ВВЕДЕНИЕ

Операционная система Linux унаследовала систему безопасности Unix, разработанную еще в 70-х годах, передовую на момент создания, но на сегодняшний день уже явно недостаточную. Каждый пользователь имеет полную свободу действий в пределах своих полномочий по принципу «все или ничего». Это приводит к тому, что для выполнения некоторых задач пользователю часто предоставляется гораздо больше прав, чем это реально необходимо. Поэтому пользователь, получивший доступ с правами системной учетной записи, может добиться практически полного контроля над системой.

В процессе работы любого приложения могут возникать различные отклонения, приводящие в итоге к его аномальному выполнению. Это могут быть как системные сбои, ошибки в программировании, так и искусственно вызванные ситуации. Хакер, обнаружив, что при определенных условиях можно повлиять на выполнение программы, естественно, попытается этим воспользоваться. Предсказать поведение программы во внештатном режиме практически нереально. Примером тому являются антивирусы, которые все время работают в «догоняющем» ритме, не обеспечивая защиту от так называемых атак нулевого дня. Однако нормальное поведение программы можно описать с помощью относительно простых правил. В результате появилось несколько проектов, реализующих концепцию упреждающей защиты.

Целью данной курсовой работы является изучение программных продуктов, направленных на укрепление безопасности операционной системы, сравнительный анализ их основных характеристик, а также подведение итогов проделанной работы и обоснование их практического применения.

1. ПОНЯТИЕ ЗАЩИЩЕННОЙ ОПЕРАЦИОННОЙ СИСТЕМЫ. SELINUX

1.1 Понятие защищенной операционной системы

Операционная система — комплекс программ, обеспечивающий управление аппаратными средствами компьютера, организующий работу с файлами и выполнение прикладных программ, осуществляющий ввод и вывод данных.

Вычислить «самую безопасную ОС» не так просто, как это кажется на первый взгляд. Главный критерий, на который ориентируются пользователи, не разбирающиеся в стандартах безопасности, — это количество выявленных уязвимостей. Однако минимум обнаруженных в системе лазеек еще не повод считать ее надежно защищенной. Говоря о безопасности, нужно учитывать целый ряд факторов, в том числе:

- осуществляется ли проверка качества исходного кода ОС;
- какие заданы стандартные настройки безопасности;
- насколько быстро и качественно выпускаются исправления;
- как устроена система распределения полномочий и многое другое.

При выборе безопасной ОС определенно должна рассматриваться операционная система Linux.

Во-первых, ОС Windows никогда не разрабатывалась непосредственно для обеспечения безопасности системы, она всегда была закрыта от внешних глаз — весь код Windows зашифрован. В теории Windows можно подготовить для безопасного использования, но этим еще никто не занимался, так как это заняло бы огромное количество времени. Linux же, в силу своей открытости, позволяет работать с исходным кодом ОС. Уже выпущены специальные версии ОС Linux, которые являются абсолютно безопасными.

Во-вторых, технология Live CD — Linux «умеет» очень быстро запускаться и развертываться без установки на жесткий диск. Такую безопасную ОС можно записать на оптический диск или usb-накопитель и всегда иметь при себе. «По мгновению ока» возможно получить операционную систему с готовым рабочим столом и сопутствующими приложениями для работы в интернете, в независимости от установленной основной операционной системе в компьютере, которым предстоит пользоваться.

Ядро является центральным компонентом операционной системы. Оно отвечает за управление ресурсами системы, связь между аппаратным и программным обеспечением и безопасность. Ядро играет важнейшую роль в поддержке безопасности на более высоких уровнях.

Как уже было обозначено ранее, существует ряд важных патчей ядра Linux, направленных на обеспечение защиты системы. Их существенные отличия заключаются, главным образом, в том, как они администрируемы и как интегрируются в уже имеющуюся систему. Также патчи обеспечивают контроль доступа между процессами и объектами, процессами и другими процессами, объектами и другими объектами.

1.2 Security-Enhanced Linux

1.2.1 Общие сведения о SELinux

SELinux (англ. Security-Enhanced Linux — Linux с улучшенной безопасностью) — это реализация системы принудительного контроля доступа, которая может работать параллельно с классической системой контроля доступа, используемой в Linux.

SELinux существенно расширяет ядро Linux, делая операционные системы, построенные на его основе, пригодными к использованию в сферах, где доступ к информации должен быть строго разграничен, а приложения, запускаемые пользователями, должны иметь определенный набор прав, который не выходит за рамки минимально необходимого. Это делает систему привлекательной для государственных учреждений и военной отрасли. [4]

Система SELinux была разработана Агентством Национальной Безопасности США и всего за несколько лет стала стандартом в области систем контроля прав доступа.

Согласно официальным данным АНБ США:

«NSA Security-Enhanced Linux представляет собой набор патчей к ядру Linux и некоторых утилит, предназначенных для включения сильной, гибкой архитектуры мандатного управления доступом (MAC) в главные подсистемы ядра. Она предоставляет механизм принудительного разделения информации на основе требований конфиденциальности и целостности, который позволяет регистрировать угрозы вмешательства и обхода механизмов безопасности приложений, а также позволяет локализовать повреждения, которые могут быть вызваны вредоносными приложениями. SELinux включает в себя набор конфигурационных файлов политики безопасности, предназначенных для удовлетворения общих целей безопасности.» [1]

Хотя изначально АНБ США представило SELinux в виде набора исправлений, теперь SELinux является основополагающим элементом ядра Linux, начиная с версии 2.6. Как полностью работающая из коробки система SELinux впервые появился в дистрибутиве Red Hat Enterprise Linux 4.

Впоследствии поддержка SELinux была интегрирована в такие дистрибутивы, как Debian, OpenSUSE и Ubuntu, а дополнительные пакеты, активирующие систему, были добавлены во многие другие популярные дистрибутивы.

Для администраторов в SELinux заданы достаточно жесткие ограничения. В частности, нельзя зайти в систему удаленно — при необходимости такой процедуры необходимо сначала произвести вход обычным пользователем, а потом переключиться на административную роль (и перейти в административный домен) при помощи команды newrole, производящей дополнительную аутентификацию. Однако в большинстве современных UNIX-подобных систем администратору также запрещен удаленный вход. [13]

SELinux сложно применить для "небольшого усиления" стандартного дистрибутива Linux — настройка достаточно сложна и невозможна без изучения специального языка конфигурации. Но это не недостаток, а скорее следствие целей создания системы.

Система хоть и трудна в понимании, но невероятно логична и удобна в сопровождении, а имеющиеся средства управления позволяют очень точно диагностировать проблемы и легко их устранять.

1.2.2 Основные понятия SELinux

На данном этапе необходимо дать определения основным терминам, используемых в SELinux.

Субъект — это процессы, выполняемые от имени запустившего их пользователя.

Объект — элементы файловой системы (файлы, каталоги, ссылки и пр.) или другие процессы, над которыми выполняются действия.

Сущность — этот термин схож с понятием "пользователь" в классической схеме доступа. Сущность может иметь такое же название, как и логин пользователя. Если провести аналогию, то сущность - это конкретный человек.

Домен — это список того, что может делать отдельный процесс. Фактически домен - это действия, минимально необходимые одному процессу для выполнения его задачи.

По аналогии из реальной жизни, доменом можно назвать набор действий для совершения какой-либо операции. Например, при отправке факса это действия:

- поднять трубку;
- набрать номер;
- дождаться характерного свиста;

– нажать кнопку отправки факса;
– положить трубку.

Действие "посвистеть голосом в трубку в ответ на характерный свист" не входит в домен "отправка факса", поэтому это действие является недопустимым.

Роль — это список доменов, которые могут быть использованы. Если некоего домена нет в списке, то роль не может выполнить действия из этого домена.

В данном случае можно провести аналогию с должностью. То есть роль — это фактически должность (или должностная инструкция), которая может выполнять определённые наборы операций, или, в понятии SELinux, домены.

Тип — это набор действий (операций) применительно к объекту. Важно понять отличие от домена. Домен относится к процессам, а тип – к объектам, таким как файлы, каталоги и пр.

Возвращаясь к аналогии с факсом, можно увидеть всё тот же набор действий, но уже применительно к самому факсу:

– поднять трубку (дождаться гудка в линии);
– набрать номер (тоновый/импульсный);
– дождаться характерного свиста (Fax handshake);
– нажать кнопку отправки факса (Fax send);
– положить трубку (Hand Up).

Контекст безопасности, или *метка* — это набор всех атрибутов, связанных с объектами и субъектами. Контекст безопасности для субъектов (процессов) состоит из сущности, роли, домена, чувствительности и категории. Обычно используется только сущность-роль-домен(или тип), а целевая политика от Fedora использует только домены и типы.

Переход — это смена контекста безопасности. Есть два основных типа переходов:

– *Переход домена процесса* - процесс меняет контекст.
– *Переход типа файла* - создание файлов в определённых подкаталогах.

Например, пользователь создаёт html-страничку в каталоге WEB-сервера. Чтобы WEB-сервер получил доступ к этой страничке, необходимо сменить контекст безопасности файла (WEB-сервер не имеет доступа к контексту пользователя).

Политика — это набор правил, контролирующих взаимодействие ролей, доменов, типов и пр.

1.2.3 Linux и пользователи SELinux

В операционных системах Linux с запущенным SELinux, существуют пользователи Linux и пользователи SELinux.

Пользователь SELinux — это сущность, определённая в политике, которая отвечает за определённый набор. Каждый пользователь Linux сопоставлен пользователю SELinux посредством политики SELinux. Это позволяет пользователям Linux наследовать ограничения, установленные на пользователей SELinux. Сопоставленные сущности пользователей SELinux используются в контексте SELinux для процессов в сессии, в порядке определения для каких ролей и уровней они применимы. Для того, чтобы посмотреть сопоставление между пользователями SELinux и Linux используется команда semanage login –l. [2]

1.2.4 Методы управления доступом

В большинстве операционных систем имеются средства управления доступом, которые определяют, может ли определенный объект (пользователь или программа) получить доступ к определенному ресурсу. В системах UNIX применяется *разграничительный контроль доступа (discretionary access control, DAC)*. Этот метод позволяет ограничить доступ к объектам на основе групп, к которым они принадлежат. Например, в Linux для каждого файла определены владелец, группа, а также указаны права доступа к этому файлу. Правами доступа определяется, кто может получить доступ к файлу, кто может открыть его для чтения, кто может внести в него изменения, кто может запустить этот файл на выполнение. Права доступа определены для трех категорий: пользователь (владелец файла), группа (все пользователи, которые являются членами группы) и другие (все пользователи, которые не являются ни владельцем файла, ни членами группы).

Полагаться только на механизмы DAC - это фундаментально неверно для построения стойкой системы безопасности. Принятие решения о предоставлении доступа DAC основано только на подлинности пользователя и его правами доступа, игнорируя другую значимую информацию, такую как, роль пользователя, функцию и достоверность (надежность) программы, а также конфиденциальность и целостность данных. Такое разграничение прав доступа может привести к возникновению ряда проблем из-за того, что программа, в которой может быть обнаружена уязвимость, наследует все права доступа пользователя. Следовательно, она может выполнять действия с тем же уровнем привилегий, какой есть у пользователя, что нежелательно.

Вместо того чтобы определять ограничения подобным образом, более безопасно использовать *принцип наименьшего уровня привилегий (principle of least privilege)*, согласно которому программы могут делать только то, что им

необходимо для выполнения своих задач, и не более того. Таким образом, даже если в программе будет обнаружена уязвимость, то возможности доступа данной программы будет жестко ограничены. Такой тип контроля называется *принудительным управлением доступом (mandatory access control, MAC)*.

Другим методом управления доступом является управление *доступом на основе ролей (role-based access control, RBAC)*. При использовании RBAC права доступа предоставляются на основе ролей, выдаваемых системой безопасности. Отличие концепции ролей от традиционных групп состоит в том, что группа представляет одного или нескольких пользователей, в то время как роль, хотя она также может быть применена к нескольким пользователям, представляет совокупность полномочий на выполнение определенных действий. [5]

1.2.5 Архитектура SELinux

Реализация принудительного управления доступом требует четкого описания правил, что может привести к образованию большого их количества, поэтому в SELinux использована концепция управления доступом на основе ролей — RBAC. SELinux оперирует ролями, поэтому несколько учетных записей Linux могут иметь одну и ту же учетную запись SELinux.

Механизм защиты в SELinux носит название *Type Enforcement (TE)* и позволяет закрепить за каждым процессом и файлом, которые необходимо контролировать, некую метку. Если процесс, запущенный от имени администратора, скомпрометирован, то ущерб, который может быть причинен системе, ограничен только тем, к чему он может обращаться, согласуясь с установленными для него правилами.

Также в SELinux реализована *многоуровневая система обеспечения безопасности (Multi-Level Security model, MLS)*, но ее задействуют только в особых случаях, например, в правительственных многопользовательских системах, требующих чрезвычайно высокого уровня защиты. [11]

SELinux управляется загружаемыми политиками. Когда происходит событие, имеющее значение по безопасности, такое как, открытие процессом файла, то это действие перехватывается SELinux на уровне ядра. Если правила политики SELinux позволяют выполнение данной операции, то выполнение продолжается, иначе, операция блокируется и процесс получает сообщение об ошибке.

1.2.6 Способ действия

Что делает SELinux для того, чтобы контролировать обращения процессов к ресурсам ОС? Концептуально можно выделить четыре шага:

1. Все субъекты и объекты взаимодействия помечаются контекстом безопасности (процессы во время запуска, файлы — во время создания или установки ОС).

2. Когда субъект пытается произвести какое-либо действие в отношении объекта, информация об этом действии поступает к обработчику SELinux.

3. Обработчик смотрит на контексты безопасности субъекта и объекта и, сверяясь с написанными ранее правилами — политикой, принимает решение о дозволенности действия.

4. Если действие оказывается правомочным (политика его разрешает), объект (программа) продолжает работать в обычном режиме, в противном случае — она либо принудительно завершается, либо получает отказ. [4]

1.2.7 Политики в SELinux

В SELinux права доступа определяются наборами правил, или политиками. Политики работают на уровне системных вызовов и обрабатываются ядром, но можно реализовать и на уровне приложения. Политики описываются при помощи специального языка описания правил доступа.

В терминологии данной системы политика безопасности определяет набор доменов и типов (Type Enforcement domain and types). Каждый субъект в каждый момент времени ассоциирован с определенный доменом, а каждый объект — с определенным типом.

В настоящий момент уже разработано несколько готовых политик безопасности, которые можно использовать по умолчанию на серверах и даже домашних компьютерах. Всё, что требуется от системного администратора — выбрать используемую политику и перезагрузить компьютер с включённым SELinux.

В среднем, политика безопасности SELinux для всей системы содержит более ста тысяч правил, так что её создание и отладка занимает значительное время. Типичный объем файла политик SELinux для основных приложений и сервисов дистрибутива может содержать до нескольких сотен тысяч строк, поэтому были созданы инструменты для их автоматической генерации.

Наиболее распространены следующие три политики:

– *Целевая (targered)*. Эта политика разработана компанией Red Hat и является наиболее используемой.

– *Многоуровневая (MLS)*. Позволяет обеспечивать уровни безопасности и может использоваться госструктурами для хранения информации различных уровней секретности.

– *Строгая (strict)*. Этот вариант политики подразумевает правило "Что не разрешено, то запрещено" — принцип наименьшего уровня привилегий (principle of least privilege). [2]

Любой Linux-дистрибутив, из коробки оснащенный SELinux, по умолчанию использует целевую политику, которая предполагает наличие всего нескольких общих SELinux-пользователей и ролей. При использовании целевой политики значимым полем контекста безопасности остается только «тип субъекта/объекта», тогда как поля «пользователь» и «роль» просто определяют отношение SELinux к процессу — либо он находится под контролем правил, либо нет.

В политике безопасности также определен набор ролей. Для пользователей первичная установка роли происходит в процедуре регистрации (login), а переключение роли — при помощи команды newrole. Системные процессы работают с ролью system_r, обычные пользователи - с ролью usr_r, а для системных администраторов зарезервирована роль sysadm_r. [4]

1.2.8 Режимы SELinux

Предусмотрена работа SELinux в трех режимах

– *enforcing* — принудительный: политика SELinux включена принудительно. SELinux запрещает доступ, основываясь на правилах политики SELinux, все, что не соответствует политике — блокируется.

– *permissive* — разрешающий: политика SELinux не принудительна. SELinux не запрещает доступ, политики анализируются, все нарушения заносятся в журнал, но блокировки не производятся.

– *disabled* — SELinux отключен. Используются только дискретные правила DAC.

Команда /usr/sbin/setenforce используется для перехода между принудительным (enforcing) и разрешающим (permissive) режимами. Команда /usr/sbin/getenforce используется для просмотра текущего режима SELinux.

Собственно настройки производятся в конфигурационных файлах, размещенных в каталоге /etc. Файл /etc/selinux/config - это основной главный конфигурационный файл SELinux. Он определяет режим SELinux и используемую политику SELinux. [5]

1.2.9 Преимущества использования SELinux

– *Все процессы и файлы маркируются определённым типом.*

Тип определяет домен для процесса и тип для файлов. Процессы отделяются один от другого запуском в различных доменах, и правила политики SELinux определяют, как процесс взаимодействует с файлами, как процессы взаимодействуют друг с другом. Доступ предоставляется только в том случае, если существует правило политики SELinux, разрешающее конкретное действие.

– *Тонко-настраиваемый контроль доступа.*

Решения доступа SELinux принимаются на основе всей доступной информации, такой как пользователи SELinux, роли, тип и уровень (опционально).

– *Политика SELinux назначается административно, вводится в действие на уровне всей системы и не основывается на разделении пользователей.*

– *Уменьшение уязвимостей, связанных с атаками по повышению привилегий.*

Так как процессы запускаются в доменах и, таким образом, отделены друг от друга, и так как правила политики SELinux определяют, как процессы получают доступ к файлам и другим процессами, то если процесс

скомпрометирован, атакующий имеет доступ только к штатным функциям данного процесса и к файлам, доступ к которым есть у данного процесса.

– *SELinux может быть использован для усиления конфиденциальности и целостности данных, а также для защиты процессов от атак.*

1.2.10 Примеры

В следующих примерах демонстрируется, как SELinux повышает безопасность:

– Действие для SELinux по умолчанию — это блокирование доступа. Если не существует правила политики SELinux, которое предоставляет доступ, к примеру, для открытия файла процессом, то доступ блокируется.

– SELinux может ограничивать пользователей Linux. Определенное число ограниченных пользователей SELinux существует в политике SELinux. Пользователи Linux могут быть сопоставлены с ограниченными пользователями SELinux для получения преимуществ использования ролей безопасности и применяемых к ним механизмов.

– Использование разделения процессов. Процессы запускаются каждый в своём домене, предотвращая, доступ к файлам, используемым другими процессами, а также блокируя процессам доступ к другим процессам.

Выводы по главе

SELinux не является:

– антивирусным программным обеспечением.

– заменой паролям, межсетевым экранам или другим системам безопасности.

– решением безопасности "всё в одном".

SELinux разработан для усовершенствования существующих решений по безопасности. Даже с запущенным SELinux, необходимо использование практик по безопасности, таких как обновление программного обеспечения последними обновлениями, использование сложных паролей, межсетевых экранов и прочего.

ОС SELinux является одной из наиболее всесторонне защищенных сред, доступных на сегодняшний день, но, безусловно, не единственной

2. СРАВНИТЕЛЬНЫЙ АНАЛИЗ ЗАЩИЩЕННЫХ ОПЕРАЦИОННЫХ СИСТЕМ LINUX

2.1 AppArmor

AppArmor — программный инструмент упреждающей защиты, основанный на политиках безопасности(известных также как профили (англ. profiles)), которые определяют, к каким системным ресурсам и с какими привилегиями может получить доступ то или иное приложение.

Изначально AppArmor был разработан компанией Immunix, которая выпускала одноименный дистрибутив Linux и специализировалась в области защиты ОС и ее приложений от различных атак. После ее приобретения компанией Novell в мае 2005 года инструмент был открыт под лицензией GNU GPL и включен в состав дистрибутива OpenSUSE под именем AppArmor. Постепенно он был адаптирован для некоторых других дистрибутивов: PLD, Pardus Linux, Annvix, RHEL 5, Slackware 10.2 и Ubuntu. [11]

Согласно официальным данным компании Novell:

«AppArmor - самая эффективная и простая в использовании система безопасности приложений Linux, доступная сегодня на рынке. AppArmor - платформа безопасности, реализующая концепцию упреждающей защиты операционной системы и приложений от внешних или внутренних угроз, даже атак нулевого дня. Профили безопасности AppArmor полностью определяют, к каким ресурсам системы программы могут получить доступ и с какими привилегиями. Ряд политик по умолчанию включены в AppArmor, тем самым политики AppArmor даже для очень сложных приложений могут быть развернуты успешно в считанные часы.» [6]

Основная идея AppArmor состоит в том, что система защиты не должна быть сложной и не должна мешать. В отличие от SELinux, AppArmor не использует расширенные атрибуты и не зависит от файловой системы.

Для определения того, к каким системным ресурсам и с какими привилегиями может получить доступ то или иное приложение, используются политики безопасности - профили, которые привязаны к пути файла или каталога, причем самого файла может и не быть на момент активации профиля.

Профиль состоит из файлов, каталогов, с указанием полных путей к ним и прав доступа к этим объектам. С помощью профилей к стандартной Unix-модели безопасности DAC (Discretionary Access Control)добавляется более мощная - MAC (Mandatory Access Control).

В отличие от SELinux, в которой настройки глобальны для всей

системы, профили в AppArmor разрабатываются индивидуально под каждое приложение (в SELinux также начинает использоваться такой подход). Однако существует недостаток - при переносе файла в SELinux за ним полностью сохраняется контекст безопасности, в AppArmor — нет. Также, если средствами SELinux можно предусмотреть несколько уровней доступа к объекту для разных субъектов, то AppArmor этого не умеет.

Для упрощения настроек в AppArmor уже включен набор стандартных профилей, запускаемых после установки. Отдельно доступны профили для многих популярных программ и серверов. Чтобы подключить готовый профиль к AppArmor, достаточно его скопировать в каталог /etc/apparmor.d.

Если же готового профиля найти не удалось, в его создании помогут специальные инструменты (genprof и logprof). Для пользователей OpenSUSE имеется графический интерфейс к этим утилитам, реализованный в интерфейсе управления пакетами Yast2. Дополнительные профили можно найти в репозитории дистрибутива (apt-cache search apparmor), кроме того, существует онлайн-банк профилей —apparmor.opensuse.org. [11]

AppArmor может обрабатывать профили в двух режимах:

– *enforce* – принудительный режим, сервис работает исключительно в пределах профиля, все попытки нарушить правила регистрируются в syslog;

– *complain* - щадящий режим (обучения), в этом случае работа сервиса просто контролируется, при нарушении профиля создается запись. Этот режим удобен при создании новых профилей и настройки профиля на конкретной системе.

По умолчанию профили AppArmor работают в принудительном enforce-режиме. Когда сервис не может выйти за рамки установок, все попытки блокируются и фиксируются в журнале. При необходимости его можно перевести в щадящий режим complain, когда нарушения лишь фиксируются. Причем, в отличие от SELinux, где режим обучения активируется глобально, в AppArmor его можно включить для отдельного профиля.

Для изменения режима достаточно открыть файл профиля и напротив исполняемого файла добавить строку «flags=(complain)»:

/bin/ls flags=(complain)

Глобально все профили перевести в режим complain можно командой:

$ sudo echo 1 > /sys/kernel/security/apparmor/control/complain

После обучения отдельное приложение без перезагрузки всех профилей можно перевести в жесткий режим с помощью специальной утилиты enforce:

$ enforce dovecot Setting /usr/sbin/dovecot to enforce mode.

Остальные профили лежат в каталоге /usr/share/doc/apparmor-profiles/extras. Каждый профиль имеет имя, которое

состоит из полного пути к исполняемому файлу, только вместо слеша используется точка. Например, usr.lib.firefox.firefox.sh. [7]

Таким образом, вся философия работы с AppArmor сводится к правильному выбору приложений, нуждающихся в ограничении привилегий, и созданию/редактированию профилей безопасности.

К слову, для версий ядер 2.4/2.6 существовала разработка *Trustees*, которая в удобной форме расписывала доступ к каталогам вплоть до указания отдельных групп и пользователей и не зависела от файловой системы. К сожалению, проект прекратил свое существование, хотя мог бы стать золотой серединой между SELinux и AppArmor. [11]

2.2 GRSecurity

Grsecurity представляет собой набор патчей для ядра Linux, который включает в себя некоторые улучшения связанные с безопасностью, включая принудительный контроль доступа, рандомизацию ключевых локальных и сетевых информативных данных, контроль сетевых сокетов, контроль возможностей, и добавочные функции аудита.

Также необходимо отметить такие улучшения, как предотвращение произвольного выполнения кода в ядре, снижение риска утечки важной информации при ошибках ядра, ограничение, обеспечивающее пользователю просматривать только его процессы, служба оповещения и аудит системы безопасности, содержащие IP адрес узла, вызывавшего тревогу.

Типичной областью применения являются web-сервера и системы, которые принимают удалённые соединения из сомнительных мест.

Патч grsecurity выпущен под лицензией GNU GPL, является свободным ПО и включает в себя набор патчей PaX, который, среди прочих других возможностей, отмечает области данных в памяти. Целью этой защиты является защита памяти от записи, что предотвращает множество уязвимостей в безопасности, таких как переполнение буфера. [9]

В grsecurity существует три уровня безопасности на выбор: низкий, средний и высокий.

Низкий уровень рекомендован, когда более высокие уровни не подходят из-за использования нестандартного набора ПО. Он содержит:

– защиту ссылок и FIFO – пользователям запрещается переходить по ссылкам (писать в FIFO), владельцем которых является другой пользователь;

– запрет пользователям на чтение dmesg – у всех пользователей, кроме администратора, отбирается право на чтение системных сообщений ядра;

– начальную защиту chroot – рабочая директория для всех только что запущенных в песочнице приложений будет принудительно установлена в корневую директорию chroot.

В *среднем уровне* защиты дополнительно добавятся технологии:

– дополнительные ограничения для приложений, запускаемых в chroot;

– ограничение прав на чтение /proc для пользователей, не входящих в заранее заданную группу;

– рандомизация адресного пространства;

– серьезное логирование подозрительных событий.

Высокий уровень:

– ограничивает права на чтение /proc – теперь пользователи смогут читать из /proc информацию только о своих процессах;

– добавляет дополнительное логирование;

– добавляет ограничение на чтение информации о ядре через системные вызовы для обычных пользователей (для администратора эта возможность остается).

Предусмотрен также вариант Custom, позволяющий отдельно выбирать все необходимые опции для уровня безопасности системы. [10]

Помимо уровней безопасности, при конфигурации можно включить/отключить RBAC (Role Based Access Control) – управление доступом на основе ролей. Управление пользователями и их ролями осуществляется с помощью специальной утилиты gradm2, присутствующей в большинстве дистрибутивов:

$ sudo apt-get install gradm2

2.3 Сравнительный анализ конкретных характеристик систем

Таблица 1. Сравнительный анализ основных характеристик расширений безопасности операционных систем Linux – SELinux, AppArmor, grsecurity

	SELinux	AppArmor	grsecurity
Уровень квалификации администратора системы	Высокий	Средний	Низкий
Мощный механизм управления доступом	Да	Да	Нет (grsecurity проще администрировать, нежели рассмотренные расширения безопасности AppArmor и SELinux. Кроме того, процесс создания политик проще, так как отсутствуют понятия ролей или сложных переходов между доменом/файлом)
Требование подробной настройки системы	Да	Да	Нет (работает в режиме обучения)
Инструменты GUI для записи / редактирования набора правил	Да	Да	Нет
Инструменты CLI для записи / редактирования набора правил	Да	Да	Да
Простота использования	Нет	Да (неформально AppArmor часто описывается, как менее сложный для среднего пользователя, нежели SELinux)	Да
Бинарный пакет	Доступно для большинства дистрибутивов Linux	Доступно для дистрибутивов Ubuntu / Suse / OpenSUSE	Доступно для дистрибутивов Ubuntu / RHEL / CentOS / Debian
Влияние на производительность	Нет	Нет	Нет

системы			
Метод разделения доступа	MAC, RBAC	MAC	MAC, RBAC
Поддержка аудита и ведение журнала	Да	Да	Да
Круг пользователей	Корпоративные пользователи	Корпоративные пользователи	Веб-сервер и хостинг
Документация	Подробно задокументировано	Документация в основном из OpenSUSE и Suse Enterprise Linux	Слабо задокументировано

Вывод:

Каждый из рассмотренных программных продуктов обеспечивает безопасность системы на должном уровне. При выборе ПО можно руководствоваться следующим принципом:

- Новый пользователь / простота использования: Grsecurity.
- Простые для понимания политика безопасности и инструменты: AppArmor.
- Наиболее мощный механизм контроля доступа: SELinux.

2.4 Другие примеры расширений ОС

2.4.1 RSBAC

Конкуренцию SELinux может составить проект *RSBAC*, реализующий мандатный и ролевой механизмы доступа. Начавшись намного раньше SELinux, проект RSBAC уже в 2000 году достиг стабильного состояния.

Разработчики гордятся тем, что совершенно не зависят от правительственных организаций и больших компаний, – их код написан «с нуля».

RSBAC распространяется под лицензией GPL и представляет собой набор патчей к текущему ядру Linux. В отличие от SELinux, в основную ветку ядра Linux RSBAC не входит.

Функциональность RSBAC достаточно велика, с его помощью можно добиться таких интересных эффектов, как организация доступа к файлу только в определенные часы.

Помимо привычного администратора в операционную систему добавляется администратор безопасности, который может ограничить всех пользователей (в том числе и суперпользователя) в доступе к информации. Это создает лишний уровень привилегий на пути злоумышленника к полному контролю над системой, но возлагает большую ответственность на

администратора безопасности.

Вся дополнительная информация, используемая RSBAC, хранится в дополнительном каталоге, который доступен только ядру системы.

Функционально RSBAC состоит из нескольких модулей, а центральный компонент принимает комплексное решение, основываясь на результатах, возвращаемых каждым из активных в данный момент модулей (какие модули задействовать и в каком объеме определяется на этапе настройки системы).

RSBAC можно интегрировать с любым дистрибутивом Linux, необходимо лишь внести соответствующие исправления в исходные тексты ядра системы. Эта процедура осуществляется при помощи стандартной утилиты patch и прилагаемого "файла-заплатки", который предлагается для разных версий ядра.

Однако явно ощущаются недостаток документации, небольшое количество разработчиков и пользователей системы. [13]

2.4.2 TOMOYO Linux

Проект *TOMOYO Linux* начат в 2003 году японской компанией NTT DATA CORPORATION как легкая реализация MAC для Linux-ядра. Через два года лицензию изменили на GNU GPL и выложили код на SF.net.

Некоторое время проект предоставлял патчи и готовые сборки ядер для разных дистрибутивов. Но начиная с версии ядра 2.6.30, код TOMOYO Linux включен в основную ветку разработки.

При беглом взгляде TOMOYO очень похож на AppArmor. Обе системы контролируют путь (pathname based), а правила имеют сходный синтаксис. Но есть и отличия.

В отличии от AppArmor, TOMOYO Linux предназначен для защиты всей системы от атак, использующих уязвимости в прикладных программах. TOMOYO Linux сопоставляет действия приложения, полученные в ходе тестового запуска с теми, что происходят в реальных условиях, тем самым записывая или\и блокируя подозрительную активность.

Также в TOMOYO можно указать поведение программы в зависимости от того, как она запущена. Например, оболочка, запущенная через SSH, может иметь больше ограничений, чем запущенная с локальной системы.

Приложения в терминологии TOMOYO называются доменами (domains). Конфигурационные файлы TOMOYO находятся в каталоге /etc/tomoyo, после запуска системы настройки имеют свое отражение в /proc/tomoyo, где их можно редактировать. Параметры работы TOMOYO хранятся в /etc/tomoyo/profile.conf и доступны в /proc/tomoyo/profile. Именно здесь определяются режимы работы TOMOYO — disable, permissive, enforsing и learning (обучаясь, система сама строит правила).

TOMOYO Linux требует создания правил безопасности с нуля в режиме обучения, в котором система будет автоматически создавать правила, содержащие необходимые и достаточные настройки доступа для отдельной системы.

TOMOYO Linux отчитывается обо всём что происходит с системой и может быть использован для анализа поведения системы. Еще одна немаловажная черта — TOMOYO может работать параллельно с SELinux и AppArmor.

TOMOYO Linux не предназначен для профессионалов в области компьютерной безопасности, однако вполне может использоваться в качестве инструмента для продвинутых пользователей и системных администраторов. [11]

2.4.3 Виртуализация операционных систем

Еще одним вариантом усиления безопасности операционных систем

является технология виртуализации операционных систем, также известная как виртуальные частные серверы. В этом подходе одна операционная система содержит множество изолированных (в плане функциональных возможностей) объектов пространства пользователя. Виртуализация операционных систем также ограничивает возможности приложений, работающих в рамках изолированного пространства пользователя. Например, объект пространства пользователя не имеет возможности изменять ядро (загружать или удалять модули ядра), монтировать и демонтировать файловые системы. Не разрешено все, что может изменить среду других объектов пользователя.

Многие операционные системы поддерживают виртуализацию. Linux поддерживает VServer, Parallels Viruozzo Containers и OpenVZ. В других операционных системах имеются контейнеры (Solaris) и «тюрьмы» (BSD). [13]

Выводы по главе

Рассмотренные системы не единственны в своем классе - большинство защищенных решений делается на базе открытых систем. Среди них — проекты *Medusa* [14], тесно сотрудничающий с RSBAC, и *LinuxIDS* [15]. Из отечественных в первую очередь следует вспомнить "легендарную" МСВС, сделанную на базе Linux для Министерства обороны в 1997 году. К сожалению, информации о данной ОС мало. Критерии классификации систем отличаются от американской "Оранжевой Книги" и изложены в Руководящем Документе "Автоматизированные системы. Защита от несанкционированного доступа к информации. Классификация автоматизированных систем и требования по защите информации". [16]

Если выйти за рамки Linux и обратиться к другим UNIX-подобным системам с открытым исходным кодом, то следует упомянуть криптографический маршрутизатор *"Шип"*, построенный на базе ОС FreeBSD. Несколько особняком стоит проект, разрабатываемый Специализированным Центром Защиты Информации Санкт-Петербургского Государственного технического Университета — операционная система *"Феникс"* [17]. Она не происходит явно от какой-либо разработки Фонда свободного программного обеспечения (Free Software Foundation - FSF), но унаследовала, по словам разработчиков, лучшие особенности микроядер других ОС.

ЗАКЛЮЧЕНИЕ

При построении системы одной из главных задач является обеспечение ее безопасности. Злоумышленник может воспользоваться имеющимися в системе уязвимостями, чтобы получить неавторизованный доступ к важной информации.

Подводя итоги, можно заключить, что в современном мире существует большое число программных продуктов, направленных на обеспечение и укрепление безопасности системы и информации в частности.

Каждый из таких продуктов обеспечивает определенный уровень безопасности системы, степень ее защищенности от внешних и внутренних атак, вследствие чего требования к уровню квалификации, накопленному опыту и знаниям специалиста, работающего с таким программным продуктом, также различаются.

Поэтому на этапе обеспечения безопасности системы крайне важно принять решение – можно ли позволить «сэкономить» на защите системы, а соответственно на обслуживающем ее специалисте, или все-таки этого не стоит делать. Для принятия такого важного решения необходимо, прежде всего, достоверно определить ценность защищаемой информации, ее актуальность и значимость.

Цели курсовой работы достигнуты. Очевидно, что проведенный анализ имеет непосредственное практическое применение при решении проблемы, описанной выше.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Security-Enhanced Linux [Электронный ресурс]: Официальный сайт SELinux. URL: http://www.nsa.gov/research/selinux/index.shtml (дата обращения 10.03.2014)
2. Настройка и использование SELinux [Электронный ресурс]: URL: http://www.opennet.ru/base/sec/selinux_setup.txt.html (дата обращения 10.03.2014)
3. SELinux [Электронный ресурс]: URL: https://wiki.archlinux.org/index.php/SELinux_(Русский) (дата обращения 13.03.2014)
4. SELinux: бронежилет для корпоративного пингвина [Электронный ресурс]: URL: http://www.xakep.ru/post/56714/default.asp (дата обращения 13.03.2014)
5. Анатомия SELinux [Электронный ресурс]: URL: http://www.ibm.com/developerworks/ru/library/linux/l-selinux/ (дата обращения 13.03.2014)
6. Using AppArmor to Create Confined Root Shells [Электронный ресурс]: Официальный сайт AppArmor. URL: https://www.novell.com/support/kb/doc.php?id=3768203 (дата обращения 10.03.2014)
7. Бронированный тукс: AppArmor для определения политик безопасности ПО [Электронный ресурс]: URL: http://www.xakep.ru/post/39931/default.asp (дата обращения 14.03.2014)
8. SDB:AppArmor [Электронный ресурс]: URL: http://ru.opensuse.org/SDB:AppArmor (дата обращения 10.03.2014)
9. grsecurity [Электронный ресурс]: Официальный сайт grsecurity. URL: http://grsecurity.net/ (дата обращения 10.03.2014)

10. Термоядерный синтез: Обзор патчей для Linux, не входящих в ванильное ядро [Электронный ресурс]: URL: http://www.xakep.ru/post/54792/ (дата обращения 13.03.2014)
11. Гонка вооружений: сравниваем популярные расширения безопасности для ОС Linux [Электронный ресурс]: URL: http://www.xakep.ru/post/53424/default.asp (дата обращения 13.03.2014)
12. Linux Kernel Security [Электронный ресурс]: URL: http://www.cyberciti.biz/tips/selinux-vs-apparmor-vs-grsecurity.html (дата обращения 10.03.2014)
13. Linux на защите информации [Электронный ресурс]: URL: http://cybervlad.net/selinux/ (дата обращения 13.03.2014)
14. Проект Medusa. [Электронный ресурс]: URL: http://medusa.fornax.sk (дата обращения 19.03.2014)
15. Проект LinuxIDS. [Электронный ресурс]: URL: http://www.lids.org/ (дата обращения 19.03.2014)
16. Документы Гостехкомиссии. [Электронный ресурс]: URL: http://www.infotecs.ru/gtc/default.htm (дата обращения 19.03.2014)
17. ОС "Феникс". [Электронный ресурс]: URL: http://www.ssl.stu.neva.ru/ssl/fenix.shtml (дата обращения 19.03.2014)