МЕХАНИЗМЫ РЕАЛИЗАЦИИ СИСТЕМЫ БЕЗОПАСНОСТИ

ТРАДИЦИОННЫЕ МЕТОДЫ УПРАВЛЕНИЯ ДОСТУПОМ В СИСТЕМАХ UNIX

В первых и самых простых версиях UNIX никогда не было единого “центра” управления доступом. Тем не менее существовали общие правила, которые оказывали влияние на проектирование систем.

• Объекты (например, файлы и процессы) имеют владельцев. Владельцы обладают
обширным (но необязательно неограниченным) контролем над своими объектами.

• Вы являетесь владельцами новых объектов, создаваемых вами.

• Пользователь root с особыми правами, известный как *суперпользователь*, может
действовать как владелец любого объекта в системе.

• Только суперпользователь может выполнять административные операции особогозначения.

Управление доступом в файловой системе

Каждый файл в традиционной модели UNIX-подобных систем принадлежит владельцу
и группе, иногда именуемой “групповым владельцем”. Владелец файла имеет особую
привилегию, недоступную другим пользователям системы: ему разрешено менять права доступа к файлу. В частности, владелец может задать права доступа так, что никто, кроме него, не сможет обращаться к файлу
Владельцем файла всегда является один человек, тогда как в группу владельцев могут
входить несколько пользователей. По традиции информация о группах хранилась в файле **/etc/group**, но теперь ее чаще хранят на сетевом сервере NIS (англ. *Network Information Service, Информационная служба сети*) или LDAP (англ. *Lightweight Directory Access Protocol* — «облегчённый протокол доступа к каталогам»).

Владение процессом

Владелец может посылать процессу сигналы, а также понижать его приоритет. С процессами связано несколько идентификаторов: реальный, текущий и сохраненный идентификатор пользователя; реальный, текущий и сохраненный идентификатор группы, а в системе Linux — “идентификатор пользователя файловой системы”, который используется только для определения прав доступа к файлам.
Вообще говоря, реальные номера применяются для учета использования

Учетная запись суперпользователя

Учетная запись root в UNIX принадлежит “всесильному” пользователю-администратору, известному как *суперпользователь*, хотя его настоящее имя — “root”. Определяющей характеристикой учетной записи суперпользователя является значение
UID, равное нулю. Ничто не запрещает менять имя этой учетной записи или создавать другую запись с нулевым идентификатором, но такие действия ни к чему хорошему
не приведут. Их следствием будет возникновение новых брешей в системе защиты,
а также растерянность и гнев других администраторов, которым придется разбираться
с особенностями конфигурирования такой системы.
Традиционная система UNIX позволяет суперпользователю (т.е. всякому процессу,
текущий идентификатор пользователя которого равен нулю) выполнять над файлом или
процессом любую допустимую операцию.

Вот примеры операций, доступных лишь суперпользователю:
- изменение корневого каталога процесса с помощью команды chroot;
- создание файлов устройств;
- установка системных часов;

Использование битов "setuid" и "setgid "

Традиционная система управления доступом в UNIX дополнена системой смены полномочий, которая реализуется ядром в сотрудничестве с файловой системой, которая позволяет выполнять специально подготовленные файлы с использованием привилегий более высокого уровня (обычно это привилегии суперпользователя). Этот механизм разрешает разработчикам и администраторам создавать условия для непривилегированных пользователей, при которых они могут выполнять привилегированные операции.

Дело в том, что существует два специальных бита, устанавливаемых в маске прав доступа к файлу: “setuid” (Set User ID — бит смены идентификатора пользователя) и “setgid” (Set Group ID — бит смены идентификатора группы). Если запускается исполняемый файл, у которого установлен один из этих битов, то текущими идентификаторами создаваемого процесса становятся идентификаторы владельца файла, а не идентификаторы пользователя, запустившего программу. Смена полномочий действительна только на время работы программы.

Например, пользователи должны иметь возможность изменять свои пароли. Но поскольку пароли хранятся в защищенном файле **/etc/shadow**, пользователям нужно использовать команду passwd с полномочиями setuid, чтобы “усилить” свои права. Следует подчеркнуть важность слова “допустимую”. Некоторые операции (например, запуск файла, для которого не установлен бит выполнения) запрещены даже суперпользователю.

Команда passwd проверяет, кто ее выполняет, и, в зависимости от результата, настраивает свое поведение соответствующим образом, поэтому обычные пользователи

могут менять только собственные пароли, а суперпользователь — любые. (Это, между

ОЧЕВИДНЫЕ НЕДОСТАТКИ ЭТОЙ МОДЕЛИ

• С точки зрения безопасности, суперпользователь представляет единственную учетную запись, которая может являться причиной потенциального отказа системы.Если суперпользователь дискредитирует себя, может нарушиться целостность всейсистемы. Взломщик в этом случае может нанести системе колоссальный вред.

• Единственный способ разделить специальные привилегии суперпользователя состоит в написании setuid-программ. К сожалению, как показывает непрекращающийся интернет-поток обновлений средств защиты систем, довольно трудно написать по-настоящему безопасное программное обеспечение. К тому же, вам не следует писать программы, назначение которых можно было бы выразить такими словами: “Хотелось бы, чтобы эти три пользователя могли выполнять задачи резервирования на файловом сервере”.

• Модель безопасности не является достаточно прочной для использования в сети. Ни один компьютер, к которому непривилегированный пользователь имеет физическийдоступ, не может гарантировать, что он точно представляет принадлежностьвыполняемых процессов. Кто может утверждать, что такой-то пользовательне переформатировал диск и не инсталлировал собственную копию Windows или Linux со своими UID?

• Во многих средах с высокими требованиями к защите системы действуют соглашения, которые просто не могут быть реализованы с использованием традиционных UNIX-систем безопасности. Например, согласно государственным стандартам США компьютерные системы должны запрещать привилегированным пользователям (например, тем, кто относится к высшей категории допуска) публиковать документы особой важности на более низком уровне безопасности.

• Поскольку многие правила, связанные с управлением доступа, встроены в код отдельных команд и демонов, невозможно переопределить поведение системы, не модифицируя исходный код и не перекомпилируя программы. Но это реально непрактикуется.

• Для отслеживания действий пользователей предусмотрена минимальная поддержка. Вы можете легко узнать, к каким группам принадлежит тот или иной пользователь, но вы не в состоянии точно определить, какие действия разрешены пользователючленством в этих

Управление доступом на основе ролей

Управление доступом на основе ролей (*r*ole-*b*ased *a*ccess *c*ontrol — **RBAC**) — теоретическая модель, формализованная в 1992 г. Дэвидом Ферраильо (David Ferraiolo) и Риком Куном (Rick Kuhn). В основе этой модели лежит идея добавления уровня косвенности в механизм управления доступом. Вместо того чтобы назначать полномочия непосредственно пользователям, они назначаются промежуточным конструкциям, именуемым “ролями”, а роли, в свою очередь, назначаются пользователям. Для того чтобы принять решение по управлению доступом, специальная библиотека перечисляет роли текущего пользователя и узнает, имеет ли хотя бы одна из этих ролей соответствующие полномочия.

Между ролями и UNIX-группами можно заметить некоторое сходство, и нет единого мнения насчет того, отличаются ли эти конструкции по сути. На практике роли оказываются более полезными, чем группы, поскольку системы, в которых они реализованы, разрешают использовать их вне контекста файловой системы. Роли могут также иметь иерархические взаимоотношения друг с другом, что значительно упрощает администрирование.

Например, вы могли бы определить роль “старшего администратора”, который имеет все полномочия “администратора”, а также дополнительные полномочия X, Y и Z.

SELinux: Linux-системы с улучшенной безопасностью

Операционная система SELinux (как проект) был разработан Агентством национальной безопасности США (АНБ), но с конца 2000 года был передан разработчикам открытого кода. Система SELinux включена в состав ядра Linux (с версии 2.6) и поэтому в настоящее время доступна в большинстве дистрибутивов (но часто в не совсем дееспособном состоянии).

SELinux — это реализация системы принудительного управления доступом доступа (*m*andatory *a*ccess *c*ontrol — MAC), в которой все привилегии назначаются администраторами.
В среде MAC пользователи не могут делегировать свои права и не могут устанавливать параметры контроля доступа на объекты, которыми они (пользователи) владеют. И поэтому такая операционная система подходит больше для узлов со специальными требованиями

Подключаемые модули аутентификации

Подключаемые модули аутентификации (Pluggable Authentication Modules — **PAM**) образуют технологию аутентификации, а не технологию управления доступом. Другими словами, технология PAM призвана искать ответ не на вопрос: “Имеет ли право пользователь
X выполнить операцию Y?”, а на вопрос: “Как узнать, что это действительно пользователь X?” Технология PAM — это важное звено в цепи управления доступом в большинстве систем.

Сетевой протокол криптографической аутентификации Kerberos

Подобно PAM, протокол Kerberos призван решать проблемы аутентификации, а не управления доступом. (Он назван в честь трехголового пса, который защищал вход в царство Аида, — Цербера, или Кербера.) Но если PAM можно назвать структурной оболочкой аутентификации, то **Kerberos** — это конкретный метод аутентификации. Реализация Kerberos использует проверенный (сторонний) сервер для выполнения задач аутентификации в масштабах всей сети. Вместо самоаутентификации на своем компьютере вы предоставляете свои мандаты (билеты) Kerberos-службе, а она выдает вам криптографические мандаты, которые вы можете презентовать другим службам как подтверждение вашей идентичности.

Списки управления доступом

Управление доступом к файловой системе — важнейшая часть систем UNIX и Linux, и поэтому ее совершенствование составляет первоочередную цель разработчиков этих
систем. Особое внимание было уделено поддержке списков доступа к файлам и каталогам (*a*ccess *c*ontrol lists — **ACL**) как обобщению традиционной модели привилегий пользователя/группы/“всех остальных”, устанавливаемых сразу для нескольких пользователей и групп. Списки ACL являются частью реализации файловой системы, поэтому их поддержку в явном виде должна обеспечивать используемая вами файловая система. В настоящее время практически все файловые системы UNIX и Linux поддерживают ACL-списки в той или иной форме.

Управление доступом в реальном мире

Несмотря на наличие описанных выше превосходных возможностей операционных
систем, в большинстве узлов для задач системного администрирования по-прежнему
используется учетная запись суперпользователя. Многие жалобы на традиционную систему имеют реальную почву, но и альтернативным вариантам присущи серьезные проблемы. Поэтому особое значение приобретают такие дополнительные программные инструменты, как **sudo**, которые в некоторой степени позволяют
преодолеть разрыв между критериями простоты использования и безопасности.

Часто для принятия решений в особых обстоятельствах используются возможности
POSIX или средства управления доступом на основе ролей (например, когда необходимо разрешить переустановку принтера или демона каждому, кто работает в конкретном отделе), в то время как при решении каждодневных задач ваша административная команда продолжает полагаться на утилиту sudo и учетную запись суперпользователя.

В некоторых случаях для узлов необходимо использовать такие мощные и “ударопрочные” системы, как SELinux. Поскольку доступ с правами суперпользователя является обязательным условием системного

Команда su: замена идентификатора пользователя

Доступ к учетной записи root рекомендуется осуществлять с помощью команды su. Без аргументов она выдает приглашение на ввод пароля суперпользователя, а затем запускает интерпретатор команд с правами пользователя root. Интерпретатор будет выполняться в привилегированном режиме, пока не завершит работу. Команда su не фиксирует действия, производимые в среде интерпретатора, но добавляет запись в журнальный файл с указанием, кто и когда вошел в систему под именем суперпользователя. Команда su способна также подставлять вместо имени root имена других пользователей.

Утилита sudo: ограниченный вариант команды su

Если учетная запись root доступна группе администраторов, то вы невозможно однозначно идентифицировать КТО и ЧТО делает. Поэтому для получения прав суперпользователя применяется утилита sudo. Утилита sudo в качестве аргумента принимает командную строку, которая подлежит выполнению от имени root- пользователя (или другого уполномоченного пользователя). Утилита обращается к файлу **/etc/sudoers**, где содержится список пользователей, имеющих разрешение на ее выполнение, и перечень команд, которые они могут вводить на конкретном компьютере. Если запрашиваемая команда разрешена, утилита sudo приглашает пользователя ввести его собственный пароль и выполняет команду от имени суперпользователя

```
# cp /usr/local/lib/skel/.[a-zA-Z]* ~tyler
# chown tyler:staff ~tyler/.[a-zA-Z]*
# chmod 600 ~tyler/.[a-zA-Z]*

# chown tyler:staff ~tyler/.*
```

О псевдопользователях

Только пользователь root имеет для ядра Linux особый статус. Есть, однако, еще несколько псевдопользовательских учетных записей, которые применяются для системных целей. Эти фиктивные учетные записи можно идентифицировать по значениям UID, которые обычно меньше 100. Как правило, учетные записи с UID меньше 10 принадлежат системе, а значения UID от 10 до 100 отведены для псевдопользователей, связанных со специальными программами.
Пароли этих псевдопользователей в файле **/etc/shadow** обычно заменяют звездочкой,
чтобы нельзя было войти в систему под служебным именем. Чтобы защититься от средств атаки на основе дистанционного входа в систему (когда вместо паролей используются файлы SSH-ключей), укажите в качестве командных интерпретаторов (вместо **/bin/bash** или **/bin/sh**) вариант **/bin/false** или **/bin/nologin**.

Файлы и процессы, которые являются частью операционной системы, но не должны
принадлежать пользователю root, иногда передаются во владение пользователям bin
или daemon. Считается, что это поможет избежать риска, связанного с действиями от
имени суперпользователя. Однако ввиду неубедительности подобной аргументации в
современных системах используется просто учетная запись root.