Защита информации и безопасность в компьютерных сетях

УДК 681.322:621.382.26:004.056.52

Необходимость обеспечения безопасности операционных систем на системном уровне

Д. С. Кулябов, А. В. Королькова

Кафедра систем телекоммуникаций
Российский университет дружбы народов
Россия, 117198, Москва, ул. Миклухо-Маклая, 6

Целью данной статьи является привлечение внимания к мандатным моделям безопасности операционных систем. Показана недостаточность обеспечения защиты системы на прикладном уровне и необходимость развития направлений, обеспечивающих мандатную безопасность на системном уровне.

Ключевые слова: MAC, дискреционная модель, мандатная модель, SELinux, RSBAC.

1. Введение

Любой человек, работающий в сфере информационных технологий, понимает необходимость обеспечения безопасности операционных систем. Но попытки обеспечения безопасности страдают от неверного предположения, что адекватная степень защиты может быть достигнута на прикладном уровне совместно с механизмами защиты, существующими в современных операционных системах. Основные усилия направлены на усиление контроля доступа и применение криптографии, что создает ложную уверенность в защищенности системы (например, The Next-Generation Secure Computing Base (NGSCB) [1]). В реальности же необходимо обеспечивать защиту на системном уровне, т.е. развивать т.н. защищенные операционные системы.

Необходимость наличия встроенных средств защиты на уровне операционной системы не вызывает сомнений. Операционная система обеспечивает защиту механизмов прикладного уровня от неправильного использования, обхода или навязывания ложной информации. Если она не сможет удовлетворить этим требованиям, появятся уязвимости в масштабах всей системы.

Именно этим вопросам посвящена данная статья, являющаяся по сути кратким обзором текущего состояния дел и основных направлений разработки в этой области за последние несколько лет.

2. Принцип минимальных привилегий

Основным путем повышения безопасности приложения является сокращение привилегий. Под *привилегиями* обычно понимается набор разрешений, установленных для исполнения каких-либо действий. Существует три пути минимизации привилегий:

— назначение привилегий для части программы,
— назначение только абсолютно необходимых привилегий,
— ограничение времени, в течение которого действуют привилегии.

Назначение привилегий для части программы. Программный продукт разбивается на несколько частей, которым назначаются разные привилегии. Реализации могут быть разные, например:

— разделение программного продукта на frontend и backend (данный подход хорошо укладывается в идеологию Unix и очень часто применяется);
— построение программного продукта по многоуровневой схеме (частный случай — клиент–серверная архитектура).

Назначение только абсолютно необходимых привилегий. Этот подход широко применяется во многих операционных системах. Для Unix-подобных ОС, например, принято назначать приложениям собственные идентификаторы пользователя и группы.

Также возможно задавать привилегии и на другие системные ресурсы, например, с помощью POSIX capabilities.

Весьма интересным подходом является использование системного вызова `chroot()` в Unix-подобных ОС. Он представляет из себя рудиментарные возможности управления *пространством имен (namespace)*.

Ограничение времени привилегий. При данном подходе приложение получает необходимые привилегии только на то время, которое необходимо для выполнения привилегированного действия.

В качестве примера можно рассмотреть проблему открытия TCP-портов, меньших 1024, для доступа к которым необходимы административные привилегии. Данные привилегии даются только на время связи сокета с таким портом.

3. Типы разграничения доступа

При построении классификации (или таксономии) типов разграничения доступа выделяют мандатные и дискреционные типы доступа. В TCSEC [2,3] приведено довольно узкое определение мандатной безопасности, которое тесно связано с политикой многоуровневой безопасности Министерства Обороны США.

Однако это определение является недостаточным для удовлетворения требованиям ни Министерства Обороны, ни частного производства, т.к. оно игнорирует такие важные качества, как непередаваемость и динамическое разделение обязанностей [4]. Вместо этого в данной статье употребляются более общие определения мандатной и дискреционной политик безопасности, данные в [5]. Это определение фактически сводит дискреционную политику безопасности к модели Харрисона-Руззо-Ульмана, а мандатную — к модели Белла-ЛаПадула. В результате такие политики, как ролевая и DTE просто выпадают из этой классификации (что уменьшает ее полезность).

Мандатной политикой безопасности будем считать любую политику, логика и присвоение атрибутов безопасности которой строго контролируются системным администратором безопасности.

Дискреционной политикой безопасности будем считать любую политику, в которой обычные пользователи могут принимать участие в определении функций политики и/или присвоении атрибутов безопасности.

3.1. Практические модели доступа

Ролевая политика безопасности представляет собой существенно усовершенствованную модель Харрисона-Руззо-Ульмана [3] (управление доступом в ней осуществляется как на основе матрицы прав доступа для ролей, так и с помощью правил, регламентирующих назначение ролей пользователям и их активацию во время сеансов).

Политика доменов и типов (Domain and Type Enforcement — *DTE*) — это модель, являющаяся расширением типизированной матрицы доступа, в которой типы приписаны не только объектам, но и субъектам.

С помощью мандатной безопасности можно реализовать политики безопасности на уровне предприятия. В некоторых статьях авторы ссылаются на эту концепцию как на недискреционную безопасность в рамках контроля доступа на базе ролевой модели [4] и модели соответствия типов [6].

В данном случае дискреционная безопасность не является синонимом контроля доступа основанного на идентификаторе; *IBAC (Identificator Based Access Control)*, равно как и любая другая политика безопасности, может быть либо мандатной, либо дискреционной [7].

4. Дискреционные механизмы безопасности

Дискреционный механизмы не могут противостоять неаккуратному пользователю или злоумышленнику, так как переносят все бремя обеспечения безопасности на пользователя, халатность которого в любой момент времени может привести к нарушению политики (в отличие от мандатных механизмов безопасности, которые перекладывают бремя обеспечения безопасности на администратора политики безопасности).

При использовании только лишь дискреционных механизмов безопасности, злоумышленник, имеющий доступ к конфиденциальной информации и приложениям, может напрямую раскрыть конфиденциальную информацию в нарушение политики. Хотя авторизованный, но неаккуратный пользователь также может осуществить утечку конфиденциальной информации путями, не затрагивающими компьютерную систему, сама возможность использования компьютерных каналов утечки информации может увеличить размеры утечки и уменьшить эффективность ее обнаружения и прослеживания. При наличии же мандатных механизмов защиты утечка конфиденциальной информации возможна лишь через скрытые каналы, что ограничивает размеры утечки и обеспечивает контроль за ней, при условии аудита скрытых каналов.

Если для применения политики безопасности используются только дискреционные механизмы, то она может быть нарушена некорректным или злоумышленным приложением (даже если пользователи благонадежны и аккуратны). В данном случае разница между некорректным и злоумышленным приложениями не так уж важна. В любом случае приложение может не использовать механизмы безопасности, требуемые политикой, или использовать их путем, несоответствующим намерениям пользователя.

Для обеспечения гарантий того, что механизмы безопасности были использованы правильно и они могут защитить пользователя от случайного запуска ненадежного приложения, необходимо использовать мандатные механизмы защиты. Даже если пользователь тщательно определит всю дискреционную политику для корректной реализации защиты, приложение может самостоятельно изменить дискреционную политику без согласования с пользователем. В то время как мандатная политика может быть изменена только лишь системным администратором безопасности.

5. Мандатная политика безопасности

5.1. Типы политик мандатной безопасности

Можно выделить три типа мандатной политики безопасности операционной системы:

— политика контроля доступа, которая определяет, каким образом субъекты могут получить доступ к объектам под контролем операционной системы;

— политика использования подсистемы идентификации и аутентификации, которая указывает, какие механизмы идентификации и аутентификации должны быть используется для получения доступа пользователя к системе;
— политика использования криптографической подсистемы, которая указывает, какие криптографические механизмы должны быть использованы для защиты данных в системе.

Дополнительно, различные подсистемы операционной системы могут иметь собственные политики использования механизмов данных подсистем. Эти политики могут зависеть от политики использования криптографической подсистемы.

5.2. Скрытые каналы утечки информации

Защищенная операционная система должна предоставлять набор средств для задания мандатной политики безопасности операционной системы и перевода этих определений в форму, понятную для низлежащих механизмов обеспечения мандатной безопасности. Если подобный набор средств отсутствует, то нет уверенности в том, что механизмы обеспечения мандатной безопасности смогут предоставить желаемые характеристики безопасности. Тем не менее, даже операционная система с мандатной защитой может страдать от наличия высокоскоростных скрытых каналов утечки информации [8–12]. Особенно это проявляется при использовании мандатной политики безопасности для обеспечения конфиденциальности.

Теоретический подход. Существование скрытых каналов и невозможность их устранения при традиционном подходе к построению информационных систем можно объяснить следующим образом. Формальные модели безопасности, такие как известная модель Белла-ЛаПадула, разграничивают доступ «в принципе», но не содержат понятия времени и не регламентируют конкуренцию за ресурсы (т. е. возможны ситуации, когда «в принципе ресурс использовать можно, но не в данный момент, поскольку данный ресурс занят»). При любом распределении прав доступа разного рода сигнальные события и, в частности, коллизии вследствие конкуренции могут быть использованы для организации скрытых каналов. Таким образом, принципиально невозможно полностью блокировать утечку конфиденциальной информации универсальными средствами, не учитывающими семантику программ.

Практический подход. В повседневной практике скрытые каналы трактуются шире, чем в теории. Расширение происходит по четырем направлениям.

1. Рассматриваются системы с произвольной дисциплиной управления доступом (а не только с многоуровневой политикой безопасности).

   В повседневной практике скрытые каналы чаще всего возникают из-за возможности доступа к данным несколькими способами. Например:
   — если в корневом каталоге файловой системы располагается tftp-сервер, то он позволит получить всем пользователям доступ на чтение ко всем файлам, независимо от установленных прав доступа;
   — если есть программа с установленным битом переустановки действующего идентификатора пользователя, владельцем root и уязвимостью, то обычный пользователь может проэксплуатировать уязвимость, захватить привилегии суперпользователя и нарушить конфиденциальность и целостность любого элемента системы;
   — пароль в незашифрованном виде, хранящийся в оперативной памяти и сбрасываемый в файл с образом памяти при аварийном завершении.
2. Рассматриваются не только нестандартные каналы передачи информации, но и нестандартные способы передачи информации по легальным каналам (потайные каналы).

- Нестандартные способы передачи информации по легальным каналам получили название «подсознательных» или потайных каналов (subliminal channels). Потайные каналы используют тогда, когда имеется легальный коммуникационный канал, но политика безопасности запрещает передавать по нему определенную информацию; иными словами, информацию передавать можно, но она не должна выглядеть подозрительно (в соответствии с некими, обычно не очень четкими критериями). Подобные каналы, используемые для управления, должны быть двунаправленными.
- Потайной канал возможен тогда, когда в передаваемых легальных данных есть незначащие или почти незначащие биты, например, некоторые биты в заголовках IP-пакетов или младшие биты интенсивности цветов в графическом файле, присоединенном к письму. (Электронная почта — идеальный легальный канал, на который удобно накладывать скрытые).

3. Рассматриваются угрозы не только конфиденциальности, но и целостности.

   Для нарушения целостности чаще всего используют уязвимости, связанные с переполнением буферов. В данном случае примером скрытого канала, основанного на возможности доступа к данным нестандартным способом, может служить изменение содержимого стека вызовов.

   Применяются также атаки пользовательских процессов на целостность транзакций, выполняемых процессами привилегированными (соответствующий англоязычный термин — race condition). Возможность подобной атаки появляется, если временные данные для транзакции располагаются в общедоступных каталогах, таких как /tmp.
4. Рассматриваются не только однонаправленные, но и двунаправленные каналы.

Методы борьбы со скрытыми каналами. Разумеется, при расширительном толковании понятия скрытого канала проблемы, описанные выше, не только остаются, но и обостряются; кроме того, к ним добавляются новые. С этими проблемами можно бороться двумя способами: формальным и содержательным.

Формальный способ усиления безопасности состоит в попытке добавить к употребительным операционным системам, таким как Linux, возможности, реализующие требования классов B2 и старше из «Оранжевой книги» [2], то есть реализовать в ядре ОС мандатный доступ ко всем ресурсам и провести анализ скрытых каналов. Вне зависимости от выполнимости последнего пункта, подобный путь представляется тупиковым. Одно дело анализировать скрытые каналы в операционной системе мэйнфрейма, изначально спроектированной с учетом требований безопасности и прошедшей многолетнюю апробацию, и совсем другое — в ОС, содержащей ошибки, связанные с переполнением буферов. Для подлинной безопасности нужно не только добавление принудительного управления доступом, но и существенное перепроектирование ОС.

Содержательный способ борьбы со скрытыми каналами состоит в выстраивании эшелонированной обороны: утечки информации признаются неизбежными, но их пытаются локализовать и отследить.

Еще один практически важный в данном контексте архитектурный принцип — разнесение доменов выполнения для приложений с разным уровнем доверия на разные узлы сети. Если такие приложения будут функционировать в распределенной среде клиент–сервер, ограничить их взаимное влияние (и, следовательно, заблокировать скрытые каналы) будет существенно проще, чем в случае единых многопользовательских систем.

Таким образом, можно сделать следующий вывод: даже при наличии скрытых каналов утечки операционная система, имеющая самые простые мандатные механизмы, повышает уровень безопасности системы, требуя большей квалификации от злоумышленника. Как только операционные системы с простыми мандатными механизмами станут использоваться повсеместно, использование скрытых каналов утечки станет обыденной практикой, что в свою очередь подогреет общественный интерес к необходимости исследования проблемы скрытых каналов в компьютерных системах.

5.3. Разграничение доменов безопасности

Доверенными приложениями будем называть приложения, которым требуются специальные привилегии для выполнения некоторых функций, связанных с безопасностью, но также существует уверенность в том, что такие приложения корректно выполнят возложенные на них функции по обеспечению безопасности и не будут использовать во вред привилегии, предоставленные им для выполнения этих функций.

5.3.1. Принцип минимальных привилегий

Если механизмы мандатной безопасности операционной системы поддерживают только упрощенное разделение привилегий, то безопасность всей системы может свестись к обеспечению безопасности доверенных приложений в операционной системе. Для снижения зависимости от доверенных приложений, механизмы мандатной безопасности операционной системы должны быть спроектированы с учетом поддержки *принципа минимальных привилегий* (соответствие типов является примером такого механизма мандатной безопасности, который может быть использован и для ограничения набора привилегий, выделяемого доверенным приложениям для выполнения их функций, и для ограничения возможного ущерба, вызванного любым злоумышленным использованием данных привилегий).

Механизмы мандатной безопасности операционной системы могут быть использованы для поддержки в приложениях функций, связанных с безопасностью, только в том случае, если строго гарантируется, что данные механизмы невозможно обойти и обмануть.

Например, соответствие типов может быть использовано для реализации защищенных каналов взаимодействия, необходимых для обеспечения такой функциональности. Защищенный канал взаимодействия гарантирует, что данные, передаваемые от определенного отправителя к определенному получателю, пройдут через подсистему безопасности. Кроме того, защищенный канал гарантирует целостность самой подсистемы. Большинство требований безопасности таких приложений может быть удовлетворено с помощью механизмов мандатной безопасности операционной системы.

5.3.2. Надежный путь доступа

Надежный путь доступа — это механизм, посредством которого пользователь может взаимодействовать с доверенным ПО напрямую, и который может быть активирован пользователем или доверенным ПО, но не может быть воспроизведен каким-либо другим ПО.

При отсутствии механизма надежного пути доступа, вредоносное ПО может выдать себя за доверенное ПО пользователю или выдать себя за пользователя для доверенного ПО. Такое вредоносное ПО потенциально может получить доступ к конфиденциальной информации, совершать действия от лица пользователя, нарушающие намерения пользователя, или обманывать пользователя, заставляя его думать, что запрошенная функция выполнена, в то время как она не вызывалась.

В дополнение к поддержке доверенного ПО в базовой системе, механизм надежного пути доступа должен быть расширяемым для поддержки последующих добавлений доверенных программ системным администратором политики безопасности.

Концепция надежного пути доступа может быть обобщена и подразумевать не только взаимодействия непосредственно между пользователями и доверенным ПО. Надежный сетевой канал (Trusted Network Interface — TNI) представляет концепцию надежного (безопасного) канала связи между доверенным ПО на различных узлах сети. В общем, механизм, который гарантирует взаимноаутентифицирующий канал связи, защищенный туннель, необходим для обеспечения того, что важные системные функции не будут введены в заблуждение.

Несмотря на то, что механизм защищенного канала связи может быть построен на прикладном уровне, все же для операционной системы желательно предоставлять собственные механизмы защищенных каналов связи, так как такие механизмы проще проверить [5] и они, скорее всего, более эффективны.

5.3.3. Политика доменов и типов

Механизмы мандатной безопасности операционной системы также могут быть использованы для строгой изоляции приложения в рамках уникального домена безопасности, который тщательно отделен от остальных доменов системы. Даже в этом случае приложение может совершить какие-либо несанкционированные действия, однако возможный ущерб от этих действий будет ограничен рамками одного домена безопасности. Эта ограничивающая функция является очень важной для контроля потоков данных при поддержке политики безопасности системы. В дополнение к поддержке безопасного исполнения ненадежного ПО, ограничивающая функция может поддерживать функциональные требования, такие как изолированная тестовая среда или замкнутая среда разработки.

С формальной точки зрения *политика доменов и типов* (Domain and Type Enforcement — *DTE*) —это модель, являющаяся расширением типизированной матрицы доступа, в которой типы приписаны не только объектам, но и субъектам.

С точки зрения реализации DTE представляет собой надстройку над ядром ОС (имеются в виду UNIX-подобные системы), которая осуществляет контроль доступа со стороны процессов (субъектов, представляющих интересы пользователей) к объектам системы (к файлам, сообщениям, сегментам общей памяти, семафорам и т.д.). Каждому объекту присвоен некоторый тип, а с каждым субъектом связан определенный домен (*домен* — совокупность субъектов, имеющих одинаковые полномочия). Контроль доступа субъектов к объектам осуществляется на основании правил, задающих отношения доступа между доменами и типами.

В процессе работы система контроля доступа перехватывает вызовы, анализирует их и блокирует запросы к ядру ОС, если полномочия, необходимые для их осуществления, отсутствуют в таблицах DTE.

Такой механизм контроля доступа является достаточно сильным и гибким, но на практике возникают следующие нюансы.

— Политика DTE предназначена для систем со значительным количеством доменов, для которых используются значительные ограничения доступа, что, в свою очередь, приводит к увеличению размеров таблицы прав доступа и, как следствие, к неуправляемости системы в целом.

  Решением проблемы может служить предусмотрение возможности как индивидуального, так и группового назначения доменов и типов, а также параметров доступа по умолчанию.

— Не существует естественного соответствия между таблицами DTE и стандартными системными структурами ОС. Таблицы прав доступа не учитывают положение субъектов и объектов в системных иерархиях ОС и не опираются на дерево порождения процессов и структуру каталогов файловых систем (хотя именно положение сущностей в этих структурах и определяет их атрибуты безопасности, т.к. процесс наследует домен своего родителя, а в каталоге группируются файлы, принадлежащие одному типу). Кроме того, необходимость хранения типа для каждого файла требует пересмотра форматов служебных структур файловой системы, что приводит к несовместимости с существующими системами.

  Решением проблемы является хранение атрибутов безопасности субъектов и объектов в неявном виде в спецификации политики, которая описывается на специально разработанном для этой цели языке DTEL [13]. Такое решение позволяет создавать многократно используемые типовые конфигурации управления доступом, рассчитанные на решение определенных прикладных задач.

Таким образом, задача усовершенствования защиты ОС может быть решена без ее существенной модификации, усложнения администрирования и потери совместимости между существующими системами и приложениями. UNIX-подобные

системы имеют четкую архитектуру, основанную на простых и четко определенных концепциях, таких как иерархия процессов, связанных отношением наследственности, и взаимодействующих с ядром и между собой с помощью стандартизованного множества вызовов, а также представление всех системных ресурсов в виде иерархии файлов и каталогов. Использование для описания спецификаций политики специального языка позволяет составлять типовые шаблоны прикладных политик, которые можно использовать в различных UNIX-подобных системах и настраивать с помощью макроподстановок. Отказ от хранения атрибутов безопасности для каждого объекта и субъекта позволяет сохранить совместимость с существующими форматами данных и файловых систем и существенно сократить объемы обрабатываемой информации.

6. Классические механизмы для обеспечения безопасности на прикладном уровне

6.1. Контроль доступа

6.1.1. Механизмы контроля доступа

Механизмы контроля доступа прикладного уровня могут быть разбиты на две компоненты:

— перехватывающую,
— решающую.

Когда субъект пытается осуществить доступ к объекту, защищенному механизмом контроля доступа, перехватывающая компонента должна вызвать решающую компоненту, передав ей соответствующие входные параметры для принятия решения в соответствии с политикой безопасности, и должна претворить в жизнь принятое решение.

Обычно необходимыми входными параметрами являются атрибуты безопасности субъекта и объекта. Решающая компонента может также запрашивать другие внешние источники для принятия решения в соответствии с политикой безопасности (например, она может использовать внешнюю базу данных политики безопасности и системную информации, такую как текущее время).

Если программа-агент злоумышленника сможет влиять хотя бы на одну из компонент механизма контроля доступа или на входные данные для механизма принятия решения, то она может нарушить работу всего механизма контроля доступа. Даже если все компоненты и все входные данные механизма контроля доступа расположены в рамках одного файла, все равно целостность файла зависит от механизмов защиты операционной системы. Только мандатные механизмы безопасности могут четко предоставить гарантии обеспечения целостности.

Если авторизованный пользователь запускает вредоносное ПО, то оно может изменить атрибуты безопасности какого-либо объекта или правила политики безопасности без уведомления или согласования с пользователем (даже при наличии строгих гарантий обеспечения целостности входных данных политики безопасности). Механизму контроля доступа необходимо использовать механизм надежного пути доступа, предоставляемого операционной системой, для того, чтобы гарантировать, что случайное распространение (передача) прав доступа пользователя не может быть произведено без явного согласования с пользователем.

Если программа-агент злоумышленника сможет выдать себя за решающую компоненту и таким образом обмануть перехватывающую компоненту, или если программа-агент злоумышленника сможет подменить источник данных для принятия решения, то тогда она может нарушить работу всего механизма. Если любая из компонент механизма или внешние источники данных для принятия решения расположены не в рамках одного приложения, то механизму контроля доступа требуется использовать механизм надежного пути доступа.

Если программа-агент злоумышленника сможет обойти перехватывающую компоненту, тогда нарушить работу механизма контроля доступа еще проще. Механизмы мандатной безопасности операционной системы могут быть использованы

для гарантии того, что все попытки доступа к защищаемым объектам проходят через перехватывающую компоненту.

6.1.2. Примеры уязвимостей

Мобильный код. В настоящее время разработано несколько независимых решений для WWW, каждое со своей моделью защиты, обеспечивающих защиту от угроз злоумышленного мобильного кода. Тем не менее, системы, использующие эти решения, все равно остаются уязвимы из-за недостаточной поддержки функций безопасности со стороны операционной системы.

Основной угрозой, с которой пытаются бороться все существующие решения, является угроза получения вредоносным мобильным кодом несанкционированного доступа к файлам и ресурсам пользователя с целью нарушения целостности или конфиденциальности. Эта угроза не ограничивается только интерпретируемыми апплетами, загружаемыми из сети веб-броузером (эта модель угроз распространяется на вспомогательные приложения, которые могут активно устанавливаться пользователем). Разница между мобильным кодом и тем, что обычно называют данными, невелика. Соответственно, вспомогательные приложения, которые работают с ненадежными данными, либо должны исполняться в ограниченном режиме использования, либо должны быть тщательно ограничены операционной системой.

Jail. Jail представляет собой системный примитив операционной системы FreeBSD (начиная с версии 4.0 и выше), изначально предназначенный для повышения безопасности системы. Его можно рассматривать как усовершенствование системного вызова `chroot()`.

На основе этого примитива можно построить набор т.н. jail-сред, каждая из которых представляет собой приблизительный аналог отдельно стоящего сервера с удаленным доступом (VPS — virtual private servers), изнутри которого доступ к физическим ресурсам компьютера и другим подобным средам ограничен.

Jail предназначен для ограничения возможностей группы процессов (то есть, если при создании процесс попадает в такого рода jail, то он ограничивается в правах и не может воздействовать на другие процессы вне своей группы, а также не может использовать некоторые системные вызовы и имеет некоторые другие ограничения).

Идея его использования состоит в создании т.н. песочницы (sandbox), внутри которой запускаются потенциально подверженные слому программы. При осуществленном удаленном сломе программы (хакерском проникновении) удается попасть только внутрь этой jail-среды. В отличие от обычной организации Internet-серверов, когда слом сервиса позволяет получить практически все полномочия в системе (при его работе под root-пользователем), в системе с jail взломщик обязан уже повторно осуществлять проникновение уже изнутри jail-системы. Таким образом, jail изначально ориентирован на осуществление функций безопасности для сетевых серверов (о чем говорит и такая специфическая деталь как возможность ассоциации с одной jail-средой только одного сетевого адреса).

Технически jail состоит из двух частей: программы, работающей в пространстве пользователя (jail) и системного вызова ядра FreeBSD jail. Пользовательская программа просто передает параметры системному вызову ядра, при запуске которого и осуществляются требуемые ограничения. Ниже приводится описание параметров системного вызова:

- номер версии структуры, который предназначен для контроля за возможным расхождением версии пользовательской части и части ядра;
- путь, т.е. адрес директории файловой системы, куда будет указывать root файловой системы для среды (аналогично системному вызову chroot);
- имя хоста, т.е. способ отличить одну jail-среду от другой;
- ip-адрес, который можно использовать внутри среды для общения с внешним миром.

Ядро ОС хранит для каждого процесса внутри jail-среды признак того, что он является «пленником». То есть, этот признак может быть проверен при выполнении

некоторых действий. Пространство имен практически всех параметров ОС является общим, то есть, например, все номера процессов (PID) являются уникальными и не может быть двух одинаковых процессов внутри разных jail-сред.

Решения на базе jail-сред обладают следующими недостатками:

— практически полное отсутствие контроля за распределением ресурсов между разными средами дает возможность осуществления DoS атак из одной jail среды на остальные;
— нет безопасной возможности использования многих существенных возможностей операционной системы внутри jail;
— отсутствует изоляция по производительности, то есть нельзя организовать гарантии получения времени CPU для процессов среды;
— отсутствует разделение в оперативной памяти кода исполняемых файлов, запущенных из одного дерева в случае использования UnionFS (UnionFS — файловая система, используемая jail);
— jail-среда с точки зрения удаленного пользователя существенно не эквивалентна удаленному серверу FreeBSD, особенно с точки зрения администрирования;

Следует заметить, что пока в Unix-подобных системах не появится реально действующие средства управления пространством имен (подобно ОС Plan9), средства типа `chroot()` и jail останутся экзотическим и неудобным инструментарием.

JAVA. Базовая модель безопасности Java (подобно Jail) основана на концепции «песочницы». Система полагается на безопасность использования типов в языке совместно с менеджером безопасности Java для предотвращения несанкционированного доступа [14]. В настоящее время предпринимаются усилия по добавлению дополнительных функций безопасности в Java, таких как привилегии, расширенная модель контроля доступа или дополнительный контроль над доступом к определенным библиотекам классов [15]. Фундаментальным ограничением этих подходов является то, что ни один из них не гарантирует невозможность обмана или обхода. Например, несмотря на то, что сам язык Java якобы является безопасным, виртуальная машина Java (JVM) исполнит код, который будет нарушать семантику языка и может привести к нарушениям безопасности [16]. Ошибки в реализации JVM привели к нарушениям семантики языка [17]. Значительная часть системы Java сейчас существует в виде «родных» методов, реализованных в виде объектного кода (для конкретной платформы), и не подлежит проверке соответствия типов данных со стороны JVM. JVM не может защитить себя от воздействия других приложений. Наконец, модель безопасности Java не может предоставить никакой защиты от многих других форм злоумышленного мобильного кода. Даже если бы проблемы с JVM не существовало, эти решения по обеспечению безопасности все равно будут страдать от фундаментального ограничения, в соответствии с которым они полагаются на обеспечение безопасности с помощью контроля доступа на прикладном уровне. Они все зависят от локальной файловой системы, сохраняющей целостность кода системы, включая файлы классов. Все системы, хранящие политику безопасности локально, зависят от контроля доступа к файлам файловой системы, сохраняющей целостность файлов политики. В разделе, посвященном контролю доступа, показана важность свойств защищенной операционной системы для поддержки контроля доступа на прикладном уровне.

Вместо того, чтобы реализовывать свои примитивы безопасности на прикладном уровне, где они являются уязвимыми, системы мобильного кода могут использовать сервисы защищенной системы, получив в итоге систему более защищенную. Правильно разработанная защищенная система предоставит гибкую и практичную базу с одной используемой непротиворечивой моделью безопасности для различных виртуальных машин.

6.2. Криптография

Еще одним путем повышения безопасности является применение криптографических методов.

Анализ криптографической защиты на прикладном уровне может быть разбит на:

— анализ вызова механизма криптографической защиты,
— анализ самого механизма.

Сделаем следующие предположения.

1. Механизм криптографической защиты является аппаратным и реализует все необходимые криптографические функции корректно.
2. Существуют защищенные методы, с помощью которых криптографические ключи помещаются в это устройство.

Даже в этом упрощенном варианте, когда конфиденциальность и целостность алгоритмов и ключей достигается без поддержки операционной системы, все еще существуют угрозы безопасности, которые эффективно обходятся только с помощью возможностей, предоставляемых защищенными операционными системами.

6.2.1. Угрозы безопасности

1. Отсутствие гарантий использования аппаратного шифратора.
   Любая законная попытка использования шифратора может не привести к обращению к шифратору. Приложение, осуществляющее вызовы криптографических функций, может быть обойдено или модифицировано злоумышленным ПО или пользователем. Вредоносное ПО может выдавать себя за шифратор приложению, обратившемуся к нему.
2. Злоупотребление аппаратным шифратором.
   Злоупотребление включает в себя использование сервиса, алгоритма, сессии или ключа неавторизованным приложением. Без поддержки идентификации вызывающих абонентов со стороны операционной системы, аппаратный шифратор может сделать гораздо больше, чем просто потребовать от пользователя инициализировать его, после чего любой сервис, алгоритм, сессия или ключ, авторизованные для этого пользователя, могут быть использованы любым приложением в системе. В этом случае, аппаратным шифратором может воспользоваться вредоносное приложение, действующее от лица авторизованного пользователя. Более того, если аппаратный шифратор не имеет прямого физического интерфейса для активации пользователем, вредоносное ПО может выдать себя за шифратор и получить аутентификационную информацию и в последствии активировать аппаратный шифратор без уведомления или запроса пользователя. Даже при наличии прямого физического интерфейса с пользователем, аппаратный шифратор не имеет возможности запрашивать у пользователя подтверждения каждой криптографической операции.

6.2.2. Мандатная безопасность как средство борьбы с угрозами безопасности

1. Отсутствие гарантий использования аппаратного шифратора.
   Мандатная безопасность и механизм надежного пути доступа в операционной системе противостоят этой угрозе. Механизмы мандатной безопасности могут быть использованы для гарантии того, что приложение, которое обращается к аппаратному шифратору, невозможно обойти и оно устойчиво к некорректным воздействиям как злоумышленного ПО, так и пользователей. Невозможность обхода может быть достигнута путем использования встроенного аппаратного шифратора, который физически встраивается между источником и приемником защищаемых данных; тем не менее, это решение менее гибкое. Механизм надежного пути доступа может быть использован для гарантии того, что вредоносное ПО не сможет выдать себя за шифратор вызывающему приложению.
2. Злоупотребление аппаратным шифратором.
   Эта угроза может быть предотвращена путем использования функций мандатной безопасности, надежного пути доступа и защищенного канала связи

операционной системы. Механизм надежного пути доступа устраняет необходимость в отдельных физических интерфейсах для активизации шифратора. Защищенный канал связи позволяет аппаратному шифратору идентифицировать своих абонентов и обеспечивает более гибкий контроль за использованием его сервисов, алгоритмов, сессий и ключей. Вместо того, чтобы шифратор разбирался в контролировании своего использования, могут быть использованы механизмы мандатной безопасности (например, механизмы мандатной безопасности могут быть использованы для выделения аппаратного шифратора для использования только приложениями, запущенными тем пользователем, который активизировал устройство). Более того, механизмы мандатной безопасности могут снизить риск возможности использования аппаратного шифратора злоумышленным ПО, а следовательно они могут ограничить применение механизма надежного пути доступа только исключительно важными действиями.

Таким образом, даже в простейшем случае, свойства защищенных операционных систем помогают противостоять угрозам, возникающим при шифровании на прикладном уровне.

Отбросим предположения, сделанные выше, и рассмотрим возникающие при этом угрозы и способы борьбы с ними.

1. Ключи могут стать доступны на чтение или модификацию неавторизованному субъекту (пользователю или программе).

   Если начальные ключи передаются не по специально выделенному физическому каналу в аппаратный шифратор, то операционная система должна обеспечить защищенный канал связи между источником начальных ключей и аппаратным шифратором, а также обеспечить защиты самого источника начальных ключей. Механизмы мандатной безопасности могут быть использованы для тщательной защиты канала и источника ключей. Надежный путь доступа может потребоваться при введении начальных ключей.
2. Если криптографический механизм реализован на программном уровне, то его конфиденциальность и целостность становятся зависимыми от операционной системы, от ее возможностей по защите оперативной и внешней памяти.

   Для обеспечения конфиденциальности и целостности этого механизма необходимо применение мандатной защиты. Если криптографическим механизмом используются несколько внешних источников данных, например начальные значения для генератора псевдослучайных чисел, то источники входных данных и сами каналы передачи данных между источниками данных и криптографическим механизмом должны быть защищены с помощью мандатных механизмов безопасности.

6.2.3. Примеры

Мобильный код. Популярным способом «обезопасить» мобильный код является цифровая подпись апплетов и ограничение их исполнения только теми, которые получены из доверенных источников. В сущности, собственная безопасность ActiveX полностью основана на цифровых подписях, так как данная технология не имеет механизмов контроля доступа ни в какой из форм [14]. Основной уязвимостью этого подхода является позиция «все или ничего»; пользователь не может поместить собственные механизмы контроля доступа ActiveX в ограниченный домен безопасности. Для этой цели могут быть использованы механизмы мандатной безопасности операционной системы, с помощью которых можно ограничить действия броузера в рамках определенного домена безопасности.

Хотя цифровые подписи не являются самодостаточными, они играют важную роль в обеспечении безопасности мобильного кода, даже в защищенных операционных системах. Они могут уменьшить риск попадания злоумышленного кода в систему, предоставляя некоторый уровень гарантий того, что апплет будет функционировать корректно, а также предоставляя дополнительную информацию для принятия решения о предоставлении доступа. Тем не менее, так же как и в случае

с криптографической подсистемой прикладного уровня механизм проверки цифровых подписей зависит от свойств защищенной операционной системы, гарантирующей правильный вызов, обеспечивающей целостность механизма проверки и целостность хранящихся локально открытых ключей.

KERBEROS. *Kerberos* [18–20] — это сетевой сервис аутентификации, первоначально разработанный в MIT в рамках проекта Athena. В дополнение к предоставлению сервиса аутентификации, Kerberos поддерживает создание ключей сессии для обеспечения сетевых сервисов конфиденциальности и целостности.

Kerberos основан на использовании симметричного шифрования с доверенным центром распределения ключей для каждого домена (группы узлов сети). Доверенный центр распределения ключей Kerberos имеет доступ ко всем секретным ключам каждого субъекта своего домена. Соответственно его компрометация может привести к непоправимым последствиям. Обычно данную проблему прикрывают требованием обеспечения физической безопасности выделенного узла сети, выступающего в роли доверенного центра распределения ключей, на котором функционирует только сервис аутентификации Kerberos [21]. Без допущения о физической безопасности среды функционирования узлов, сервису аутентификации Kerberos и серверным приложениям, которые используют Kerberos, потребуются свойства защищенных операционных систем (для строгой гарантии их устойчивости к внешним воздействиям и невозможности обхода).

Kerberos был спроектирован для сред, в которых клиентские рабочие станции и сеть являются абсолютно ненадежными [22]. Тем не менее, так как ПО клиентской рабочей станции является посредником всех взаимодействий между пользователем и серверным приложением, пользующимся сервисом Kerberos, это предположение подразумевает, что серверное приложение, пользующееся сервисом Kerberos, должно рассматривать все клиентские приложения как потенциально вредоносное ПО. Более того, серверное приложение, пользующееся сервисом Kerberos, не имеет никаких возможностей для порождения надежного пути доступа для пользователя на клиентской рабочей станции, так как это потребовало бы наличия доверенного кода на клиентской рабочей станции. Поэтому в системах, использующих Kerberos, вредоносное ПО, запущенное пользователем, может совершенно свободно и произвольным образом модифицировать или передавать третьим лицам информацию пользователя без каких-либо ограничений (выделить различие между законным запросом пользователя и запросом злоумышленного ПО не представляется возможным). Учитывая возрастающую легкость, с которой вредоносное ПО может быть привнесено в систему, модель среды Kerberos кажется непригодной для использования. Как указано в [23], защищенные транзакции между конечными абонентами требуют наличия доверенного кода на обоих концах канала связи.

Предположим, что на клиентской рабочей станции существует некий базовый набор доверенного ПО, который защищен от внешних воздействий, но операционная система на клиентской рабочей станции не содержит механизмов мандатной безопасности и надежного пути доступа. Кроме того, предположим, что клиентская рабочая станция является однопользовательской системой, не предоставляющей никаких сервисов другим системам. Несмотря на эти допущения, пользователь все еще уязвим для атак со стороны злоумышленного ПО, такого как мобильный код, полученный пользователем из сети.

Если вредоносное ПО сможет выдать себя пользователю за клиентскую часть аутентификационной программы, тогда оно сможет получить пароль пользователя. Даже при применении одноразовых паролей эта атака даст возможность злоумышленному ПО действовать от имени пользователя в течение сеанса работы.

Для предотвращения осуществления подобной атаки необходимо применять механизм надежного пути доступа в операционной системе клиентской рабочей станции, а для предоставления надежного пути доступа между пользователями и серверными приложениями необходимо использовать механизм надежного пути доступа в комбинации с поддержкой сетевого защищенного канала связи.

Если вредоносное ПО имеет доступ на чтение к файлам, используемым клиентским ПО Kerberos для хранения билетов и ключей сессий, то вредоносное ПО может прямо выдавать себя за пользователя серверным приложениям, пользующимся сервисом Kerberos. Даже если ключи сессии помещены в аппаратный шифратор, вредоносное ПО может обратиться к аппаратному шифратору от имени пользователя, используя уязвимость неправомерного использования. Механизмы мандатной безопасности могут быть использованы для тщательной защиты и файла, и аппаратного шифратора от доступа со стороны злоумышленного ПО.

Сетевые протоколы обеспечения безопасности. Сетевые протоколы обеспечения безопасности IPSEC [24–26] используются для предоставления сервисов аутентификации, конфиденциальности и целостности на уровне протокола IP. Типичные реализации этого протокола основываются на серверах управления ключами прикладного уровня, которые используются для обмена и создания ключей для защищенных объединений. Модуль IPSEC в сетевом стеке взаимодействует с локальным сервером управления ключами посредством обращения из ядра к приложению для получения необходимой информации.

SSL [27] является другим сетевым протоколом обеспечения безопасности, который предоставляет сервисы аутентификации, целостности и конфиденциальности, а также сервис установления сессионных ключей и алгоритмов шифрования. Однако, SSL реализован полностью на прикладном уровне и не требует модификации ядра. SSL реализован в библиотеке, которая вклинивается в вызовы сокетов для включения протокола SSL между нижележащим протоколом транспортного уровня сокета (например, TCP) и протоколом прикладного уровня (например, HTTP).

Из-за того, что IPSEC использует криптографические сервисы прикладного уровня, используемый им сервер управления ключами подвержен уязвимостям, описанным в подразделе, посвященном криптографии, и нуждается в механизмах мандатной безопасности в операционной системе для реализации адекватной защиты. В свою очередь, так как защита, предоставляемая IPSEC, зависит от защиты ключей, то наличие мандатных механизмов защиты в операционной системе так же является критичным для удовлетворения требованиям безопасности IPSEC. А поскольку реализация SSL полностью работает на прикладном уровне, она целиком и полностью подвержена уязвимостям, описанным в подразделе, посвященном криптографии, и нуждается в механизмах мандатной безопасности в операционной системе для реализации адекватной защиты.

И IPSEC и SSL предназначаются для создания защищенных туннелей. Тем не менее, как отмечено в [23], защищенное взаимодействия между конечными абонентами требует наличия защищенных конечных точек доступа.

7. Операционные системы с встроенными мандатными механизмами безопасности

7.1. Коммерческие операционные системы

Существует несколько коммерческих операционных систем со встроенными мандатными механизмами безопасности (например, Windows NT, Solaris / Trusted Solaris, AIX), но ни одна из этих систем не нашла широкого применения. Эти системы имеют лишь ограниченное представление о мандатной безопасности, что ограничивает их рыночную привлекательность. К тому же, в этих системах обычно отсутствует адекватная поддержка необходимых доверенных приложений. Для достижения более широкого охвата рынка, операционная система должна поддерживать более общее представление о мандатной безопасности и должна предоставлять возможность гибкой настройки мандатных политик.

Принцип минимальных привилегий. Наиболее популярные операционные системы редко поддерживают принцип минимума привилегий, даже в своих архитектурах дискреционного контроля доступа. Многие операционные системы предоставляют только лишь разграничение между полностью привилегированным доменом безопасности и полностью непривилегированным доменом безопасности. Даже в Microsoft Windows NT механизм привилегий не обеспечивает адекватной защиты от злоумышленной программы, потому что он не ограничивает привилегии, которые программа наследует от вызывающего ее процесса, основываясь на степени надежности (доверенности) программы.

Надежный путь доступа. В большинстве используемых коммерческих операционных систем поддержка как механизма надежного пути доступа, так и защищенного канала связи, отсутствует. Microsoft Windows NT предоставляет поддержку надежного пути доступа для маленького подмножества функций, таких как аутентификация при регистрации в системе и смена пароля, но она не поддерживает расширение этого механизма для других доверенных программ. При локальном взаимодействии Microsoft Windows NT предоставляет серверам идентификационную информацию о клиентах, однако она не предоставляет клиентам идентификационную информацию о серверах.

7.2. Открытые ОС

7.2.1. Микроядерные операционные системы

Современные микроядерные исследовательские операционные системы имеют тенденцию к предоставлению примитивных механизмов защиты, которые могут быть использованы для гибкого построения архитектуры безопасности более высокого уровня. Многие из этих систем, такие как микроядро *Fluke* [28] и Exokernel [29], используют механизм привилегий, управляемых на уровне ядра ОС, как механизмы защиты нижнего уровня. Впрочем, как обсуждается в [5], типичная архитектура привилегий не является адекватной для обеспечения мандатного контроля доступа с высокой степенью гибкости и гарантий. *L4* [30] предоставляет некоторую поддержку мандатного контроля посредством своего механизма «кланов и начальников» и механизма межпроцессных коммуникаций (*IPC*) для идентификации отправителя и получателя сообщений, но все же не имеет согласованного набора инструментов для использования этих механизмов, чтобы они удовлетворяли требованиям мандатной политики. К тому же L4 предполагает, что будет использовано только небольшое количество отдельных доменов безопасности [30]. *Flask* [31], как вариант микроядра *Fluke*, предоставляет базовые механизмы мандатной безопасности, аналогичные *DTOS* [32], варианта микроядра *Mach*; обе системы предоставляют механизмы для обеспечения мандатного разграничения доступа и поддержки мандатной политики безопасности.

7.2.2. Реализации MAC для Linux

На данный момент существует несколько реализаций MAC (Mandatory Access Control) для Linux:

- SELinux
  При этом присутствуют:
  - реализация, основанная на микроядре;
  - ядро безопасности;
  - разные политики безопасности;
  - модель TE (Type Enforsment).
- RSBAC
  При этом присутствуют:
  - ядро безопасности;
  - подключаемые модули;
  - разные политики безопасности;

  - модель RC (на основе ролевой модели).
- Grsecurity

  При этом присутствуют:
  - монолитное построение системы;
  - одна модель безопасности, основанная на списках управления доступом (ACL);
  - дополнительные патчи к ядру Linux (PAX и др.).

Преимущества и недостатки SELinux. SELinux имеет следующие преимущества:

- наличие гибкой, четко сформулированной, простой модели безопасности — TE (Type Enforsment), базирующуюся на DTE и ролевой моделях;
- удобство администрирования;
- простота перехода на защищенную систему;
- переносимость (можно перенести на любую unix-подобную ОС);
- входит в стандартную поставку ядра Linux.

Что касается недостатков, то следует заметить, что основная часть критики SELinux есть критика конкретной реализации для ядра Linux, а именно подсистемы *LSM (Loadable Security Modules — загружаемые модули безопасности)*. Дело в том, что данная разработка была декларирована как универсальный фреймворк, хотя таковым (по крайней мере на данный момент) не является.

Преимущества и недостатки RSBAC. К положительным чертам следует отнести следующее:

- четкое разделение политики и механизма и, как следствие, модульность;
- разделение административных прав.

К сожалению основная модель RSBAC (RC) представляется не совсем удовлетворительной как с формальной, так и с практической точек зрения. Кроме того, представляется не совсем адекватным использование идентификатора пользователя в качестве идентификатора безопасности.

Преимущества и недостатки Grsecurity. Grsecurity на данный момент предполагает реализацию MAC на основе списков управления доступом (т. е. на основе IBAC) и, следовательно, ее модель безопасности нельзя считать достаточно выразительной и гибкой. Формальная модель безопасности фактически отсутствует.

Основным преимуществом Grsecurity является пакет дополнительных патчей к ядру ОС (например, PAX — защита от исполняемого стека) и их тесная интеграция с реализуемой политикой безопасности. Впрочем, большинство из этих патчей могут быть применены и отдельно.

8. Заключение

Возмем на себя смелость заявить, что ни одно из существующих технических решений в области безопасности не может обеспечить полную безопасность системы, поэтому необходимо достигнуть определенного баланса в использовании механизмов безопасности. Каждый механизм обеспечения безопасности предоставляет набор специфических функций и должен быть разработан только для выполнения этих функций. Он может опираться на другие механизмы и необходимые сервисы безопасности. В защищенной системе весь набор механизмов дополняет друг друга, таким образом вместе образуя полный комплект программ обеспечения безопасности. Системы, в которых данный баланс не достигнут, останутся уязвимыми.

Защищенная операционная система является важным и необходимым звеном на пути к разрешению вопроса полной безопасности системы, но она не является единственным звеном. Высоко-защищенная операционная система не будет эффективной без подсистемы безопасности прикладного уровня, построенной на ее

основе. Некоторые проблемы на самом деле лучше разрешаются на уровне выше операционной системы. Одним из таких примеров являются системы электронной торговли, которые требуют наличия цифровой подписи каждой из транзакций. Криптографический механизм прикладного уровня в системе транзакций, защищенной с использованием свойств защищенных операционных систем, возможно является наилучшим решением по обеспечению безопасности.

Ни один из механизмов обеспечения безопасности не способен обеспечить полную защиту в одиночку. Неразрешенные технические проблемы, ошибки реализации и неверные предположения о среде функционирования приведут в результате к остаточным уязвимостям. Например, скрытые каналы остаются серьезной технической проблемой для разработчиков защищенных операционных систем. Эти ограничения должны быть осмыслены, и предприняты соответствующие меры для внедрения дополнительных механизмов, разработанных для корректировки данной проблемы. В случае со скрытыми каналами необходимо применять аудит и механизмы обнаружения для минимизации шансов того, что кто-то воспользуется известными каналами утечки. Аудит и детекторы в свою очередь должны зависеть от защищенной операционной системы, обеспечивающей защиту таких важных компонентов, как журналы регистрации событий и датчики вторжения, так как они являются объектом тех же типов уязвимостей, которые обсуждались в данной статье.

Литература

1. Доверительная открытая патформа / П. Инглэнд, Б. Лэмпсон, Д. Манферделли и др // Открытые системы. — № 7-8. — 2003.
2. DOD 5200.28-STD // Department of Defense Trusted Computer System Evaluation Criteria. — 1985.
3. *Зегжда Д. П., Ивашко А. М.* Основы безопасности информационных систем. — М.: Горячая линия — Телеком, 2000. — 452 с.
4. *Ferraiolo D., Kuhn R.* Role-Based Access Control // Proceedings of the 15th National Computer Security Conference. — 1992.
5. DTOS General System Security and Assurability Assessment Report: Technical report md a904-93-c-4209 cdrl a011 / Secure Computing Corporation. — 1997. — `http://www.securecomputing.com/randt/HTML/dtos.html`.
6. *Badger L.* Practical Domain and Type Enforcement for UNIX // Proceedings of IEEE Symposium on Security and Privacy. — 1995.
7. DTOS Generalized Security Policy Specification: Technical report md a904-93-c-4209 cdrl a019 / Secure Computing Corporation. — 1997. — `http://www.securecomputing.com/randt/HTML/dtos.html`.
8. *Галатенко А.* О скрытых каналах и не только // Jet Info. — № 11(114). — 2002. — `http://www.jetinfo.ru/2002/11/2/article2.11.2002.html`.
9. *Тимонина Е. Е.* Скрытые каналы (обзор) // Jet Info. — № 11(114). — 2002. — `http://www.jetinfo.ru/2002/11/2002.11.pdf`.
10. *Грушо А. А., Тимонина Е. Е.* Теоретические основы защиты информации. — М.: Агентство «Яхтсмен», 1996. — 186 с.
11. *Грушо А. А.* Скрытые каналы и безопасность информации в компьютерных системах // Дискретная математика. — Т. 10, вып. 1. — 1998.
12. *Грушо А. А.* О существовании скрытых каналов // Дискретная математика. — Т. 11, вып. 1. — 1999.
13. Practical Domain and Type Enforcement for UNIX / L. Badger, D. F. Sterne, D. L. Sherman et al // 1995 IEEE Symposium on Security and Privacy. — 1995.
14. *Gong L.* Java Security: Present and Near Future // IEEE Micro. — 1997.
15. *Wallach D.* Extensible Security Architectures for Java: Technical report 546-97 / Dept. of Computer Science, Princeton University. — 1997.
16. *Ladue M.* When Java Was One: Threats from Hostile Byte Code // Proceedings of the 20th National Information Systems Security Conference. — 1997.

17. *Dean D.* Java Security: From HotJava to Netscape and Beyond // Proceedings of the IEEE Symposium on Security and Privacy. — 1996.
18. *Kohl J., Neuman C.* The Kerberos Network Authentication Service V5 // IETF RFC 1510. — 1993.
19. *Neuman C.* The Kerberos Network Authentication Service V5 R6 // IETF Draft July 1997. — 1998.
20. *Вьюкова Н.* Сервер аутентификации Kerberos // Открытые системы. — № 1. — 1996. — http://www.osp.ru/os/1996/01/44.htm.
21. *Neuman C., Ts'o T.* Kerberos: An Authentication Service for Computer Networks // IEEE Communications Magazine. — 1994.
22. *Bellovin S., Merritt M.* Limitations of the Kerberos Authentication System // Computer Communications Review. — No 20(5). — 1990.
23. *Brewer E.* Basic Flaws in Internet Security and Commerce. — 1995. — http://http.cs.berkeley.edu/simgauthier/endpoint-security.htm%l.
24. *Atkinson R.* IP Authentication Header (AH) // IETF RFC 1826. — 1995.
25. *Atkinson R.* IP Encapsulating Security Payload (ESP) // IETF RFC 1827. — 1995.
26. *Atkinson R.* Security Architecture for the Internet Protocol // IETF RFC 1825. — 1995.
27. *Wagner D., Schneier B.* Analysis of the SSL 3.0 Protocol // Proceedings of the 2nd USENIX Workshop on Electronic Commerce. — 1996.
28. *Ford B.* Microkernels Meet Recursive Virtual Machines // Proceedings of 2nd USENIX Symposium on Operating Systems Design and Implementation. — 1996.
29. *Mazieres D., Kaashoek M.* Secure Applications Need Flexible Operating Systems // Proceedings of the 6th Workshop on Hot Topics in Operating Systems. — 1997.
30. *Liedtke J.* L4 Reference Manual: Research report rc 20549 / IBM T. J. Watson Research Center. — 1996.
31. Assurance in the Fluke Microkernel: Formal Security Policy Model: Technical report md a904-97-c-3047 cdrl a003 / Secure Computing Corporation. — 1998.
32. *Minear S.* Providing Policy Control Over Object Operations in a Mach Based System // Proceedings of the 5th USENIX Security Symposium. — 1995.

UDC 681.322:621.382.26:004.056.52

The Necessity of Providing Operating Systems Security on a System Layer

D. S. Kulyabov, A. V. Korolkova

Telecommunication Systems Department
Peoples' Friendship University of Russia
Miklukho-Macklaya str., 6, Moscow, 117198, Russia

The purpose of this article is to attract attention to mandatory models of security. The insufficiency of providing system security on an application layer and the necessity to develop the ways, in which the mandatory security on a system layer is provided, are described in the work.