента:
Порт или диапазон портов - одно или два целых (16 бит, беззнаковые целые)
Реализация:
Упакованное LONG: начало=младшее слово, конец=старшее слово. Старшее слово равно нулю, если не диапазон портов

NetworkIPv4AddressSet

Данные, передаваемые для каждого элемента:
Один из:
Адрес IPv4 - строка (может содержать групповые символы)
Диапазон адресов IPv4 - 2 строки
FQDN - строка
Имя хоста - строка

Реализация:
Упакованная BSTR: «T:Строка 1:Строка 2»
T - тип - один символ для адреса, диапазона адресов, имени хоста или FQDN
Строка 1 - адрес, начальный адрес, имя хоста или FQDN
Строка 2 - конечный адрес диапазона адресов
UserSet
Данные, передаваемые для каждого элемента:
Имя учетной записи пользователя - строка
Реализация:
BSTR
UserGroupSet
Данные, передаваемые для каждого элемента:
Имя группы пользователей - строка
Реализация:
BSTR
FileOpSet
Данные, передаваемые для каждого элемента:
Файловая операция - строка (или перечисление)
Реализация:
BSTR
DirOpSet
Данные, передаваемые для каждого элемента:
Операция над директориями - строка (или перечисление)
Реализация:
BSTR
ProcessOpSet
Данные, передаваемые для каждого элемента:
Операция над процессом - строка (или перечисление)
Реализация:
BSTR
RegKeyOpSet
Данные, передаваемые для каждого элемента:
Операция над ключами реестра - строка (или перечисление)
Реализация:
BSTR
RegValueOpSet
Данные, передаваемые для каждого элемента:
Операции над значениями реестра - строка (или перечисление)
Реализация:
BSTR
UserOpSet
Данные, передаваемые для каждого элемента:
Операция над учетными записями пользователей - строка (или перечисление)
Реализация:
BSTR
UserGroupOpSet
Данные, передаваемые для каждого элемента:
Операция над группами пользователей - строка (или перечисление)

Реализация:
BSTR
JobOpSet
Данные, передаваемые для каждого элемента:
Операция над заданиями - строка (или перечисление)
Реализация:
BSTR
Generic
Данные, передаваемые для каждого элемента:
Значение - строка
Реализация:
BSTR
QuerySet

Для подсистемы безопасности QuerySet выглядит как коллекция из одного элемента, содержащая результат запроса пользователя. Соответствующий контекст передается подсистеме безопасности как отдельный параметр. Внутренняя структура QuerySet обычно не нужна подсистеме безопасности, нужны только контекст и результат запроса.

Boolean (boolDefine)
Данные, передаваемые для каждого элемента:
Булевы значения - истина или ложь
Реализация:
LONG - ложь=0, истина=1

На фиг.9 показана общего вида вычислительная среда 900, которую можно использовать для реализации описанных здесь принципов. Вычислительная среда 900 является всего лишь примером вычислительной среды и не призвана как-либо ограничивать объем использования или функциональные возможности компьютерной и сетевой архитектур. Также, вычислительную среду 900 не следует рассматривать как имеющую какую-либо зависимость или требование в отношении к какому-либо одному компоненту, проиллюстрированному в иллюстративной вычислительной среде 900, или их комбинации.

Вычислительная среда 900 включает в себя вычислительное устройство общего назначения в виде компьютера 902. Компоненты компьютера 902 могут включать в себя, но не исключительно, один или несколько процессоров или блоков обработки 904 (в необязательном порядке, включающих в себя криптографический процессор или сопроцессор), системную память 906 и системную шину 908, которая подключает различные компоненты системы, включая процессор 904, к системной памяти 906.

Системная шина 908 относится к любому из нескольких типов шинных структур, включая шину памяти или контроллер памяти, двухточечное соединение, коммутирующее волокно, периферийную шину, ускоренный графический порт и процессор или локальную шину, с использованием разнообразных шинных архитектур. В порядке примера, такие архитектуры могут включать в себя шину архитектуры промышленного стандарта (ISA), шину микроканальной архитектуры (MCA), шину расширенного стандарта ISA (EISA), локальную шину Ассоциации по стандартам в области видеоэлектроники (VESA) и шину подключений периферийных компонентов (PCI), также именуемую шиной расширения.

Компьютер 902 обычно содержит разнообразные компьютерно-считываемые

носители. Такие носители могут представлять собой любые имеющиеся носители, к которым может осуществлять доступ компьютер 902, и включают в себя энергозависимые и энергонезависимые носители, сменные и стационарные носители.

Системная память 906 содержит компьютерно-считываемые носители в виде энергозависимой памяти, например оперативной памяти (ОЗУ) 910, и/или энергонезависимой памяти, например постоянной памяти (ПЗУ) 912. Базовая система ввода/вывода (BIOS) 914, содержащая основные процедуры, которые помогают переносить информацию между элементами компьютера 902, например при запуске, хранится в ПЗУ 912. ОЗУ 910 обычно содержит данные и/или программные модули, которые непосредственно доступны блоку обработки 904 и/или в данный момент обрабатываются им.

Компьютер 902 может также включать в себя другие сменные/стационарные, энергозависимые/энергонезависимые компьютерные носители данных. В порядке примера, на фиг.9 показан жесткий диск 916, который производит считывание со стационарного энергонезависимого магнитного носителя (не показан) и запись на него, привод 918 магнитного диска, который производит считывание со сменного энергонезависимого магнитного диска 920 (например, «флоппи-диска») и запись на него, и привод 922 оптического диска, который производит считывание со сменного энергонезависимого оптического диска 924, например CD-ROM, DVD-ROM или другого оптического носителя, и запись на него. Привод 916 жесткого диска, привод 918 магнитного диска и привод 922 оптического диска подключены к системной шине 908 посредством одного или нескольких интерфейсов 925 носителей данных. Альтернативно, привод 916 жесткого диска, привод 918 магнитного диска и привод 922 оптического диска могут быть подключены к системной шине 908 посредством одного или нескольких интерфейсов (не показаны).

Дисководы и соответствующие компьютерно-считываемые носители обеспечивают хранение компьютерно-считываемых команд, структур данных, программных модулей и других данных для компьютера 902. Хотя пример иллюстрирует жесткий диск 916, сменный магнитный диск 920 и сменный оптический диск 924, очевидно, что для реализации иллюстративной вычислительной системы и среды можно также использовать другие типы компьютерно-считываемых носителей, в которых можно хранить данные, доступные компьютеру, например магнитные кассеты или другие магнитные запоминающие устройства, карты флэш-памяти, CD-ROM, цифровые универсальные диски (DVD) или другие оптические носители данных, оперативную память (ОЗУ), постоянную память (ПЗУ), электрически стираемую программируемую постоянную память (ЭСППЗУ) и т.п.

На жестком диске 916, магнитном диске 920, оптическом диске 924, ПЗУ 912 и/или ОЗУ 910 может храниться любое количество программных модулей, включая, например, операционную систему 926, одну или несколько прикладных программ 928, другие программные модули 930 и программные данные 932. Каждая из операционной системы 926, одной или несколько прикладных программ 928, других программных модулей 930 и программных данных 932 (или некоторая их комбинация) может реализовать все или часть резидентных компонентов, которые поддерживают распределенную файловую систему.

Пользователь может вводить команды и информацию в компьютер 902 через устройства ввода, например клавиатуру 934 и указательное устройство 936 (например, «мышь»). Другие устройства ввода 938 (не показаны) могут включать в себя микрофон, джойстик, игровую панель, спутниковую антенну, последовательный порт,

сканер и/или т.п. Эти и другие устройства ввода подключены к блоку обработки 904 через интерфейсы 940 ввода/вывода, которые подключены к системной шине 908, но могут быть подключены посредством других структур интерфейса и шины, например, параллельного порта, игрового порта или универсальной последовательной шины (USB).

Монитор 942 или устройство отображения другого типа может также быть подключен к системной шине 908 через интерфейс, например видеоадаптер 944. Помимо монитора 942, другие периферийные устройства вывода могут включать в себя такие компоненты, как громкоговорители (не показаны) и принтер 946, которые могут быть подключены к компьютеру 902 через интерфейсы 940 ввода/вывода.

Компьютер 902 может работать в сетевой среде с использованием логических соединений с одним или несколькими удаленными компьютерами, например удаленным вычислительным устройством 948. Например, удаленное вычислительное устройство 948 может представлять собой персональный компьютер, портативный компьютер, сервер, маршрутизатор, сетевой компьютер, равноправное устройство или другой общий сетевой узел, игровую приставку и т.п. Удаленное вычислительное устройство 948 показано в виде портативного компьютера, который может содержать многие или все элементы, описанные здесь применительно к компьютеру 902.

Логические соединения между компьютером 902 и удаленным компьютером 948 представлены в виде локальной сети (ЛС) 950 и глобальной сети (ГС) 952. Такие сетевые среды обычно используются в офисных, производственных компьютерных сетях, интрасетях и в Интернете.

При использовании в сетевой среде ЛС компьютер 902 подключен к локальной сети 950 через сетевой интерфейс или адаптер 954. При использовании в сетевой среде ГС, компьютер 902 обычно содержит модем 956 или другие средства установления соединений по глобальной сети 952. Модем 956, который может быть внутренним или внешним по отношению к компьютеру 902, может быть подключен к системной шине 908 через интерфейсы 940 ввода/вывода или другие соответствующие механизмы. Очевидно, что показанные сетевые соединения являются иллюстративными и что можно использовать другие средства установления линии(й) связи между компьютерами 902 и 948.

В сетевой среде, например проиллюстрированной вычислительной среде 900, программные модули, указанные в отношении компьютера 902, или часть из них, могут храниться в удаленном запоминающем устройстве. В порядке примера, удаленные прикладные программы 958 размещены в запоминающем устройстве на удаленном компьютере 948. Очевидно, что показанные сетевые соединения являются иллюстративными, и что можно использовать другие средства установления линии(й) связи между компьютерами. В иллюстративных целях, прикладные программы и другие выполнимые программные компоненты, например операционная система, показаны здесь в виде отдельных блоков, хотя очевидно, что такие программы и компоненты в разное время размещаются в разных запоминающих компонентах вычислительного устройства 902 и выполняются процессором(ами) данных компьютера.

Различные модули и способы можно описать в общем контексте компьютерно-выполняемых инструкций, например, программных модулей, выполняемых одним или несколькими компьютерами или другими устройствами. В общем случае программные модули включают в себя процедуры, программы, объекты, компоненты, структуры данных и т.п., которые выполняют конкретные

задания или реализуют определенные абстрактные типы данных. Обычно функциональные возможности программных модулей можно, по желанию, объединять или распределять в различных вариантах осуществления.

Реализация этих модулей и способов может храниться или передаваться посредством того или иного вида компьютерно-считываемого носителя. Компьютерно-считываемый носитель может представлять собой любой имеющийся носитель, к которому компьютер может осуществлять доступ. В порядке примера, но не ограничения, компьютерно-считываемый носитель может представлять собой «компьютерный носитель данных» или «среду передачи данных».

Компьютерные носители данных включают в себя энергозависимые и энергонезависимые, сменные и стационарные носители, реализованные посредством любого способа или технологии для хранения информации, например компьютерно-считываемых команд, структур данных, программных модулей или других данных. Компьютерные носители данных включают в себя, но не исключительно, ОЗУ, ПЗУ, ЭСППЗУ, флэш-память или другую технологию памяти, CD-ROM, цифровые универсальные диски (DVD) или иные оптические носители данных, магнитные кассеты, магнитную ленту, магнитные дисковые носители данных или иные магнитные запоминающие устройства или любой другой носитель, который можно использовать для хранения полезной информации и к которому компьютер может осуществлять доступ.

Среды передачи данных обычно воплощают компьютерно-считываемые команды, структуры данных, программные модули или другие данные в виде модулированного сигнала данных, например несущей волны или иного транспортного механизма. Среды передачи данных также включают в себя любые среды доставки информации. Термин «модулированный сигнал данных» означает сигнал, одна или несколько характеристик которого изменяется так, чтобы кодировать информацию в сигнале. В порядке примера, но не ограничения, среды передачи данных содержат проводные среды, например проводную сеть или прямое проводное соединение, и беспроводные среды, например акустические, РЧ, инфракрасные и другие беспроводные среды. В число компьютерно-считываемых сред входят также комбинации любых из вышеперечисленных позиций.

Теоретически, программный интерфейс можно рассматривать обобщенно, как показано на фиг.10 или фиг.11. На фиг.10 показан интерфейс Интерфейс 1, служащий каналом связи между первым и вторым сегментами кода. На фиг.11 показан интерфейс, содержащий интерфейсные объекты I1 и I2 (которые могут быть или не быть частью первого и второго сегментов кода), которые позволяют первому и второму сегментам кода связываться посредством носителя M. Согласно фиг.11 интерфейсные объекты I1 и I2 можно рассматривать как отдельные объекты одной системы, а также можно считать, что объекты I1 и I2 плюс носитель M образуют интерфейс. Хотя на фиг.10 и 11 показаны двусторонний поток и интерфейсы по обе стороны потока, некоторые реализации могут обеспечивать только один информационный поток в одном направлении (или ни одного информационного потока, как описано ниже) или интерфейсный объект с одной стороны. В порядке примера, но не ограничения, определение программного интерфейса охватывает такие термины, как программный интерфейс приложения (API), точка ввода, метод, функция, подпрограмма, вызов удаленной процедуры и интерфейс модели компонентных объектов (COM).

Аспекты такого программного интерфейса могут включать в себя метод,

посредством которого первый сегмент кода передает информацию (где термин «информация» используется в самом широком смысле и включает в себя данные, команды, запросы и т.д.) второму сегменту кода; метод, посредством которого второй сегмент кода принимает информацию; и структуру, последовательность, синтаксис, организацию, схему, хронирование и содержимое информации. В этой связи, сам по себе нижележащий транспортный носитель может быть не важен для действия интерфейса, является ли носитель проводным или беспроводным или комбинированным, при условии, что информация переносится так, как определено интерфейсом. В некоторых случаях информация может не передаваться в одном или обоих направлениях в традиционном смысле, поскольку перенос информации может либо осуществляться другим механизмом (например, информация может помещаться в буфер, файл и т.п., отдельный от информационного потока между сегментами кода), либо не существовать, как в случае, когда один сегмент кода просто обращается к функции, осуществляемой вторым сегментом кода. Некоторые или все эти аспекты могут быть важны в данном случае, например, в зависимости от того, являются ли сегменты кода частью системы в свободно связанной или тесно связанной конфигурации, и поэтому этот список следует рассматривать в порядке иллюстрации, но не ограничения.

Это определение программного интерфейса известно специалистам в данной области и явствует из вышеприведенного подробного описания изобретения. Существуют, однако, другие пути реализации программного интерфейса и, если явно не указано обратное, их также следует рассматривать в объеме формулы изобретения, приведенной в конце этого описания изобретения. Такие другие пути могут оказаться более изощренными или сложными, чем упрощенная функция для достижения, в принципе, того же результата. Теперь кратко будут описаны некоторые иллюстративные альтернативные реализации программного интерфейса.

РАЗЛОЖЕНИЕ

Коммуникация от одного сегмента кода к другому может осуществляться косвенно, путем разбиения коммуникации на множество дискретных коммуникаций. Это схематически представлено на фиг.12 и 13. Там показано, что некоторые интерфейсы можно описывать в терминах делимых множеств функций. Таким образом, функции интерфейса, показанные на фиг.10 и 11, можно разлагать для получения того же результата, так же, как число 24 можно представить как 2, умноженное на 2, умноженное на 3, умноженное на 2. Соответственно, согласно фиг.12, функцию, обеспечиваемую интерфейсом Интерфейс 1, можно разложить для преобразования коммуникаций интерфейса по множественным интерфейсам Интерфейс 1A, Интерфейс 1 B, Интерфейс 1C и т.д., получая тот же результат. Согласно фиг.13 функцию, обеспечиваемую интерфейсом I1, можно разложить по множественным интерфейсам I1a, I1b, I1c и т.д., получая тот же результат. Аналогично, интерфейс I2 второго сегмента кода, который принимает информацию от первого сегмента кода, можно разложить на множественные интерфейсы I2a, I2b, I2c и т.д. При разложении количество интерфейсов, входящих в состав первого сегмента кода, не обязано совпадать с количеством интерфейсов, входящих в состав второго сегмента кода. В любом из случаев, представленных на фиг.12 и 13, функциональная сущность интерфейсов Интерфейс 1 и I1 остается такой же, как показано на фиг.10 и 11 соответственно. Разложение интерфейсов можно также осуществлять в соответствии со свойствами ассоциативности, коммутативности и другими математическими свойствами, так что разложение может быть трудно распознаваемым. Например,

упорядочение операций может быть не важно, и, следовательно, функция, осуществляемая интерфейсом, может хорошо осуществляться до достижения интерфейса другим фрагментом кода или интерфейса или осуществляться отдельным компонентом системы. Кроме того, профессиональному программисту очевидно, что одного и того же результата можно достичь разными путями, делая разные функциональные вызовы.

ПЕРЕОПРЕДЕЛЕНИЕ

В некоторых случаях имеется возможность игнорировать, добавлять или переопределять определенные аспекты (например, параметры) программного интерфейса, тем не менее получая нужный результат. Это показано на фиг.14 и 15. Пусть, например, интерфейс Интерфейс 1, показанный на фиг.10, содержит функциональный вызов Square (input, precision, output) (квадрат), вызов, который содержит три параметра, input (вход), precision (точность) и output (выход), и который исходит от 1-го сегмента кода на 2-й сегмент кода. Если средний параметр, precision, не играет роли в данном сценарии, как показано на фиг.14, его можно просто игнорировать или даже заменить (в данной ситуации) параметром meaningless (не имеющий смысла). Можно также добавить параметр additional (дополнительный), не играющий роли. В любом случае функция «квадрат» может выполняться при условии, что выходное значение возвращается после того, как входное значение возводится в квадрат вторым сегментом кода. «Точность» может быть весьма важным параметром для некоторой последующей или другой части вычислительной системы; если же выясняется, что для узкой цели вычисления квадрата «точность» не требуется, ее можно заменить или игнорировать. Например, вместо того, чтобы передавать действительное значение «точность», можно передавать не имеющее смысла значение, например дату рождения, не оказывая отрицательного влияния на результат. Аналогично, как показано на фиг.15, интерфейс I1 можно заменить интерфейсом I1′, переопределенным на игнорирование или добавление параметров к интерфейсу. Интерфейс I2 можно аналогично переопределить как интерфейс I2′, переопределенный на игнорирование ненужных параметров или параметров, которые можно обрабатывать в другом месте. Суть в том, что в некоторых случаях программный интерфейс может включать в себя аспекты, например параметры, которые по некоторой причине не нужны и потому могут игнорироваться или переопределяться или обрабатываться в другом месте для других целей.

ВНУТРИТЕКСТОВОЕ КОДИРОВАНИЕ

Возможно также осуществлять слияние некоторых или всех функций двух отдельных модулей кода, в результате чего «интерфейс» между ними изменяет форму. Например, функции, указанные на фиг.10 и 11, можно преобразовать в функции, указанные на фиг.16 и 17 соответственно. Согласно фиг.16 предыдущие 1-й и 2-й сегменты кода, показанные на фиг.10, сливаются в модуль, содержащий их обе. В этом случае сегменты кода могут по прежнему связываться друг с другом, но интерфейс можно преобразовать к форме, более пригодной для единого модуля. Так, например, формальные операторы Call и Return могут уже не требоваться, но аналогичные обработка или отклик(и), соответствующие интерфейсу Интерфейс 1, по-прежнему могут иметь силу. Аналогично, как показано на фиг.17, интерфейс I2, показанный на фиг.11, частично (или полностью) можно записать внутритекстово в интерфейс I1, чтобы сформировать интерфейс I1″. Показано, что интерфейс I2 делится на I2a и I2b, и часть I2a интерфейса закодирована внутритекстово с интерфейсом I1 для формирования интерфейса I1″. Для конкретного примера, положим, что интерфейс I1,

показанный на фиг.11, осуществляет функциональный вызов square (input, output), получаемый от интерфейса I2, который после обработки значения, переданного посредством input (для возведения его в квадрат) вторым сегментом кода, передает результат возведения в квадрат обратно посредством output. В этом случае обработка, производимая вторым сегментом кода (возведение input в квадрат), может осуществляться первым сегментом кода без обращения к интерфейсу.

РАЗВЕДЕНИЕ

Коммуникация от одного сегмента кода к другому может осуществляться косвенно путем разбиения коммуникации на множественные дискретные коммуникации. Это схематически обозначено на фиг.18 и 19. Согласно фиг.18 один или несколько фрагментов связующего программного обеспечения (интерфейс(ы) разведения, поскольку они разводят функциональные возможности и/или функции интерфейса от исходного интерфейса) предусмотрены для преобразования коммуникаций на первом интерфейсе, Интерфейсе 1, чтобы согласовывать их с другим интерфейсом, в данном случае интерфейсами Интерфейсом 2A, Интерфейсом 2 B и Интерфейсом 2C. Это должно происходить, например, когда установлена база приложений, предназначенная для связи, скажем, с операционной системой в соответствии с протоколом Интерфейса 1, но затем операционная система изменилась для использования другого интерфейса, в данном случае интерфейсов Интерфейса 2A, Интерфейса 2 B и Интерфейса 2C. Дело в том, что исходный интерфейс, используемый 2-м сегментом кода, изменился так, что он перестал быть совместимым с интерфейсом, используемым 1-м сегментом кода, в результате чего используется посредник для обеспечения совместимости старого и нового интерфейсов. Аналогично, как показано на фиг.19, можно ввести третий сегмент кода с интерфейсом разведения DI1, чтобы принимать коммуникации от интерфейса I1, и с интерфейсом разведения DI2, чтобы передавать функциональные возможности интерфейса, например на интерфейсы I2a и I2b, перенастроенные на работу с DI2, но при этом получать тот же функциональный результат. Аналогично, DI1 и DI2 могут работать совместно, чтобы преобразовывать функциональные возможности интерфейсов I1 и I2, указанных на фиг.11, к новой операционной системе, в то же время обеспечивая такой же или аналогичный функциональный результат.

ПЕРЕПИСЫВАНИЕ

Еще один возможный вариант состоит в динамическом переписывании кода для замены функциональных возможностей интерфейса чем-то другим, что тем не менее приводит к тому же общему результату. Например, может иметь место система, в которой сегмент кода, представленный на промежуточном языке (например, Microsoft IL, Java ByteCode и пр.) поступает на компилятор или интерпретатор, работающий по принципу «точно вовремя» (JIT), в среде выполнения (например, обеспечиваемый инфраструктурой Net, средой выполнения Java или другими подобного типа средами выполнения). JIT-компилятор может быть прописан так, чтобы динамически преобразовывать коммуникации от 1-го сегмента кода ко 2-му сегменту кода, т.е. согласовывать их с другим интерфейсом, который может требоваться 2-му сегменту кода (либо исходному, либо другому 2-му сегменту кода). Это обозначено на фиг.20 и 21. Согласно фиг.20 этот подход аналогичен вышеописанному сценарию разведения. Это должно происходить, например, когда установленная база приложений предназначена для связи с операционной системой в соответствии с протоколом Интерфейса 1, но затем операционная система изменилась для использования другого интерфейса. JIT-компилятор можно использовать для согласования коммуникаций в

оперативном режиме от приложений установленной базы к новому интерфейсу операционной системы. Согласно фиг.21, этот подход динамического переписывания интерфейса(ов) можно применять также для динамического разложения или иного изменения интерфейса(ов).

Заметим также, что вышеописанные сценарии для достижения того же или аналогичного результата, как интерфейс через альтернативные варианты осуществления, можно также комбинировать по-разному, последовательно и/или параллельно, или с другим промежуточным кодом. Таким образом, представленные выше альтернативные варианты осуществления не являются взаимоисключающими, и их можно смешивать, подбирать и комбинировать для получения сценариев, идентичных или эквивалентных исходным сценариям, представленным на фиг.10 и 11. Заметим также, что, как и в большинстве программных конструкций, имеются другие аналогичные пути достижения тех же или сходных функциональных возможностей интерфейса, которые здесь могут не быть описаны, но тем не менее отвечают сущности и объему изобретения, т.е. следует отметить, что это, по меньшей мере, частично, функциональные возможности, представленные интерфейсом, и преимущественные результаты, обеспечиваемые им, которые обуславливают ценность интерфейса.

Хотя вышеприведенное описание приведено в отношении структурных особенностей и/или этапов способа, следует понимать, что изобретение, заданное прилагаемой формулой изобретения, не ограничивается описанными конкретными особенностями или действиями. Напротив, конкретные особенности и действия раскрыты как иллюстративные формы реализации изобретения.

## Формула изобретения

1. Компьютерно-считываемый носитель, хранящий инструкции, которые при исполнении их компьютером, реализуют программный интерфейс, содержащий первую группу функций, относящихся к передаче новой политики безопасности совокупности подсистем безопасности, причем каждая из совокупности подсистем безопасности выполнена с возможностью замены существующей политики безопасности новой политикой безопасности, и вторую группу функций, относящихся к передаче указания о готовности каждой подсистемы безопасности к реализации новой политики безопасности.

2. Компьютерно-считываемый носитель по п.1, отличающийся тем, что первая группа функций включает в себя метод, который предписывает каждой из совокупности подсистем безопасности удалить новую политику безопасности.

3. Компьютерно-считываемый носитель по п.1, отличающийся тем, что первая группа функций включает в себя метод, который инициализирует конкретную подсистему безопасности.

4. Компьютерно-считываемый носитель по п.1, отличающийся тем, что первая группа функций включает в себя метод, который предписывает каждой из совокупности подсистем безопасности реализовать новую политику безопасности.

5. Компьютерно-считываемый носитель по п.1, отличающийся тем, что первая группа функций дополнительно содержит метод, который передает новые данные, связанные с существующей политикой безопасности, по меньшей мере, одной из совокупности подсистем безопасности.

6. Компьютерно-считываемый носитель по п.1, отличающийся тем, что первая группа функций дополнительно содержит метод, который передает информацию

конфигурации, по меньшей мере, одной из совокупности подсистем безопасности.

7. Компьютерно-считываемый носитель по п.1, отличающийся тем, что вторая группа функций включает в себя метод, который указывает, реализовала ли конкретная подсистема безопасности новую политику безопасности.

8. Компьютерно-считываемый носитель по п.1, отличающийся тем, что вторая группа функций дополнительно содержит метод, который извлекает обновленные данные, связанные с конкретной политикой безопасности.

9. Компьютерно-считываемый носитель по п.1, отличающийся тем, что вторая группа функций дополнительно содержит метод, который передает новые данные, идентифицированные одной из совокупности подсистем безопасности, агенту безопасности.

10. Компьютерно-считываемый носитель по п.1, отличающийся тем, что вторая группа функций дополнительно содержит метод, который позволяет одной из совокупности подсистем безопасности запрашивать пользователя системы, содержащей совокупность подсистем безопасности.

11. Компьютерно-считываемый носитель по п.1, отличающийся тем, что, по меньшей мере, одна из совокупности подсистем безопасности реализует антивирусную услугу.

12. Компьютерно-считываемый носитель по п.1, отличающийся тем, что, по меньшей мере, одна из совокупности подсистем безопасности реализует приложение брандмауэра.

13. Компьютерно-считываемый носитель по п.1, отличающийся тем, что совокупность подсистем безопасности реализуют новую политику безопасности после того, как все подсистемы безопасности указали готовность к реализации новой политики безопасности.

14. Компьютерная система, включающая в себя один или несколько микропроцессоров и один или несколько компьютерно-считываемых носителей, содержащих инструкции для реализации одной или нескольких компьютерных программ и инструкции для реализации программного интерфейса приложения, причем одна или несколько программ используют программный интерфейс приложения для реализации политики безопасности на совокупности подсистем безопасности, причем программный интерфейс приложения содержит

первую функцию, которая передает новую политику безопасности совокупности подсистем безопасности,

вторую функцию, которая проверяет, подготовилась ли каждая из подсистем безопасности к применению новой политики безопасности, и

третью функцию, которая предписывает каждой из совокупности подсистем безопасности реализовать новую политику безопасности после того, как определено, что все подсистемы безопасности подготовились к применению новой политики безопасности.

15. Компьютерная система по п.14, отличающаяся тем, что дополнительно содержит четвертую функцию, которая инициирует удаление новой политики безопасности каждой из совокупности подсистем безопасности, если, по меньшей мере, одна из совокупности подсистем безопасности не способна применять новую политику безопасности.

16. Компьютерная система по п.14, отличающаяся тем, что дополнительно содержит четвертую функцию, относящуюся к передаче информации о событии, идентифицированной первой подсистемой безопасности, другим подсистемам

безопасности.

17. Компьютерная система по п.14, отличающаяся тем, что дополнительно содержит четвертую функцию, относящуюся к передаче информации, связанной с безопасностью, идентифицированной первой подсистемой безопасности, менеджеру событий.

18. Компьютерная система по п.17, отличающаяся тем, что менеджер событий передает информацию, связанную с безопасностью, по меньшей мере, одной из совокупности подсистем безопасности.

19. Компьютерная система по п.14, отличающаяся тем, что, по меньшей мере, одна из совокупности подсистем безопасности связана с первым типом атаки на безопасность.

20. Компьютерная система по п.19, отличающаяся тем, что, по меньшей мере, одна из совокупности подсистем безопасности связана со вторым типом атаки на безопасность.

21. Способ для осуществления политики безопасности на множестве подсистем безопасности, содержащий этапы, на которых

вызывают одну или несколько первых функций для облегчения передачи политики безопасности первой подсистеме безопасности, вызывают одну или несколько вторых функций для облегчения определения, применила ли первая подсистема безопасности политику безопасности, и вызывают одну или несколько третьих функций для облегчения передачи информации, связанной с безопасностью, от первой подсистемы безопасности ко второй подсистеме безопасности.

22. Способ по п.21, отличающийся тем, что информация, связанная с безопасностью, идентифицирует тип атаки на безопасность.

23. Способ по п.21, отличающийся тем, что дополнительно содержит этап, на котором вызывают одну или несколько четвертых функций для обеспечения взаимодействия с пользователем системы, содержащей первую подсистему безопасности.

24. Способ по п.21, отличающийся тем, что дополнительно содержит этап, на котором вызывают одну или несколько четвертых функций для обеспечения передачи информации конфигурации первой подсистеме безопасности.

25. Способ по п.21, отличающийся тем, что дополнительно содержит этап, на котором вызывают одну или несколько четвертых функций для обеспечения выдачи предписания первой подсистеме безопасности и второй подсистеме безопасности реализовать политику безопасности.

26. Способ по п.21, отличающийся тем, что дополнительно содержит этап, на котором вызывают одну или несколько четвертых функций для обеспечения передачи пересмотренной политики безопасности первой подсистеме безопасности.

27. Система для осуществления политики безопасности на множестве подсистем безопасности, содержащая

средство объявления первой функции, которая передает событие, относящееся к безопасности, менеджеру событий,

средство объявления второй функции, которая идентифицирует совокупность подсистем безопасности, связанных с событием, относящимся к безопасности, и

средство объявления третьей функции, которая передает событие, относящееся к безопасности, идентифицированным подсистемам безопасности.

28. Система по п.27, отличающаяся тем, что дополнительно содержит средство объявления четвертой функции, которая передает новую политику безопасности

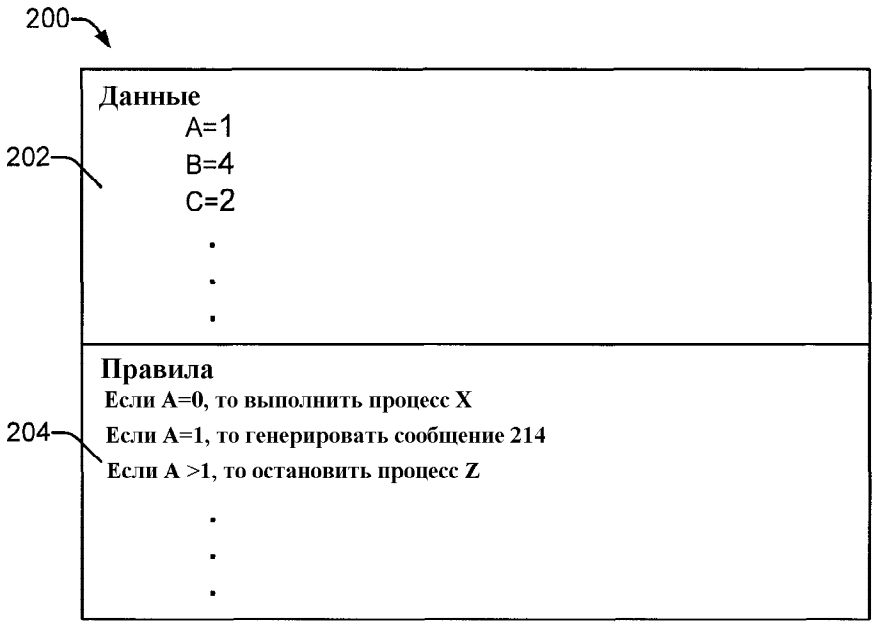
совокупности подсистем безопасности, и средство объявления пятой функции, которая предписывает совокупности подсистем безопасности заменить существующую политику безопасности новой политикой безопасности.

29. Система по п.28, отличающаяся тем, что дополнительно содержит средство объявления шестой функции, которая предписывает совокупности подсистем безопасности удалить новую политику безопасности, если, по меньшей мере, одна из совокупности подсистем безопасности не может реализовать новую политику безопасности.

30. Система по п.27, отличающаяся тем, что событие, относящееся к безопасности, представляет собой обнаружение вируса.

31. Система по п.27, отличающаяся тем, что событие, относящееся к безопасности, представляет собой попытку несанкционированного доступа к запоминающему устройству.

32. Система по п.27, отличающаяся тем, что дополнительно содержит средство объявления четвертой функции, которая извещает менеджера событий о том, что конкретная подсистема безопасности закончила обработку другого функционального вызова.



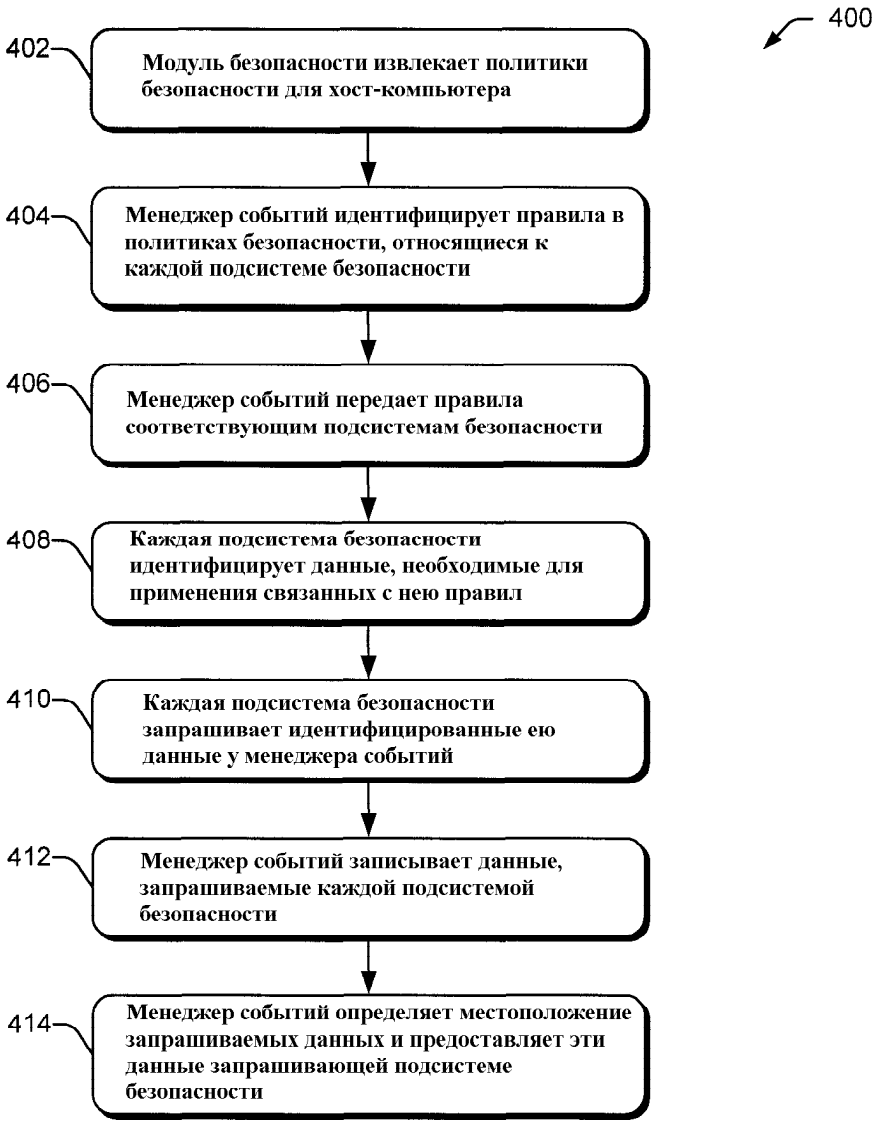

**Фиг. 2**

300

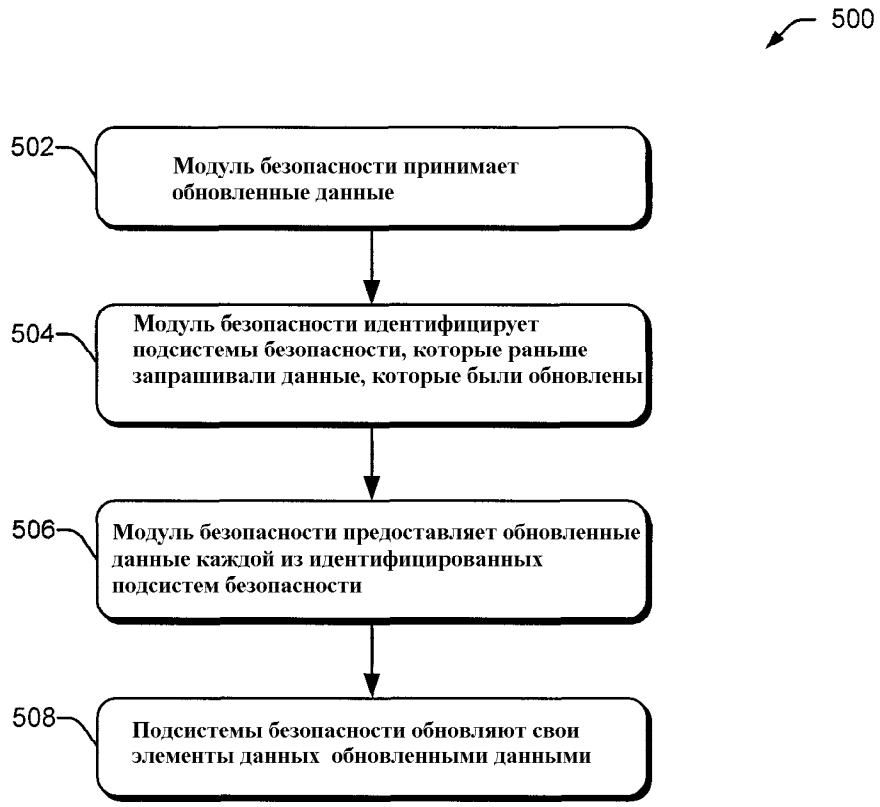
302 304

| Элемент данных | Подсистемы безопасности |
|---|---|
| A | 1 |
| B | 1, 2 |
| C | 2, 5 |
| D | 1, 4, 6 |
| E | 2, 3 |
| . | . |
| . | . |
| . | . |

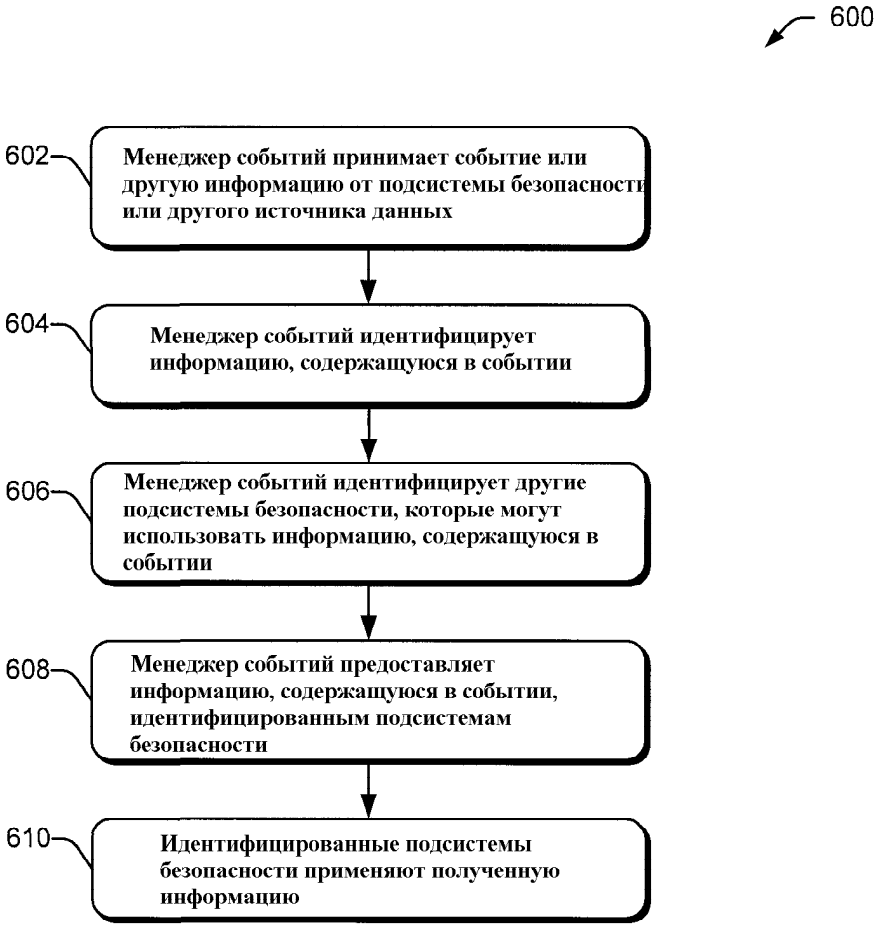
**Фиг. 3**


400
402
Модуль безопасности извлекает политики безопасности для хост-компьютера
404
Менеджер событий идентифицирует правила в политиках безопасности, относящиеся к каждой подсистеме безопасности
406
Менеджер событий передает правила соответствующим подсистемам безопасности
408
Каждая подсистема безопасности идентифицирует данные, необходимые для применения связанных с нею правил
410
Каждая подсистема безопасности запрашивает идентифицированные ею данные у менеджера событий
412
Менеджер событий записывает данные, запрашиваемые каждой подсистемой безопасности
414
Менеджер событий определяет местоположение запрашиваемых данных и предоставляет эти данные запрашивающей подсистеме безопасности

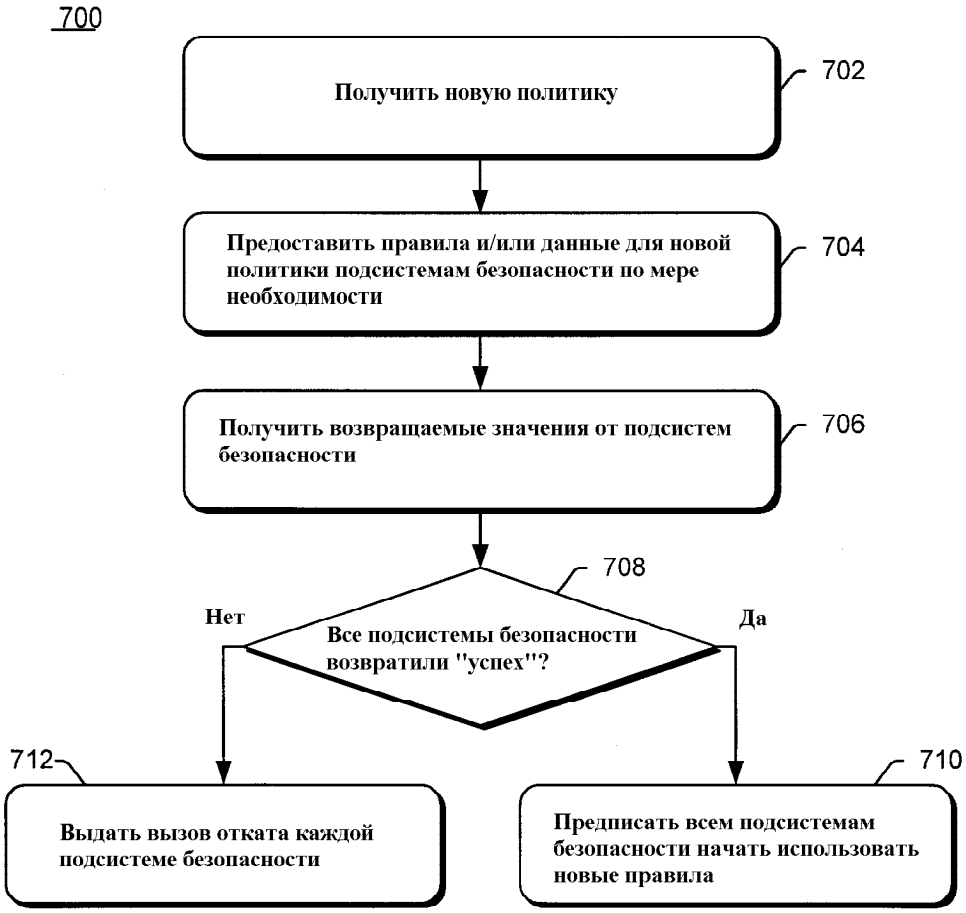
**Фиг. 4**


500
502
Модуль безопасности принимает обновленные данные
504
Модуль безопасности идентифицирует подсистемы безопасности, которые раньше запрашивали данные, которые были обновлены
506
Модуль безопасности предоставляет обновленные данные каждой из идентифицированных подсистем безопасности
508
Подсистемы безопасности обновляют свои элементы данных обновленными данными

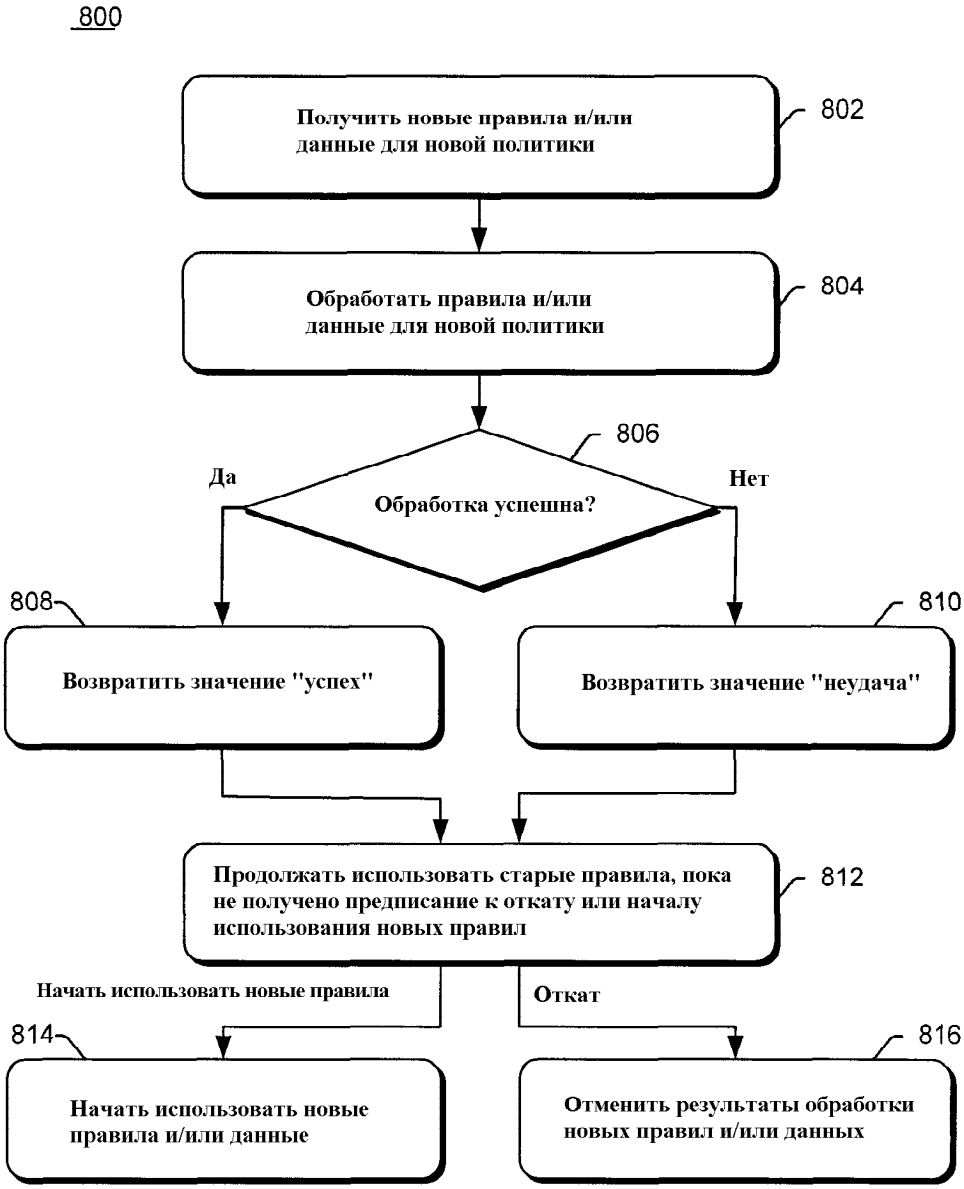
**Фиг. 5**


600
602
Менеджер событий принимает событие или другую информацию от подсистемы безопасности или другого источника данных
604
Менеджер событий идентифицирует информацию, содержащуюся в событии
606
Менеджер событий идентифицирует другие подсистемы безопасности, которые могут использовать информацию, содержащуюся в событии
608
Менеджер событий предоставляет информацию, содержащуюся в событии, идентифицированным подсистемам безопасности
610
Идентифицированные подсистемы безопасности применяют полученную информацию

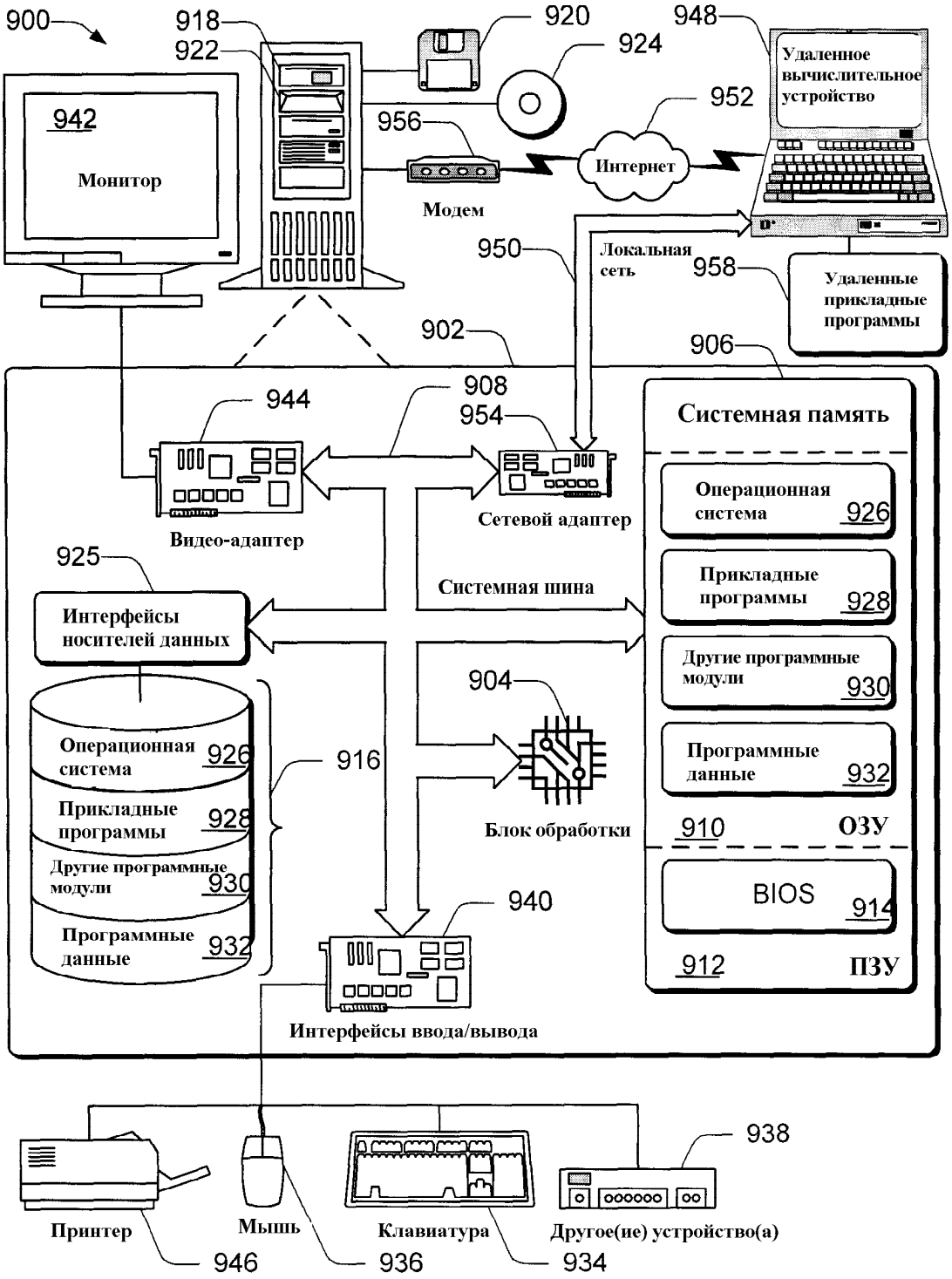
**Фиг. 6**


700
Получить новую политику
702
Предоставить правила и/или данные для новой политики подсистемам безопасности по мере необходимости
704
Получить возвращаемые значения от подсистем безопасности
706
708
Все подсистемы безопасности возвратили "успех"?
Нет
Да
712
Выдать вызов отката каждой подсистеме безопасности
710
Предписать всем подсистемам безопасности начать использовать новые правила

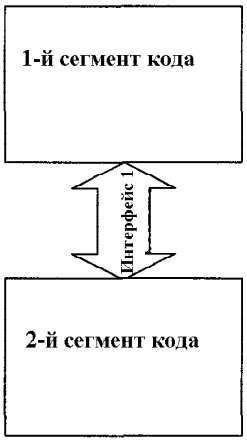
**Фиг. 7**


800
Получить новые правила и/или данные для новой политики
802
Обработать правила и/или данные для новой политики
804
806
Обработка успешна?
Да
Нет
808
Возвратить значение "успех"
810
Возвратить значение "неудача"
Продолжать использовать старые правила, пока не получено предписание к откату или началу использования новых правил
812
Начать использовать новые правила
Откат
814
Начать использовать новые правила и/или данные
816
Отменить результаты обработки новых правил и/или данных

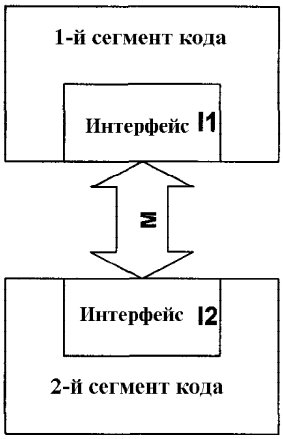
**Фиг. 8**


900
918
922
920
924
948
Удаленное вычислительное устройство
942
Монитор
956
952
Интернет
Модем
950
Локальная сеть
958
Удаленные прикладные программы
902
906
944
908
954
Системная память
Операционная система 926
Видео-адаптер
Сетевой адаптер
925
Системная шина
Прикладные программы 928
Интерфейсы носителей данных
Другие программные модули 930
904
Операционная система 926
916
Программные данные 932
Блок обработки
910
ОЗУ
Прикладные программы 928
BIOS 914
Другие программные модули 930
940
912
ПЗУ
Программные данные 932
Интерфейсы ввода/вывода
938
Принтер
Мышь
Клавиатура
Другое(ие) устройство(а)
946
936
934

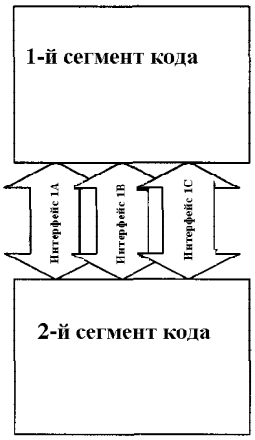
**Фиг. 9**



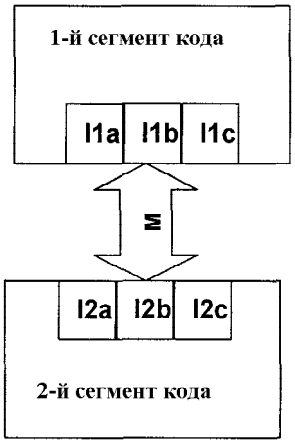

Фиг. 10



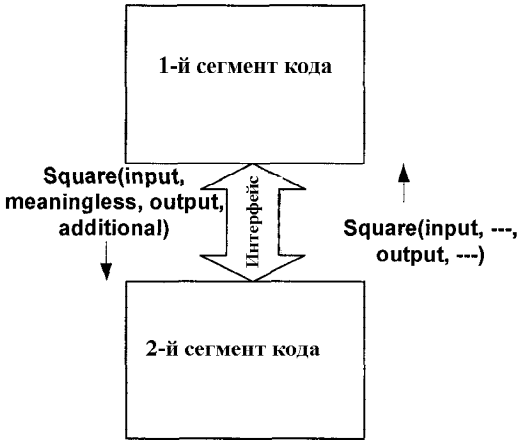

Фиг. 11



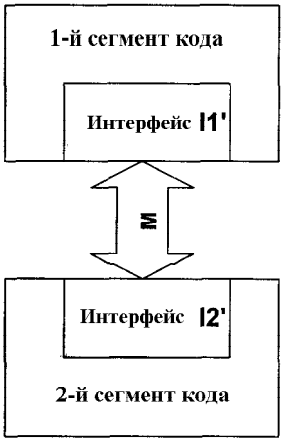

Фиг. 12



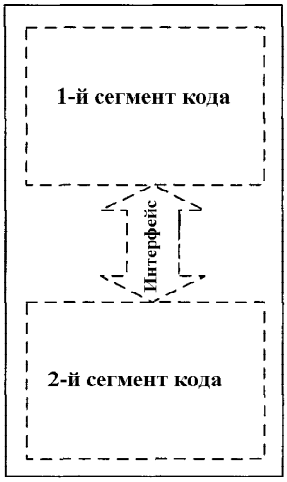

Фиг. 13



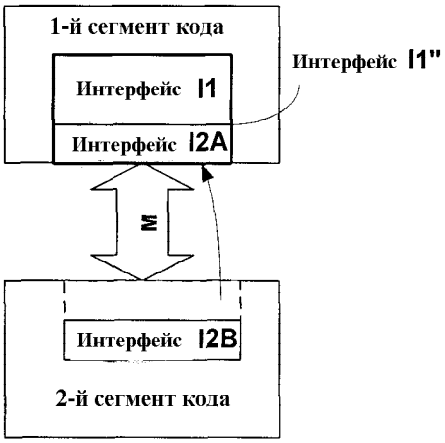

Фиг. 14



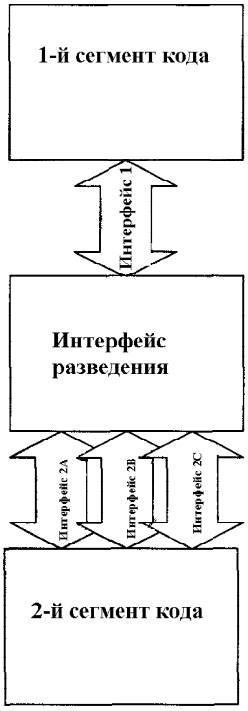

Фиг. 15



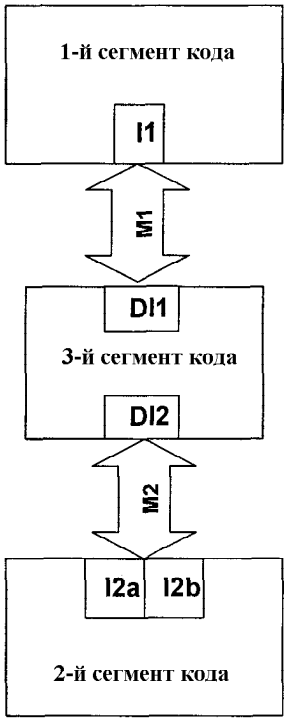

Фиг. 16



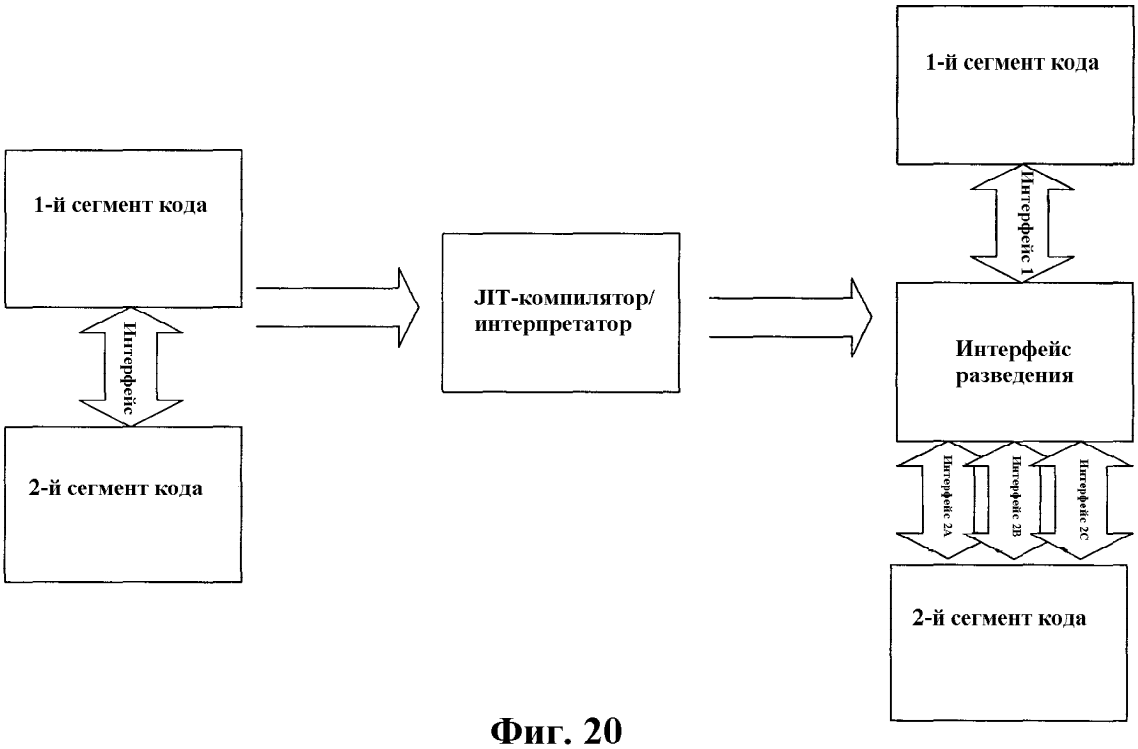

Фиг. 17



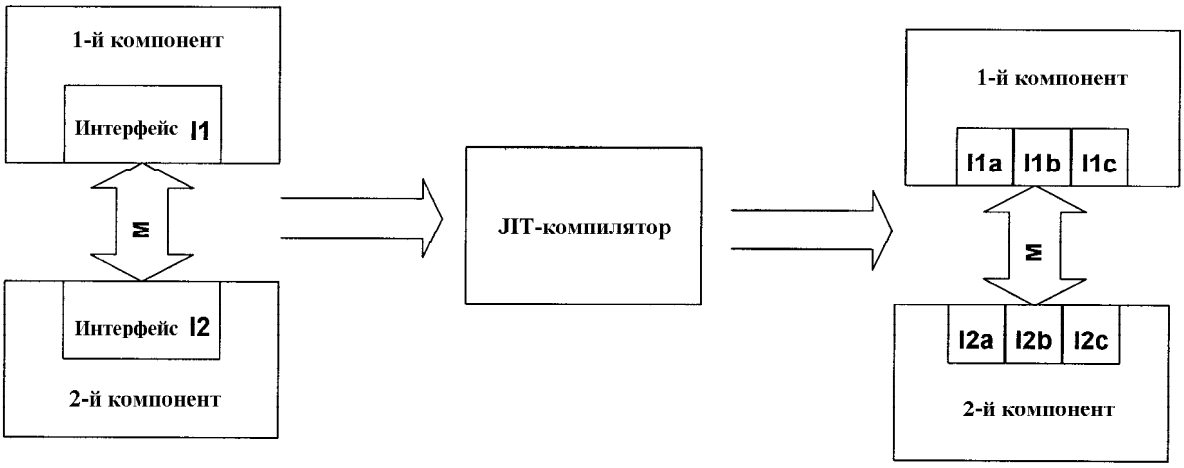

Фиг. 18


1-й сегмент кода
I1
M1
DI1
3-й сегмент кода
DI2
M2
I2a
I2b
2-й сегмент кода

Фиг. 19


1-й сегмент кода
Интерфейс
2-й сегмент кода
JIT-компилятор/
интерпретатор
1-й сегмент кода
Интерфейс 1
Интерфейс
разведения
Интерфейс 2A
Интерфейс 2B
Интерфейс 2C
2-й сегмент кода

Фиг. 20


1-й компонент
Интерфейс I1
M
Интерфейс I2
2-й компонент
JIT-компилятор
1-й компонент
I1a
I1b
I1c
M
I2a
I2b
I2c
2-й компонент

**Фиг. 21**