УДК 681.322

ЗАЩИЩЕННАЯ ОПЕРАЦИОННАЯ СИСТЕМА BASTION НА БАЗЕ ЯДРА LINUX

Р.Х. САДЫХОВ, Д.С. КОЧУРОВ

Белорусский государственный университет информатики и радиоэлектроники
П. Бровки, 6, Минск, 220013, Беларусь

Поступила в редакцию 9 января 2003

Предлагаются принципы построения защищенной операционной системы Bastion, «свободной» от недостатков в обеспечении безопасности, присущих традиционным Unix-подобным операционным системам. В качестве основы используется система контроля доступа LPS, ориентированная на ядро Linux и реализующая различные модели безопасности в комплексе, обеспечивая их совместную работу.

Ключевые слова: Защищенная операционная система, система контроля доступа, модель безопасности.

Введение

Как известно, Linux — это клон Unix семейства операционных систем, объединенных общей идеологией. Первые Unix-системы создавались исключительно как средство совместного ведения проектов научными коллективами университетов и ее разработчики не придавали особого значения вопросам безопасности и защиты информации. Это привело к тому, что ни одну из Unix-подобных систем нельзя сегодня назвать по-настоящему безопасной. На протяжении всей истории существования ОС ее постоянно взламывают. Создается иллюзия, что это самая незащищенная из всех операционных систем.

Относительно защищенный вариант Linux-системы можно построить на базе ее штатных средств, и для большинства применений это будет вполне адекватный уровень безопасности. Этот процесс достаточно подробно описан во многих документах: начиная от классических рекомендаций UNIX Configuration Guidelines (www.cert.org) и заканчивая циклом публикаций Ланса Спитцнера, посвященных "усилению" различных ОС: Windows NT, Solaris, Linux [1]. Однако в самой идеологии построения Linux имеется ряд фундаментальных изъянов, которые невозможно преодолеть только грамотным администрированием [4].

1. Недостатки организации защиты в Linux

Фактически в Linux предполагается всего два варианта статуса пользователя: обычный и суперпользователь (root), что сразу порождает две проблемы.

Первая проблема заключается в том, что root никому не подконтролен: он может изменить любую настройку, имеет доступ абсолютно ко всем объектам в файловой системе, может стереть или модифицировать любые данные и записи в журналах регистрации. Такая «концентрация власти», конечно же, неприемлема для защищенной системы.

Вторая проблема связана с необходимостью изменения текущего уровня привилегий пользователя для выполнения некоторых действий (например, для смены пароля). Традиционно это достигалось путем установки флага SUID (или SGID) на файлы исполняемых модулей про-

грамм, в результате чего порождаемый при запуске такого файла процесс приобретает не права запустившего его пользователя, а права владельца файла. Нет необходимости говорить, к чему может привести даже небольшая ошибка в программе с установленным флагом SUID и владельцем root.

Реализованная в Linux модель разграничения доступа дискретна — права доступа субъектов (пользователей, процессов) к объектам (файлам, каталогам и т. п.) задаются явно, в матричной форме. На практике же часто возникает необходимость в реализации принудительной модели, при которой права доступа субъекта к объекту определяются уровнем субъекта и классом объекта, либо комбинированной модели, включающей в себя и принудительную и дискреционную модель. Причем в Linux реализация некоторых требований по разграничению может оказаться попросту невозможной. Это связано с тем, что права доступа в Linux задаются всего лишь для трех категорий субъектов: владельца файла, ассоциированной с файлом группы и всех остальных. В частности, невозможно задать разные права доступа к файлу для начальника отдела, менеджера проекта и сотрудника, непосредственно работающего с файлом, закрыв доступ всем остальным пользователям.

Для решения приведенных проблем с организацией защиты и построения защищенной ОС Bastion применена система LPS (Linux Protection System). В качестве основы при разработке системы LPS была взята система контроля доступа RSBAC (Rule Set Based Access Control — контроль доступа на базе набора правил). RSBAC — это надстройка над ядром Linux и комплект утилит управления, позволяющие построить защищенную операционную систему на базе ядра Linux.

2. Структура системы RSBAC

Упрощенно архитектуру системы LPS можно представить в виде трех функциональных блоков (рис. 1) — AEF (Access Enforcement Facility — модуль принужденного доступа), ADF (Access Decision Facility — модуль принятия решений) и ACI (Access Control Information — информация контроля доступа).

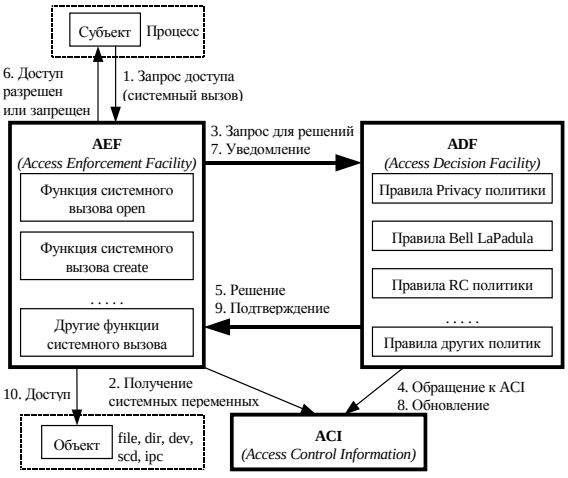

Рис. 1. Архитектура системы LPS

AEF обеспечивает абстрагирование системы контроля доступа от ядра. В современном ядре Linux очень много системных вызовов (свыше 200). AEF транслирует их в 36 видов запросов на доступ.

Модуль принятия решений ADF производит опрос модулей, реализующих свои модели доступа, и возвращает ответ в стандартной форме (GRANTED, NOT GRANTED, DO NOT CARE) обратно AEF. Последний обеспечивает возврат ответа в формате исходного системного вызова.

Все перечисленные компоненты активно используют информацию от ACI – общего хранилища атрибутов объектов и субъектов системы. База ACI в LPS хранится между перезагрузками на диске в специальном каталоге, доступ в который имеет исключительно ядро, а в остальное время – в оперативной памяти.

LPS имеет модульную структуру, причем каждый модуль реализует свою собственную модель защиты. Окончательное решение о предоставлении доступа или отказе в нем получается как суммарное после обсуждения этого вопроса всеми модулями. Модули принятия решений можно активировать и деактивировать прямо во время работы системы.

В систему LPS включены следующие модули:

AUTH (Authentication) — головной модуль, следящий за сменой владельца у процессов. В обычной среде процессы, запускаемые от имени администратора, могут менять владельца на любого пользователя без указания пароля, что неблагоприятно сказывается на конфиденциально-

сти информации. Теперь же процессы могут менять владельца только на конкретных разрешенных пользователей или некоторый диапазон пользователей.

FF (File Flags) — простая модель, позволяющая быстро и просто настроить права на целые деревья каталогов (только чтение, только запуск, только чтение и запуск и т. д.) и отключать при необходимости наследование этих прав на определенные файлы (например, внутри каталога, настроенного только на чтение, могут быть файлы, в которые необходимо обеспечить запись).

MAC (Mandatory Access Control) — мандатная модель доступа на основе известной модели Белла-ЛаПадулы [2], только по сравнению с ней несколько модифицированная.

RC (Role Compatibility) — ролевая модель. Задаются RC-роли и RC-типы, а затем определяется, что может делать та или иная роль с тем или иным типом. Таким образом, создается некоторая абстрактная модель, которая затем привязывается к реальным пользователям, программам и файлам. Независимость модели от реальных субъектов и объектов позволяет производить мгновенную перенастройку политики безопасности быстрым перепривязыванием ролей и типов.

ACL (Access Control Lists) — списки управления доступом. ACL определяет, какие субъекты могут получать доступ к данному объекту и какие типы запросов им разрешены. Субъектом доступа может быть как простой пользователь, так и роль RC и/или группа ACL. Объекты группируются по видам, но каждый имеет собственный список ACL. Если права доступа к объекту не заданы явно, они наследуются от родительского объекта с учетом маски наследования прав. Эффективные права доступа субъекта к объекту складываются из прав, полученных непосредственно, и прав, полученных через назначение на роль или членство в группе ACL.

PM (Privacy Model). Данный модуль реализует модель безопасности, направленную на обеспечение приватности личных данных. Основная идея [3] состоит в том, чтобы пользователь мог получить доступ к персональным данным, только если они ему необходимы для выполнения текущей задачи и если он авторизован на ее выполнение. Кроме того, цели выполнения текущей задачи должны совпадать с целями, для которых эти данные собраны, либо должно быть получено согласие субъекта этих данных.

REG (Module Registration). Модуль REG позволяет подключать "на лету" модули собственного изготовления, т.е. можно описать свою модель доступа, подключить ее и использовать.

В отличие от системы LIDS (Linux Intrusion Detection System — система выявления вторжений Linux), защита которой базируется на знании пароля администратора LIDS, система LPS четко разграничивает полномочия администратора системы и администратора безопасности. Администратор системы занимается обеспечением корректности функционирования системы, а администратор безопасности — обеспечением конфиденциальности данных. Такое разделение позволяет разграничивать ответственность и выполнять требование по обязательному присутствию нескольких лиц при принятии ответственных решений.

Такой универсальный подход позволяет защитить не только конфиденциальные данные, но и данные системы, добавляя дополнительный уровень защиты. Для того чтобы преодолеть механизм защиты LPS, необходимо получить и права администратора системы и права администратора безопасности, притом что каждый из них контролирует действие другого.

Основные достоинства системы контроля доступа LPS [5]:

модульность, гибкость и настраиваемость системы — возможна тонкая настройка с точностью до пользователя, программы, системного вызова;

возможность одновременного использования нескольких политик безопасности;

реализована возможность подключения своих моделей безопасности без пересборки ядра и перезагрузки системы;

все вносимые изменения схватываются "на лету", что позволяет производить локальные изменения политики без прерывания работы критичных приложений сервера;

жесткое разделение администратора системы и администратора безопасности.

В таблице приведена зависимость времени выполнения определенного процесса от количества моделей безопасности и различных опций системы LPS. Измерения производились на ПК с процессором Celeron 333 МГц и 256 Мб ОЗУ. Каждое измерение состояло из трех запусков ‘make bzImage’ и производилось утилитой ‘time’ в однопользовательском режиме. Время приводится в секундах.

Производительность ядра ОС Linux с системой LPS

Опции LPS	Общее время	Ядро+процесс	Ядро	Процесс
Нет (чистое ядро)	711,75	711,74	34,83	676,91
RC+AUTH, никаких опций	719,36 (+1,07%)	719,35 (+1,07%)	45,41 (+30,38%)	673,94 (–0,44%)
AUTH+ACL, никаких опций	721,18 (+1,32%)	721,19 (+1,33%)	44,56 (+27,94%)	676,63 (–0,04%)
REG+FF+RC+AUTH+ACL, Net support, ind. Log (def. Config)	729,33 (+2,47%)	729,33 (+2,47%)	52,76 (+51,48%)	676,57 (–0,05%)
Все модели и опции	763,35 (+7,25%)	763,07 (+7,21%)	81,63 (+134,37%)	681,44 (+0,67%)

Из анализа таблицы видно, что прирост общего времени выполнения определенного процесса с установленным необходимым набором модулей системы LPS, составляет всего лишь 2.5%, однако при этом увеличение времени выполнения микроядерных операций достигает 50%.

К недостаткам системы LPS следует отнести ее сложность. Использование сложных моделей доступа требует очень четкого контроля за логикой преобразований и проверок доступа.

Заключение

Помимо применения системы контроля доступа LPS в защищенную ОС Bastion также интегрированы современные средства, обеспечивающие безопасное хранение (Kernel International Crypto patch) и передачу информации (Secure Shell, Gnu Privacy Guard). В направлении разграничения прав доступа в LPS необходимо реализовать технологию создания и поддержки так называемых "chrooted environments" — изолированных сред выполнения. Влияние программ, работающих в таких средах, на другие программы и систему в целом минимально. Даже в случае обнаружения уязвимости программа, запущенная в такой среде, не представляет угрозы для безопасности остальных компонентов системы.

Следует отметить, что открытое ядро ОС Linux предоставляет неограниченные возможности по модификации и гарантированному закрытию всех потенциальных брешей в обороне. Не менее важно и то, что большинство из операционных систем с открытым кодом — UNIX-подобные — наиболее изученная на сегодняшний день архитектура. Потенциальные уязвимости уже хорошо изучены и ясно, в каком направлении следует прилагать усилия. Кроме того, ввиду широкого распространения Linux в мире постоянно появляется огромное число новых, оригинальных методов защиты.

PROTECTED OPERATING SYSTEM BASTION ON THE BASIS OF LINUX KERNEL

R.KH. SADYKHOV, D.S. KOCHUROV

Abstract

Principles of construction of protected operating system Bastion, free from disadvantages of a safety-maintanence, inherent in traditional Unix-like operating systems, are offered. As a basis the monitoring system of access LPS which is oriented to Linux kernel and realizing various security models in the complex, providing their teamwork, is used.

Литература

1. *Lance Spitzner,* http://www.enteract.com/ ~lspitz/pubs.html.
2. *Bell, LaPadula,* http://osp.asu.pstu.ac.ru/ dbms/1997/01/78.htm#part5.
3. *Simone Fischer-Hubner*, Amon Ott. From a Formal Privacy Model to its Implementation, http://www.rsbac.org/niss98.htm.
4. *Хатч Б., Ли Дж., Курц Дж.* Секреты хакеров. Безопасность Linux — готовые решения: Пер. с англ. М., 2002.
5. *Садыхов Р.Х., Кочуров Д.С.* Методы и средства защиты информационных потоков в системах с открытыми кодами. Мн., 2002. (Препринт/ Ин-т техн. кибернетики НАН Беларуси: № 3).