

Аудит системных событий в Linux

Блог компании Selectel

Tutorial



Одним из инструментов, позволяющих повысить уровень безопасности в Linux, является подсистема аудита. С её помощью можно получить подробную информацию обо всех системных событиях.

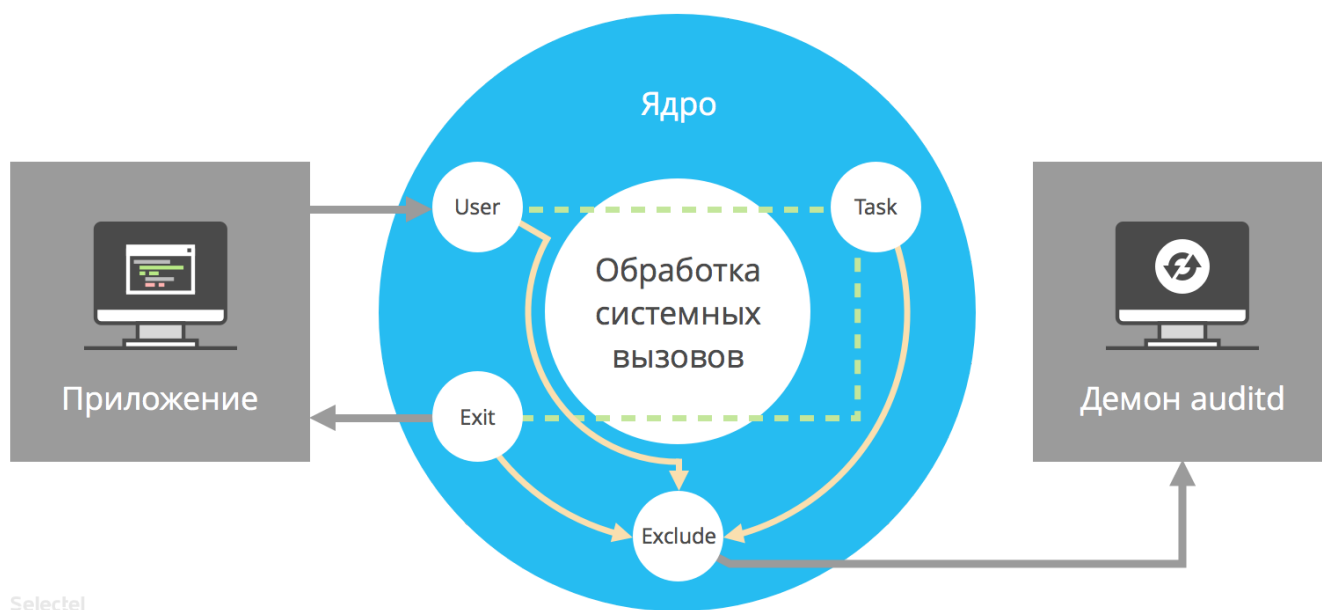
Она не обеспечивает никакой дополнительной защиты, но предоставляет подробную информацию о нарушениях безопасности, на основании которой можно принять конкретные меры. Особенности работы с подсистемой аудита мы рассмотрим в этой статье.

Подсистема аудита: архитектура и принцип работы

Подсистема аудита была добавлена в ядро Linux начиная с версии 2.6. Она предназначена для отслеживания критичных с точки зрения безопасности системных событий. В качестве примеров таких событий можно привести следующие (список далеко не полный):

- запуск и завершение работы системы;
- чтение, запись и изменение прав доступа к файлам;
- инициация сетевых соединений;
- попытки неудачной авторизации в системе;

- Ни одно из названных событий не может произойти без использования системных вызовов ядра. Чтобы их отслеживать, достаточно просто перехватывать соответствующие системные вызовы. Именно это и делает подсистема аудита:



Получив вызов от приложения в пространстве пользователя, подсистема аудита пропускает его через один из следующих фильтров: `user`, `task` или `exit` (более подробно о них речь пойдёт ниже). После этого вызов пропускается через фильтр `exclude`, который исходя из правил аудита передаёт его демону `auditd` для дальнейшей обработки.

Такая простая схема позволяет вполне эффективно отслеживать любой аспект работы ОС, а в случае компрометации системы выявлять подозрительные действия и определять их причину.

Установка

Чтобы начать работать с подсистемой аудита, нужно установить пакет auditd (здесь и далее приводятся примеры команд для ОС Ubuntu 14.04):

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

В состав этого пакета входят демон `auditd` и несколько вспомогательных утилит:

- `auditctl` — утилита для управления демоном `auditd`; позволяет получать информацию о текущем состоянии подсистемы аудита, а также добавлять и удалять правила;
- `autrace` — утилита для аудита событий, порождаемых процессами (работает по тому же принципу, что и `strace`);
- `ausearch` — утилита для поиска событий в журнальных файлах;
- `aureport` — утилита для генерации отчётов о работе системы аудита.

Конфигурирование

Настройки подсистемы аудита хранятся в конфигурационном файле `etc/audit/auditd.conf`. Он содержит в числе прочих следующие параметры:

- `log_file` — файл, в котором будут храниться логи подсистемы аудита;
- `log_format` — формат, в котором будет сохранены логи;
- `freq` — максимальное число записей протокола, которые могут храниться в буфере;
- `flush` — режим синхронизации буфера с диском (`none` — ничего не делать, `incremental` — переносить данные из буфера на диск с частотой, указанной в значении параметра `freq`; `data` — синхронизировать немедленно, `sync` — синхронизировать как данные, так и метаданные файла при записи на диск);
- `max_log_file` — максимальный размер файла лога в мегабайтах;
- `max_log_file_action` — действие при превышении максимального размера файла лога;
- `space_left` — минимум свободного пространства в мегабайтах, по достижении которого должно быть осуществлено действие, указанное в следующем параметре;
- `space_left_admin` — указывает, что делать, когда на диске недостаточно свободного места (`ignore` — ничего не делать; `syslog` — отправлять в `syslog`, `email` — отправлять уведомление по почте; `suspend` — прекратить запись логов на диск; `single` — перейти в однопользовательский режим; `halt` — выключить машину);
- `disk_full_action` — действие, которое нужно осуществить при переполнении диска (этот параметр может принимать те же значения, что и `space_left_admin`).

Создание правил

Для добавления и настройки правил используется команда `auditctl`. Вот список её опций:

- `-l` — вывести список имеющихся правил;
- `-a` — добавить новое правило;
- `-d` — удалить правило из списка;
- `-D` — удалить все имеющиеся правила.

Чтобы создать новое правило, нужно выполнить команду вида:

```
$ auditctl -a <список>, <действие> -S <имя системного вызова> -F <фильтры>
```

Сначала после опции `-a` указывается список, в который нужно добавить правило. Всего существует 5 таких списков:

- `task` — события, связанные с созданием новых процессов;
- `entry` — события, которые имеют место при входе в системный вызов;
- `exit` — события, которые имеют место при выходе из системного вызова;
- `user` — события, использующие параметры пользовательского пространства;
- `exclude` — используется для исключения событий.

Затем указывается, что нужно делать после наступления события. Здесь возможны два варианта: `always` (события будут записываться в журнал) и `never` (не будут).

После опции `-S` идёт имя системного вызова, при котором событие нужно перехватить (`open`, `close` и т.п.).

После опции `-F` указываются дополнительные параметры фильтрации. Например, если нам требуется вести аудит обращений к файлам из каталога `/etc`, правило будет выглядеть так:

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

Можно установить и дополнительный фильтр:

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

Аббревиатура `aw` означает следующее: `a` — изменение атрибута (`attribute change`), `w` — запись (`write`). Формулировка `perm = aw` указывает, что для директории `/etc` нужно отслеживать все факты изменения атрибутов (`a` — `attribute change`) и `w` (`w` — `write`).

При настройке слежения за отдельными файлами можно опустить опцию `-S`, например:

```
$ auditctl -a exit,always -F path =/etc/ -F perm = aw
```

Файлы правил

Правила можно не только задавать через командную строку, но и прописывать в файле `etc/audit/audit.rules`.

Начинается этот файл с так называемых метаправил, в которых задаются общие настройки журналирования:

```
# удаляем все ранее созданные правила
-D
```

```
# задаём количество буферов, в которых будут храниться сообщения
-b 320
```

```
# указываем, что делать в критической ситуации (например, при переполнении
0 - ничего не делать; 1 - отправлять сообщение в dmesg, 2 - отправлять я
-f 1
```

Далее следуют пользовательские правила. Их синтаксис предельно прост: достаточно просто перечислить соответствующие опции команды `auditctl`. Рассмотрим пример типового файла правил:

```
# отслеживать системные вызовы unlink () и rmdir()
-a exit,always -S unlink -S rmdir
```

```
# отслеживать системные вызовы open () от пользователя с UID 1001
-a exit,always -S open -F loginuid=1001
```

```
# отслеживать доступ к файлам паролей и групп и попытки их изменения:
-w /etc/group -p wa
-w /etc/passwd -p wa
-w /etc/shadow -p wa
-w /etc/sudoers -p wa
```

```
# отслеживать доступ к следующей директории:
```

```
# Отслеживать доступ к следующей директории.
```

```
-w /etc/my_directory -p r
```

```
# закрыть доступ к конфигурационному файлу для предотвращения изменений  
-e 2
```

Изменения конфигурации вступят в силу после перезапуска демона auditd:

```
$ sudo service auditd restart
```

Анализ журнальных файлов: утилита aureport

Все журнальные файлы сохраняются в директории /var/log/audit в машиночитаемом формате. Их можно сделать человекопонятными с помощью утилиты aureport.

Если ввести команду aureport без аргументов, мы увидим общую системную статистику (количество пользователей системы, общее количество системных вызовов, число открытых терминалов и т.п.):

```
$ sudo aureport
```

Summary Report

```
=====
```

```
Range of time in logs: 07/31/2015 14:04:23.870 - 08/04/2015 09:37:13.200
```

```
Selected time for report: 07/31/2015 14:04:23 - 08/04/2015 09:37:13.200
```

```
Number of changes in configuration: 0
```

```
Number of changes to accounts, groups, or roles: 3
```

```
Number of logins: 0
```

```
Number of failed logins: 0
```

```
Number of authentications: 0
```

```
Number of failed authentications: 61205
```

```
Number of users: 2
```

```
Number of terminals: 5
```

```
Number of host names: 73
```

```
Number of executables: 6
```

```
Number of files: 0
```

```
Number of AVC's: 0
```

```
Number of MAC events: 0
```

```
Number of failed syscalls: 0
```

```
Number of anomaly events: 0
```

```
Number of responses to anomaly events: 0
```

```
Number of crypto events: 0
```

```
Number of keys: 0
```

```
Number of process IDs: 17858
```

Number of events: 61870

Она не имеет особой практической ценности. Гораздо больший интерес представляют специализированные отчёты. Вот так, например, можно просмотреть информацию обо всех системных вызовах:

```
$ sudo aureport -s
```

Syscall Report

```
=====
# date time syscall pid comm auid event
=====
1. 08/03/2015 15:45:03 313 10285 modprobe -1 52501
2. 08/03/2015 15:45:03 313 10290 modprobe -1 52502
3. 08/03/2015 15:45:03 54 10296 iptables -1 52503
4. 08/03/2015 15:45:03 54 10302 iptables -1 52504
5. 08/03/2015 15:45:03 54 10305 iptables -1 52505
6. 08/03/2015 15:45:03 54 10313 iptables -1 52506
7. 08/03/2015 15:45:03 54 10325 iptables -1 52507
8. 08/03/2015 15:45:03 54 10329 iptables -1 52508
9. 08/03/2015 15:45:03 54 10343 iptables -1 52509
10.08/03/2015 15:45:03 54 10345 iptables -1 52510
11.08/03/2015 15:45:03 54 10349 iptables -1 52511
```

Воспользовавшись опцией -au (или --auth), можно просмотреть информацию обо всех попытках входа в систему:

```
$ sudo aureport -au
```

Authentication Report

```
=====
# date time acct host term exe success event
=====
1. 08/31/2015 18:00:19 ubnt static-166-6-249-80.stalowa.pilicka.pl ssh /u
2. 08/31/2015 18:01:38 root 59.63.188.31 ssh /usr/sbin/sshd no 334
3. 08/31/2015 18:01:41 root 59.63.188.31 ssh /usr/sbin/sshd no 335
4. 08/31/2015 18:01:45 root 59.63.188.31 ssh /usr/sbin/sshd no 336
5. 08/31/2015 18:01:53 root 59.63.188.31 ssh /usr/sbin/sshd no 337
6. 08/31/2015 18:01:57 root 59.63.188.31 ssh /usr/sbin/sshd no 338
7. 08/31/2015 18:01:59 root 59.63.188.31 ssh /usr/sbin/sshd no 339
```

В aureport поддерживается фильтрация по дате и времени:

```
$ sudo aureport -s --start 07/31/15 12:00 --end 07/31/15 13:00
```

Можно указывать как конкретные время и дату, так и специальные человекопонятные конструкции:

- now — текущий момент;
- yesterday — вчерашнее сутки;
- recent — 10 минут назад;
- this-week (или this-month, this-year) — текущая неделя (месяц, год).

С помощью aureport можно просмотреть информацию о действиях любого пользователя системы. Для этого нужно сначала узнать id этого пользователя:

```
$ id user
uid=1000(user) gid=1000(andrei) groups=1000(andrei),27(sudo)
```

и затем выполнить следующую команду:

```
$ sudo ausearch -ui 1000 --interpret
```

Ausearch: поиск и анализ событий

Для просмотра детальной информации о событии используется утилита ausearch:

```
$ sudo ausearch -a <номер события>
```

Вывод приведённой выше команды выглядит так:

```
type=SYSCALL msg=audit(1364481363.243:24287): arch=c000003e syscall=2 success=no
exit=-13 a0=7fffd19c5592 a1=0 a2=7fffd19c4b50 a3=a items=1 ppid=2686 pid=3538
auid=500 uid=500 gid=500 euid=500 suid=500 fsuid=500 egid=500 sgid=500 fsgid=500
tty=pts0 ses=1 comm="cat" exe="/bin/cat"
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key="sshd_config"
```

Рассмотрим его структуру более подробно. В поле type указывается тип записи; type = syscall означает, что запись была сделана после выполнения системного вызова. В поле msg указано время события в формате Unix Timestamp и его уникальный идентификационный номер.

В поле `arch` содержится информация об используемой архитектуре системы (`c000003e` означает `x86_64`), представленная в закодированном шестнадцатеричном формате. Чтобы она выводилась в человекочитаемом виде, можно воспользоваться опцией `-i` или `--interpret`.

В поле `syscall` указан тип системного вызова — в нашем случае это 2, то есть вызов `open`. Параметр `success` сообщает, был ли вызов обработан успешно или нет. В нашем примере вызов был обработан неудачно (`success = no`).

Для каждого вызова в отчёте также перечисляются индивидуальные параметры; более подробно о них можно почитать в [официальном руководстве](#). Вывести на консоль информацию о любом параметре в человекочитаемой форме можно получить при помощи упомянутой выше опции `-i` или `--interpret`, например:

```
$ sudo ausearch --interpret --exit -13
```

Опция `-sc` позволяет включать в список события, относящиеся к указанному системному вызову, например:

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

Опция `-ui` служит для поиска событий по идентификатору пользователя:

```
$ ausearch -ui 33
```

Поиск по именам демонов осуществляется с помощью опции `-tm`:

```
$ ausearch -x -tm cron
```

Для поиска нужных событий можно также использовать ключи, например:

```
$ sudo auditctl -k root-actions
```

Приведённая команда выведет список всех действий, совершённых от имени `root`-пользователя. Поддерживается также фильтрация по дате и времени, аналогичная той, что была описана выше. Вывести список событий, завершившихся неудачно, можно с помощью опции `--failed`.

Анализ процессов с помощью утилиты `autrace`

В некоторых случаях бывает полезным получить информацию о событиях, связанных с одним конкретным процессом. Для этой цели можно воспользоваться утилитой `autrace`. Предположим, нам нужно отследить процесс `date` и узнать, какие системные вызовы

и файлы он использует. Выполним команду:

```
$ sudo autrace /bin/date
```

На консоли появится следующий текст:

```
Waiting to execute: /bin/date
Mon Aug 31 17:06:32 MSK 2015
Cleaning up...
Trace complete. You can locate the records with 'ausearch -i -p 29234'
```

Обратим внимание на последнюю строку вывода: в ней указана команда, с помощью которой можно получить более подробную информацию. Выполним эту команду и передадим вывод утилите aureport, которая преобразует его в человекочитаемый формат:

```
$ sudo ausearch -p 29215 --raw | aureport -f -i
```

В результате мы получим вот такой отчёт:

File Report

```
=====
# date time file syscall success exe auid event
=====
1. 08/31/2015 16:52:16 /bin/date execve yes /bin/date root 25
2. 08/31/2015 16:52:16 /etc/ld.so.nohwcap access no /bin/date root 27
3. 08/31/2015 16:52:16 /etc/ld.so.preload access no /bin/date root 29
4. 08/31/2015 16:52:16 /etc/ld.so.cache open yes /bin/date root 30
5. 08/31/2015 16:52:16 /etc/ld.so.nohwcap access no /bin/date root 34
6. 08/31/2015 16:52:16 /lib/x86_64-linux-gnu/libc.so.6 open yes /bin/date
7. 08/31/2015 16:52:16 /usr/lib/locale/locale-archive open yes /bin/date
8. 08/31/2015 16:52:16 /etc/localtime open yes /bin/date root 56
```

Централизованное хранение логов

Для отправки логов подсистемы аудита в централизованное хранилище используется плагин audisp-remote. Он входит в пакет audisp-plugins, который нужно устанавливать дополнительно:

```
$ sudo apt-get install audisp-plugins
```

Конфигурационные файлы всех плагинов хранятся в директории /etc/audisp/plugins.d.

Настройки удалённого логгирования прописываются в конфигурационном файле `/etc/audit/plugins.d/auditd-remote.conf`. По умолчанию этот файл выглядит так:

```
active = no
direction = out
path = /sbin/auditd-remote
type = always
#args =
format = string
```

Чтобы активировать отправку логов в удалённое хранилище, заменим значение параметра `active` на `yes`. Затем откроем файл `/etc/audit/auditd-remote.conf` и в качестве значения параметра `remote_server` укажем буквенный или IP-адрес сервера, на котором будут храниться логи.

Чтобы принимать логи с удалённых хостов и сохранять их на сервере, в файле `/etc/audit/auditd.conf` нужно прописать следующие параметры:

```
tcp_listen_port = 60
tcp_listen_queue = 5
tcp_max_per_addr = 1
##tcp_client_ports = 1024-65535 #optional
tcp_client_max_idle = 0
```

Заключение

В этой статье мы изложили основы работы с подсистемой аудита Linux. Мы рассмотрели принцип работы системы аудита, научились формулировать правила, читать логи и пользоваться вспомогательными утилитами.

Для желающих изучить тему более подробно приводим несколько полезных ссылок:

- [полный перечень типов записей аудита](#);
- [полный перечень возможных событий](#);
- [статья об использовании возможностей аудита в расследовании атак на ядро](#).