Пингвин под колпаком: Аудит системных событий в Linux

Журнал «Хакер» 30.03.2011 0 55409

Мобильная версия статьи

Содержание статьи

WARNING
INFO
WWW

Грамотная после-установочная настройка Linux-дистрибутива – лишь первый шаг на пути к безопасности и стабильности. Чтобы операционная система и дальше продолжала отвечать этим требованиям, придется приучить себя к постоянному слежению за ее состоянием. Этого можно достичь с помощью мониторинга и анализа лог-файлов, однако получение полного контроля над системой возможно только с помощью аудита системных событий.

Перед тем как перейти к основной части статьи, постараемся разобраться с тем, что же все-таки представляет собой системный аудит (или аудит системных событий). По самой своей сути это не что иное, как постоянное и подробное протоколирование любых событий, происходящих в операционной системе. Аудиту могут быть подвержены такие события, как чтение/запись файлов, выполнение входа в ОС, запуск и остановка приложений, инициация сетевого соединения и многое, многое другое. Linux, как и любая другая современная серверная ОС, включает в себя все необходимые инструменты для проведения аудита системы, однако разобраться в них с наскоку не так-то просто.

Система аудита ядра 2.6

Начиная с версии 2.6, ядро Linux включает в себя подсистему, специально разработанную для проведения аудита. Она позволяет вести слежение за такими системными событиями, как:

- Запуск и завершение работы системы (перезагрузка, остановка);
- Чтение/запись или изменение прав доступа к файлам;
- Инициация сетевого соединения или изменение сетевых настроек;
- Изменение информации о пользователе или группе;
- Изменение даты и времени;
- Запуск и остановка приложений;
- Выполнение системных вызовов.

На деле этот список гораздо длиннее, но другие типы событий не представляют для нас практического интереса. Кроме самого факта возникновения события, система аудита представляет такую информацию, как дата и время возникновения события, ответственность пользователя за событие, тип события и его успешность. Сама по себе она способна лишь сообщать о произошедшем событии, тогда как процесс журналирования возлагается на плечи демона auditd.

Установка

Демон auditd доступен в любом современном Linux-дистрибутиве и может быть установлен с помощью стандартного менеджера пакетов. Например, чтобы установить auditd в Debian/Ubuntu, достаточно выполнить следующую команду:

```
$ sudo apt-get install auditd
```

Кроме самого демона, вместе с пакетом будут установлены три подсобные утилиты, используемые для управления системой аудита и поиска записей в журнальных файлах:

- auditctl – утилита, управляющая поведением системы аудита. Позволяет узнать текущее состояние системы, добавить или удалить правила;
- autrace – аудит событий, порождаемых указанным процессом (аналог strace);
- ausearch – команда, позволяющая производить поиск событий в журнальных файлах;
- aureport – утилита, генерирующая суммарные отчеты о работе системы аудита;

По умолчанию система аудита находится в спящем состоянии и не реагирует ни на какие события (ты можешь убедиться в этом, выполнив команду «sudo auditctl -l»). Чтобы заставить систему сообщать нам подробности каких-либо событий, необходимо добавить правила. Но чтобы понять, как их составлять, придется разобраться с тем, как работает подсистема аудита ядра Linux.

Аудит системных событий

Ни одно событие в любой операционной системе не может произойти без использования системных вызовов ядра. Запуск нового процесса, открытие файлов и работа с ними, запрос времени или типа ОС, обращение к оборудованию, создание сетевого соединения, вывод информации на экран – все эти операции производятся с помощью обращения к функциям ядра операционной системы, для краткости называемых системными вызовами. Если приложение не прибегает в своей работе к вызовам ядра, оно оказывается замкнутым в самом себе и просто не способно к какому-либо взаимодействию со своим окружением, не говоря уже о пользователе. Следовательно, чтобы отследить любое системное событие, достаточно просто перехватывать все обращения к системным вызовам.

Именно это делает подсистема аудита. Она устанавливает триггеры до и после всех функций, ответственных за обработку системных вызовов, и ждет. Когда происходит системный вызов, триггер срабатывает, подсистема аудита получает всю информацию о вызове и его контексте, передает ее демону auditd и отдает дальнейшее управление функции, обрабатывающей системный вызов. После ее завершения срабатывает «выходной» триггер, и вся информация о системном вызове вновь поступает к подсистеме аудита и демону auditd.

Естественно, держать все эти триггеры в активном состоянии все время работы ОС накладно и слишком избыточно (только во время старта приложение может выполнить несколько тысяч системных вызовов и для полного анализа журнала аудита придется

убить уйму времени). Поэтому по умолчанию триггеры отключены и могут быть выборочно активированы с помощью правил, которые задают имя системного вызова, его успешность, пользователя, от имени которого произошел вызов и т.д.

Благодаря такой нехитрой схеме, системный администратор может вести наблюдение за любым аспектом работы операционной системы (кроме, конечно же, тех событий, которые вызваны компонентами самого ядра). Поэтому, если в системе происходят какие-либо подозрительные действия, вызванные работой взломщика или вредоносного ПО, с помощью системного аудита не составит труда их выявить и найти виновного.

Правила аудита

Для создания, удаления и модификации правил аудита предназначена утилита auditctl. Есть три основных опции, которые принимает эта команда:

- -a – добавить правило в список;
- -d – удалить правило из списка;
- -D – удалить все правила;
- -l – вывести список заданных правил.

Если ты сейчас выполнишь команду «auditctl -l» от имени администратора, то, скорее всего, увидишь «No rules», а это значит, что ни одного правила аудита еще не существует. Для добавления правил используется следующая форма записи команды auditctl:

```
# auditctl -a список,действие -S имя_системного_вызова -F фильтры
```

Здесь список – это список событий, в который следует добавить правило. Всего существует пять списков:

- task – события, связанные с созданием процессов;
- entry – события, происходящие при входе в системный вызов;
- exit – события, происходящие во время выхода из системного вызова;
- user – события, использующие параметры пользовательского пространства, такие как uid, pid и gid;
- exclude – используется для исключения событий.

Не беспокойся, если смысл списков кажется тебе непонятным. По сути, это всего лишь фильтры, которые позволяют сделать правила более точными. На деле из них используются только entry и exit, которые позволяют зарегистрировать либо сам факт обращения к системному вызову, либо его успешную отработку.

Второй параметр опции – '-a' – это действие, которое должно произойти в ответ на возникшее событие. Их всего два: never и always. В первом случае события не записываются в журнал событий, во втором – записываются.

Далее указывается опция '-S', которая задает имя системного вызова, при обращении к которому должен срабатывать триггер (например, open, close, exit, и т.д.). Вместо имени может быть использовано числовое значение.

Необязательная опция '-F' используется для указания дополнительных параметров фильтрации события. Например, если мы хотим вести журнал событий, связанных с использованием системного вызова open(), но при этом желаем регистрировать только обращения к файлам каталога /etc, то должны использовать следующее правило:

```
# auditctl -a exit,always -S open -F path=/etc/
```

Чтобы еще более сузить круг поисков, сделаем так, чтобы регистрировались только те события, при которых файл открывается только на запись и изменение атрибутов:

```
# auditctl -a exit,always -S open -F path=/etc/ -F perm=aw
```

Здесь 'a' – изменение атрибута (то есть attribute change), а 'w' – запись (то есть write). Также можно использовать 'r' – чтение (read) и 'x' – исполнение (execute). Другие полезные фильтры включают в себя: pid – события, порождаемые указанным процессом, apid – события, порождаемые указанным пользователем, success – проверка на то, был ли системный вызов успешным, a1, a2, a3, a4 – первые четыре аргумента системного вызова. Фильтр key используется для указания так называемого ключа поиска, который может быть использован для поиска всех событий, связанных с этим ключом. Количество возможных фильтров достаточно велико, их полный список можно найти в man-странице auditctl.

Вообще, для слежения за файлами в auditctl предусмотрен специальный синтаксис, при котором опцию '-S' можно опустить. Например, описанное выше правило может быть задано следующим образом (здесь опция '-p' – это эквивалент фильтра perm):

```
# auditctl -a exit,always -F dir=/etc/ -F perm=wa
```

Или используя более короткую форму (здесь оцпия '-p' – это эквивалент фильтра perm, а '-k' – фильтра key):

```
# auditctl -w /etc/ -p wa -k access_etc
```

Таким же образом может быть установлена «слежка» за любым индивидуальным файлом:

```
# auditctl -w /etc/passwd -p wa
```

Конфигурационные файлы

Правила не обязательно задавать, используя командную строку. Во время старта демон auditd читает два файла: /etc/audit/auditd.conf и /etc/audit/audit.rules (в некоторых дистрибутивах они могут находиться прямо в /etc). Первый описывает конфигурацию демона и содержит такие опции, как имя журнала, его формат, частота обновления и другие параметры. Нет смысла их изменять, разработчики дистрибутива уже позаботились о грамотной настройке.

Второй файл содержит правила аудита в формате auditctl, поэтому все, что нужно сделать, чтобы получить правило, пригодное для записи в этот файл – просто опустить имя команды. Например:

```
-w /etc/passwd -p wa
```

В начале файла обычно размещаются несколько мета-правил, задающих конфигурацию и очищающих правила:

```
# Удаляем все правила из всех списков
-D

# Указываем количество буферов, хранящих сообщения аудита
-b 8192

# Что делать в чрезвычайной ситуации (например, если все буферы будут заполнены)
```

0 – ничего не делать

1 – поместить сообщение в dmesg

2 – отправить ядро в панику (kernel panic)

```
-f 1
```

Далее можно разместить свои правила.

Примеры правил

Первое, что следует сделать – настроить наблюдение за всеми конфигурационными и лог-файлами системы аудита. Если взломщик сможет компрометировать один из них, он наверняка изменит поведение демона, и мы просто не сможем найти его следы.

 # vi /etc/audit/audit.rules

```
# Наблюдение за конфигурационными файлами
-w /etc/audit/auditd.conf -p wa
-w /etc/audit/audit.rules -p wa
-w /etc/libaudit.conf -p wa
-w /etc/default/auditd -p wa

# Наблюдение за журнальными файлами
-w /var/log/audit/
-w /var/log/audit/audit.log
```

Далее установим наблюдение за наиболее важными системными конфигурационными файлами. Изменение хотя бы одного из них может привести к открытию дыры в безопасности системы, чего допустить нельзя. Обрати внимание, что некоторые файлы актуальны только для дистрибутивов Debian/Ubuntu!

 # vi /etc/audit/audit.rules

```
# Настройки и задания at
-w /var/spool/at
-w /etc/at.allow
-w /etc/at.deny

# Задания cron
-w /etc/cron.allow -p wa
-w /etc/cron.deny -p wa
-w /etc/cron.d/ -p wa
-w /etc/cron.daily/ -p wa
-w /etc/cron.hourly/ -p wa
-w /etc/cron.monthly/ -p wa
-w /etc/cron.weekly/ -p wa
-w /etc/crontab -p wa
-w /var/spool/cron/root

# Файлы паролей и групп
-w /etc/group -p wa
-w /etc/passwd -p wa
-w /etc/shadow

# Конфигурационные и журнальные файлы входа в систему
-w /etc/login.defs -p wa
-w /etc/securetty
```

```
-w /var/log/faillog
-w /var/log/lastlog

# Список и имена хостов
-w /etc/hosts -p wa

# Стартовые скрипты демонов
-w /etc/init.d/
-w /etc/init.d/auditd -p wa

# Пути поиска библиотек
-w /etc/ld.so.conf.d
-w /etc/ld.so.conf -p wa

# Настройки времени
-w /etc/localtime -p wa

# Системные переменные
-w /etc/sysctl.conf -p wa

# Правила загрузки модулей
-w /etc/modprobe.d/

# Модули системы PAM
-w /etc/pam.d/

# Настройки сервера SSH
-w /etc/ssh/sshd_config
```

Настроим наблюдение за всеми системными вызовами, которые могут угрожать безопасности системы. Эти правила следует применять только в случае особой необходимости, они создадут высокую нагрузку на систему аудита и приведут к существенному разрастанию журнальных файлов, однако взамен ты получишь тотальный контроль над системой и сможешь легко выявить малейшее нарушение ее работы.

 # vi /etc/audit/audit.rules

```
# Изменение прав доступа к файлам
-a entry,always -S chmod -S fchmod -S chown -S chown32 -S fchown -S
fchown32 -S lchown -S lchown32

# Создание, открытие или изменение размеров файлов
-a entry,always -S creat -S open -S truncate -S truncate64 -S ftruncate
```

```
-S ftruncate64

# Создание и удаление каталогов
-a entry,always -S mkdir -S rmdir

# Удаление или создание ссылок
-a entry,always -S unlink -S rename -S link -S symlink

# Изменение расширенных атрибутов файлов
-a entry,always -S setxattr
-a entry,always -S lsetxattr
-a entry,always -S fsetxattr
-a entry,always -S removexattr
-a entry,always -S lremovexattr
-a entry,always -S fremovexattr

# Создание файлов устройств
-a entry,always -S mknod

# Монтирование файловых систем
-a entry,always -S mount -S umount -S umount2

# Использование системного вызова ptrace для отладки процессов
-a entry,always -S ptrace
```

Анализ журнальных файлов

Журнальные файлы, создаваемые демоном auditd в каталоге /var/log/audit, не предназначены для чтения человеком, но хорошо подходят для анализа с помощью специальных утилит, устанавливаемых вместе с самим демоном. Самая важная из них – утилита aureport, генерирующая отчеты из лог-файлов. Вызвав ее без аргументов, мы узнаем общую статистику использования системы, включая такие параметры, как количество входов и выходов из системы, открытых терминалов, системных вызовов и т.д. Эта информация малоинтересна, поэтому лучше запросить более детальный отчет. Например, запустив команду с флагом '-f', мы получим список файлов, к которым происходил доступ:

```
$ sudo aureport -f
```

Скорее всего вывод будет слишком длинным, но его можно сократить с помощью запроса информации только за определенный период времени (аргумент '--end' не обязателен):

```
$ sudo aureport -f --start 08/20/10 12:00 --end 08/20/10 13:00
```

Кроме числового значения времени можно использовать следующие специальные сокращения: now (сейчас), recent (десять минут назад), today (начиная с полуночи), yesterday (вчерашние сутки), this-week (неделя), this-month (месяц) или this-year (год). Вывод команды разбит на несколько столбцов, которые имеют следующие значения (слева направо):

1. Просто числовой индекс;
2. Дата возникновения события;
3. Время возникновения события;
4. Имя файла;
5. Номер системного вызова (чтобы увидеть его имя, используй флаг '-i');
6. Успешность системного вызова (yes или no);
7. Имя процесса, вызвавшего событие;
8. Audit UID (AUID). О нем читай ниже;
9. Номер события.

Вывод этой команды также можно существенно сократить, если указать флаг '--summary', который заставляет aureport выводить не все случаи доступа к файлом, а только их общее количество по отношению к каждому из файлов:

```
$ sudo aureport -f -i --start recent --summary
```

Вывод команды будет разбит на две колонки, первая из которых отражает количество попыток доступа к файлу (успешных или нет), а вторая – его имя. Просматривая суммарный отчет использования файлов и заметив подозрительную попытку доступа к одному из системных/скрытых/личных файлов, можно вновь вызвать aureport, чтобы найти процесс, который произвел эту попытку:

```
$ sudo aureport -f -i --start today | grep /etc/passwd
```

Получив список всех попыток доступа и номера событий, каждое из них можно проанализировать индивидуально с помощью утилиты ausearch:

```
$ sudo auserch -a номер_события
```

Также ausearch можно использовать для поиска событий по именам системных вызовов:

```
$ sudo ausearch -sc ptrace -i
```

Идентификаторам пользователей:

```
$ sudo ausearch -ui 2010
```

Именам исполняемых файлов:

```
$ sudo ausearch -x /usr/bin/nmap
```

Имени терминала:

```
$ sudo ausearch -tm pts/0
```

Именам демонов:

```
$ sudo ausearch -tm cron
```

Или ключам поиска:

```
$ sudo auserch -k etc_access
```

Вывод ausearch также может быть сокращен с помощью использования временных промежутков, наподобие тех, что мы использовали при вызове aureport. Сам aureport позволяет генерировать отчеты не только по использованию файлов, но и многих других типов событий, как, например, системные вызовы (флаг '-s'), попытки аутентификации (флаг '-au') , успешные логины (флаг '-l'), модификации аккаунта (флаг '-m') и многих других (полный список смотри в man-странице). Отчеты можно получать только для событий, завершившихся неудачно (флаг '--failed').

AUID и PAM

Во время своей работы в системе пользователь может использовать команды su или sudo для изменения своего системного идентификатора (UID), из-за чего процесс отслеживания его деятельности существенно усложняется. Чтобы обойти эту проблему система аудита использует так называемые Audit UID (AUID), которые закрепляются за каждым пользователем во время его входа в систему и остаются неизменными даже несмотря на смену пользователем своего UID с помощью su или sudo. Однако, по умолчанию функция присваивания AUID отключена (именно поэтому восьмой столбец вывода aureport всегда содержит значение -1), и, чтобы ее активировать, необходимо отредактировать некоторые конфигурационные файлы PAM. Для этого открой файл /etc/pam.d/login и добавь строку «session required pam_loginuid.so» перед строкой «session include common-session». Таким же образом измени конфигурационные файлы /etc/pam.d/sshd, /etc/pam.d/gdm (kdm, если ты используешь среду KDE), /etc/pam.d/crond и /etc/pam.d/atd.

Выводы

Система аудита – достаточно низкоуровневый компонент Linux, который требует глубоких знаний ОС для своей настройки и анализа журнальных файлов. Однако, разобравшись в нем, ты получишь мощнейший инструмент слежения за системой, который поможет обнаружить аномалии в поведении системы и найти их виновника.

WARNING

Чтобы изменения, внесенные тобой в конфигурационные файлы демона auditd, вступили в силу, необходимо перезагрузить демон с помощью команды «/etc/init.d/auditd restart».

INFO

На нашем диске ты найдешь пример рабочего файла правил auditd, который сможешь использовать на своем сервере под управлением Debian/Ubuntu.

WWW

Steve Grubb из компании Red Hat написал два небольших скрипта для визуализации отчетов системы аудита:

- http://people.redhat.com/sgrubb/audit/visualize/mkgraph
- http://people.redhat.com/sgrubb/audit/visualize/mkbar