Дополнительные атрибуты файлов и безопасность файловых систем EXT 2, 3, 4.

Вступление

Дополнительные атрибуты для файлов и каталогов, поддерживают ext2 и выше.
Концепция большинства полезных атрибутов была реализована еще в ядрах начиная с серии 1.1.
Это простые дополнительные возможности укрепления безопасности, но, используется редко.
Например, используются в утилитах для поддержки целостности системы: LIDS, Tripwire и др.

Что такое атрибуты файловой системы ext2?

Файловая система в ядрах серий 2.2, 2.4 и 2.6 позволяет работать со следующими атрибутами:

A Atime система не апдейтит atime(access time) для данного файла.

S Sync система фиксирует все изменения, происходящие в данном файле на физическиом диске синхронно с приложением изменяющим данный файл.

a Append система позволяет открывать данный файл с целью его дополнения и не позволяет никаким процессам перезаписывать или усекать его.Если данный атрибут применяется к директории процесс может создавать

или модифицировать файлы в этой директории , но не удалять их.

i Immutable система запрещает любые изменения данного файла. В случае директории, процессы могут модифицировать файлы уже содержащиеся в данной директории, но не могут удалять файлы или создавать новые.

d No Dump программе создающей дампы дается указание игнорировать данный файл во время создания backup.

c Compress система применяет прозрачную компрессию для данного файла. Т.е. чтение из него дает уже декомпрессованые данные и предварительно перед записью на диск производится их сжатие.

s Secure Deletion удаление такого файла сопровождается перезаписью блоков диска,на которых он располагался нулями.

u Undelete когда приложение запрашивает файл на удаление, система должна сохранить блоки на диске, на которых расположен данный файл, чтобы потом его можно было восстановить.

Несмотря на то, что файловая система поддерживает данный набор атрибутов, у ядра и

различных приложений остается выбор, учитывать или не учитывать их.

В таблице приведено соответствие, как различные версии ядра учитывают каждый атрибут:
* позволяет устанавливать флаг и учитывает его значение
i позволяет устанавливать флаг,но игнорирует его значение
- полностью игнорирует флаг

	1.0	1.2	2.0	2.2	2.4
A	-	-	*	*	*
S	*	*	*	*	*
a	-	*	*	*	*
i	-	*	*	*	*
d	-	*	*	*	*
c	i	i	i	i	i
s	*	*	i	i	i
u	i	i	i	i	i

Хотя ядра более ранних версий учитывают флаг 'secure deletion', но позже разработчики исключили этот атрибут из последующих серий, поскольку, на их взгляд, его использование повышает общую безопасность лишь незначительно. А в худшем случае создает ложное чувство безопасности у пользователей незнакомых с проблемами свойственными для любой схемы 'secure deletion'.

Флаг 'A' или 'atime' для определенных файлов может дать некоторую прибавку производительности, так как избавляет систему от необходимости апдейтить access time для этих файлов каждый раз, когда их открывают на чтение.

Атрибут 'S' или 'synchronous предоставляет новый уровень поддержки целостности данных. Но

поскольку все изменения в файлах немедленно сохраняются на диске несколько понижается производиельность.

Основное внимание мы уделим флагу 'a' ('append only') и 'i' ('immutable'), так как их использование дает наиболее ощутимые приемущества в плане обеспечения безопасности и целостности файловой системы.

Различные open source BSD системы, такие как FreeBSD и OpenBSD, также поддерживают аналогичные флаги в своих файловых системах (UFS и FFS соответственно).

Какие команды используются для установки и чтения атрибутов в ext2?

Стандартная команда ls не показывает эти расширенные атрибуты. Но, наборе утилит для ext2 содержит 2 специальные утилиты для установки и чтения данных атрибутов: chattr и lsattr.

Поскольку ext2 рассматривается как основная рабочая файловая система Linux, почти все дистрибутивы содержат package 'e2fsprogs', инсталлированый по умолчанию.

Команда lsattr поддерживает очень небольшой набор опций, вот наиболее важные:

-a вывод листинга всех директорий и файлов
-d листинга директорий, но без вывода их содержимого

Результа использования этих опций похож на команду ls. Я обычно создаю шел alias, делая 'la' эквивалентным 'lsattr -a', поскольку последняя становится нудной в случае частого использования.

Команда chattr может использоваться тремя различными способами:
chattr +Si test.txt -- установить флаги sync и immutable для файла test.txt

chattr -ai test.txt -- убрать флаги append-only и immutable y test.txt

chattr =aiA test.txt -- установить ограничение на использование только флагов a, i и A
Наконец, обе команды поддерживают опцию -R ,т.е. рекурсивная работа.

Как отличаются атрибуты ext2 от обычных прав доступа?

Большинство администраторов хорошо знакомы со стандартным управлением правами доступа:
[root@typhoid shaffer]# ls -al test*

-rw-rw-r-- 1 shaffer users 0 Nov 17 17:02 test.conf
-rw-rw-r-- 1 shaffer users 0 Nov 17 17:02 test.logM
-rw-rw-r-- 1 shaffer users 0 Nov 16 19:41 test.txt

Вывод команды ls показывает, что владелец всех файлов test* - shaffer и члены группы имеют право на чтение и запись в эти файлы. Прочим пользователям эти файлы доступны для чтения. Команда lsattr примененная к той же группе файлов выдаст уже другие данные:

[root@typhoid shaffer]# lsattr -a test*
---i-------- test.conf
----a------- test.log
------------ test.txt

Она показывает, что test.log имеет флаг append-only, а test.conf имеет флаг immutable.

Копцепция поведения, при которой root или процесс запущенный от имени root в обычных условиях могут игнорировать расширенные атрибуты, была реализована во многих unix системах и является одной из основных причин успешной работы многих локальных и удаленных exploit'ов.

С другой стороны атрибуты ext2 учитываются различными системными вызовами, такими как sys_open() и sys_truncate(), причем вне зависимости от uid процесса их вызывающего или каких-либо других условий. Присутствие флага immutable в inode приводит к тому, что все системные вызовы, касающиеся модификации файлов перестают работать в не зависимости от других условий.

Наличие данных атрибутов и специальных режимов работы ядра в Linux позволяет администратору просто и эффективно укрощать традиционно абсолютные возможности, которые дает uid 0. Цель комплексной настройки заключается в том, чтобы флаги a и i должны накладывать ограничения для всех процессов безотносительно их правам доступа и уровню привилегий.

Они могут служить в качестве эффективной низкоуровневой защиты против атак на любой привилегированный процесс, в котором могут присутствовать какие-либо неизвестные уязвимости(или он не пропатчен от уже известных). Необходимо также добавить, что эти атрибуты дают защиту только на уровне поддерки целостности файловой системы, то есть от атак строго определенного типа. Поэтому в случае denial-of-service это не поможет.

Наконец, использование для защиты критичных файлов атрибутов файловой системы само по себе является полумерой. Хотя атрибуты a и i предотвратят изменение защищенных ими файлов даже со стороны процессов, владельцем которых является root, в обычных обстоятельствах super user все равно может убрать эти флаги и продолжить работу уже без этой помехи. То есть вся проблема в обычных обстоятельствах свелась бы к тому, что в большинстве exploits перед основной работой добавилась бы проверка на то, что эти атрибуты сняты.

В ядрах серий до 2.1, свойство ядра 'secure level' (уровень безопасности) использовалось 'узким кругом' администраторов, поскольку его значение большее нуля означало запрет на любые изменения в файлах с атрибутами a и i.

Для этих ранних версиях ядер, уровень безопасности пределялось через переменную sysct "kernel.securelevel". Если ее значение выставлялось во время загрузки единицей или более высоким уровнем безопасности, то тогда система не позволяла изменять файлы с атрибутами immutable и append-only до момента перегрузки в режим single-user .

В ядрах версий выше 2.0, новая более гибкая система настройки параметров ядра позволяет конфигурировать систему в похожем режиме полной защиты файлов с атрибутами immutable и append-only.

Утилита lcap(8) позволяет конфигурировать множество параметров ядра, в том числе те, которые определяют работу ext2 с расширенными атрибутами. Вот наиболее важные вызовы lcap, которые нас будут интересовать:

```
lcap CAP_LINUX_IMMUTABLE
lcap CAP_SYS_RAWIO
```

Первый параметр убирает привилегию у процесов root изменять флаги a и i, второй - запрещает низкоуровневый доступ к блочным устройствам, таким как диски, чтобы предотвратить изменение флагов, используя прямой доступ к файлам. Свойство CAP_SYS_RAWIO всегда необходимо убирать в начале процесса загрузки, если вы собираетесь испльзовать специальные возможности ядра, поскольку в противном случае процесс, владельцем которого является root, может изменить установки параметров ядра ('kernel capability bounding set') обращаясь напрямую

к памяти через /dev/kmem.

Если запустить lcap без параметров, то она выдаст полный список 'kernel capability bounding set' и выделит в нем те , которые в данный момент доступны. Ссылка на статью, где обсуждаются kernel capabilities приводится ниже.

После того, как помощью lcap был изменен какой-либо параметр из 'kernel capability bounding set', его повторное изменение возможно только после перезагрузки системы. Эта особенность дает уверенность (для правильно сконфигурированных систем), что в система не могут незаметно производиться изменения без получения физического доступа и перезагрузки в режиме single user.
Дополнительная информация по lcap также как ее исходный код доступны здесь:

LCAP - Linux Kernel Capabilities Bounding Set Editor: http://pw1.netcom.com/~spoon/lcap/

Вещи, для которых обычно используют 'chattr'

Когда я конфигурирую системы, которые будут затем напрямую подключены к Internet и на которых предполагается хостить shell аккаунты или службы типа http и ftp, то обычно после настройки всего необходимого ПО и пользовательских аккаунтов выполняю следующие команды:

```
chattr -R +i /bin /boot /etc /lib /sbin
chattr -R +i /usr/bin /usr/include /usr/lib /usr/sbin
chattr +a /var/log/messages /var/log/secure (...)
```

Если новые аккаунты добавляются, изменяются, удаляются относительно редко, тогда установка на /home(но не на содержащиеся в нем личные каталоги пользователей) атрибута immutable не вызывет особых проблем. Во многих случаях на все поддерево /usr можно поставить атрибут immutable. Дополнительно, помимо 'chattr -R +i /usr', я обычно располагаю /usr на отдельной разделе и монтирую его как read-only с опцией 'ro' в /etc/fstab. Установка флага append-only на системные лог файлы будет значительным препятствием для потенциального взломщика, пытающегося скрыть свое присутствия и действия в системе.

Конечно, система, в которой обеспечивается безопасность подобным образом, требует соответствующего изменения стандартных процедур администрирования. Расмотрим некоторые аспекты этого.

Инсталляция и апгрейд ПО

Для инсталляции и апгрейда необходимо удалить флаги a и i у тех каталогов и файлов, которые будут подвергаться изменению. В системах , где используется rpm, команда:

rpm -qpl newpackage.rpm

выдаст список файлов в устанавливаемом package. Для большинства инсталляций с использованием rpm необходимо иметь права на запись в следующие каталоги:

/bin
/sbin
/usr/bin
/usr/sbin
/usr/man
/lib
/etc

Обратите внимание, чтобы надлещащим образом организовать замену файла /usr/sbin/someprogram обычно потребуется убрать флаги immutable как с самого файла, так и с содержащей его директории /usr/sbin.

Управление пользовательскими и групповыми аккаунтами

Все из нижеперечисленного

/etc
/etc/.pwd.lock
/etc/passwd
/etc/passwd-
/etc/shadow
/etc/shadow-
/etc/group

/etc/group-
/etc/gshadow
/etc/gshadow-

должно остаться доступным на запись и удаление, поскольку команды passwd, chsh, chfn, vipw, vigr и useradd используют эти файлы. Они создают их временную копию в /tmp, модифицируют ее, затем удаляют исходный файл и записывают на его место копию.

Вещи, для которых никогда не следует применять chattr:

/

Это создаст множество проблем(в особенности для syslog). В первую чередь проблемы коснутся syslogd. У него не будет возможности записывать в логи ошибки, появляющиеся в результате работы других приложений. В результате мы получим головоломную проблему. Когда я впервые стал эсперементировать с атрибутами ext2, то потратил целый день на разгребание проблем после того, как дал команду 'chattr +i /'.

/dev

Syslog удаляет и создает заново сокет /dev/log во время своего старта и установка флага immutable или append only на /dev не позволит выполнить это. Кроме этого, когда скрипт запуска выполняет syslogd с опцией '-p', которая задает альтернативный сокет, такой как /var/run/syslog.sock.

Более того даже, когда syslog запускается с опцией '-p', приложения использующие syslog будут

все равно предполагать , что сокет расположен в /dev/log, поэтому необходимо создать символьную ссылку , указывающую на место реального сокета. Наконец, даже этого не будет достаточно для систем, которые используют lpd, поскольку package lpr содержит /dev/printer в качестве жесткой ссылки на lpd service socket. И lpd пытается удалить и создать заново свой сокет перед запуском. Изменение этого пути требует перекомпиляции всего пакета lpr.

/tmp

Установка атрибутов на данный каталог будет препятствием для всех приложений, которые хранят какие-либо временные файлы в нем.

/var

Атрибуты append only и immutable в /var относительно редко следует применять к файлам, для которых их установка не вызывает побочных эффектов. Например, применение chattr +a для большинства файлов /var/log не создаст проблем, за некоторым исключением(logrotate). Контрпример: файл /var/log/sendmail.st должен остаться доступным для записи и удаления, поскольку sendmail не просто добавляет в него статистику, но и периодически урезает и перезаписывает.