Перед тем как продолжить озвучу очень важное правило при администрировании серверов, которое нужно заучить как рефлекс - без необходимости не работать с повышенными привилегиями ( в **linux** под пользователем **root**).

Утилита **sudo** занимается не ограничением доступа, а наоборот - позволяет запускать команды и программы с правами суперпользователя (**root**).

А вот какие команды и программы можно запускать конкретному пользователю как раз и настраивается через конфигурацию sudo.

В **ubuntu**, к слову **sudo** всегда установлено по умолчанию, а пароля суперпользователя мы вообще не знаем. Тот пользователь, который создавался при установке может выполнять все что угодно через **sudo**.

Если у тебя другой дистрибутив, то возможно sudo придется установить.

В **Debian**

apt-get install sudo

В **CentOS**

yum install sudo

В **Gentoo**

emerge sudo

С конфигурационного файла - **/etc/sudoers**

Изначально все настройки писались в нем. Но в какой-то момент разработчики добавили инструкцию **include** и стало возможно разбивать конфигурацию на несколько файлов.

Поэтому сейчас общая конфигурация обычно хранится в **/etc/sudoers**, а отдельные логические блоки хранятся в **/etc/sudoers.d**

Я под каждого пользователя завожу отдельный файл в **/etc/sudoers.d**. Но никто тебя не ограничивает. Ты можешь также выделять группы или что-то еще.

Открой файл **/etc/sudoers** и рассмотри его.

Как видишь там два типа записей.

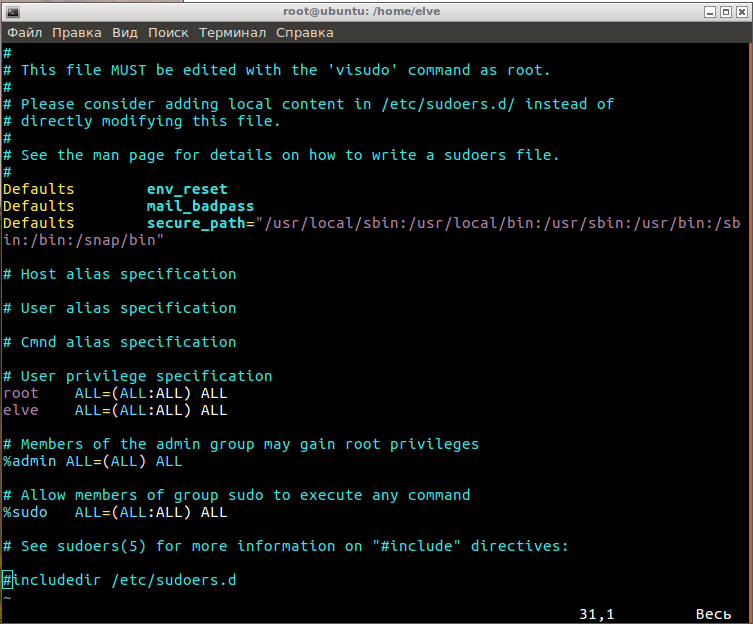
Те что начинаются с **Defaults** это настройки по-умолчанию. Они общие для всех пользователей.

А дальше идут записи вида

*пользователь и какие-то параметры*

*группа и какие-то параметры*

Тут описываются персональные разрешения. Эти разрешения можно выдавать за раз в одной строке, а можно разбить на несколько. Они все будут учтены.



Параметров, которые можно настроить довольно много. В этом уроке мы рассмотрим часть из них, чтобы ты увидел и мощь и гибкость настройки sudo.

Сначала немного о настройках безопасности по умолчанию.

***Defaults secure\_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin"***

Тут мы переопределяем системную переменную **PATH** для сеанса **sudo**.

В этой переменной задаются пути, по которым командный интерпретатор будет искать команду, которую ты ввел.

Есть возможность добавить в **PATH** своего сеанса отдельную папку, где будут программы одноименные со стандартными, но выполняющие другие действия. Тем самым сделать что-то неразрешенное так, чтобы в логах ничего подозрительного не было видно.

Так что задать все пути жестко идея хорошая.

***Defaults requiretty***

Тут сам параметр уже говорящий. Включаем необходимость реального сеанса через **tty**. Этот параметр отсекает запуск через **sudo** скриптов и приложений **CGI** (а это к примеру скрипты с веб-сайта).

Но при этом также не дает запускать через **sudo** команды прописанные в **cron**.

Тут каждый сам решает что ему важнее. Но на веб-сервера я бы эту строку прописал от греха.

***Defaults use\_pty***

Так уж повелось еще со времен мейнфреймов, что для консольного сеанса в **Linux** требуется устройство ввода - телетайп или **tty**. Сейчас **tty** как отдельного устройства уже нет, но в **Linux** они эмулируются с помощью специальных утилит. Во всех современных дистрибутивах их как минимум 12 и некоторые имеют свое определенное предназначение. К примеру на 8-м обычно висит графическая оболочка.

Но я сейчас о другом. Т.к. **tty** у нас ограниченное количество, то для того, чтобы открывать множество консольных и не только сеансов (к примеру через **ssh** или консолька в окне графической оболочки) был сделан виртуальный **tty**. **Pseudo tty** или **pty**.

При работе с tty есть возможность запустить процесс, который останется жить после завершения сеанса работы с tty (после разлогина пользователя). А **pty** при завершении сеанса завершает свое существование вместе со всем запущенным.

Так вот если мы включаем **use\_pty**, то все что запущено через **sudo** будет запускаться только на псевдотерминале.

***Defaults timestamp\_timeout=10***

Для выполнения команд через **sudo** обычно пользователь должен ввести свой пароль. Это может раздражать. Есть возможность создавать, т.н. сеанс пользователя **sudo**. Выполнив команду через **sudo** с вводом пароля мы открываем сеанс. И все время жизни сеанса повторно пароль больше не запрашвается.

Время жизни сеанса задается администратором сервера через конфиг **sudo**. Пользователь может повлиять на него только в сторону уменьшения. Командой **sudo -k**

можно этот сеанс закончить раньше времени. После этой команды пароль снова будет запрошен. Опять один раз за временной промежуток.

А этот временной промежуток задается командой выше в минутах. Т.е. 10 там означает 10 минут.

На первый взгляд это уход от безопасности в сторону удобства. Но может быть так, что более редкий ввод пароля снижает вероятность, что его подсмотрят из-за спины.

Кстати можно еще и следить за тем, кто что делает через **sudo**. Прямо как "большой брат" следит за гражданами =).

Для этого есть два параметра:

***Defaults log\_host, log\_year, logfile="/var/log/sudo.log"***

По умолчанию этот лог пишется в **syslog**. Так что мы просто аккумулируем записи в отдельный файл для удобства.

***Defaults log\_input, log\_output***

Этот параметр приказывает **sudo** записывать текст сеанса работы пользователя.

Этот лог записывается в папку **/var/log/sudo-io**

Там есть и лог команд и сообщения стандартных каналов ввода-вывода (**stdin, stderr, stdout**) и лог с **tty/pty**.

Теперь будет немного гибкости.

Можно переопределить сообщение, которое выдается при неверном вводе пароля.

***Defaults badpass\_message="Ах ты паскудник! Пароль забыл? Али и не знал никогда?"***

Ругаться будет после каждого неверного ввода.

Можно еще заставить **sudo** ругаться самостоятельно (естественно в рамках, заложенных разработчиком). Чтобы это включить надо добавить следующую строку в конфиг:

***Defaults insults***

Сразу предупреждаю. Она отменяет **badpass\_message**.

Ну и также можно переопределить текст "приветствия" **sudo**. Ты же обращал внимание, что в первый раз при запуске sudo выдается большой текст, рассказывающий о том, что надо пароль ввести. Вот это оно и есть.

Можно переопределить частоту его появления (**always,never,once**):

***Defaults lecture="always"***

А также текст

***Defaults lecture\_file="/path/to/file"***

Можно перейти к главному - управление разрешениями для пользователей и групп.

Для этого примера надо будет создать несколько пользователей. Будем раздавать им отдельные привилегии. Но это позже - сначала немного теории.

Общая схема написания правил такова:

**users hosts = (runas) commands**

**users** - имя пользователя или алиас. Тут еще можно использовать линуксовые группы. Чтобы программулина поняла, что это группа - ее отмечают символом % в начале названия.

**hosts** - имя компьютера, к которому относится правило. Можно алиас. Можно текстовое имя (то что выдает команда hostname), либо **ip** адрес или подсеть.

**runas** - от кого можно запускать команду

**tag** - опциональное поле. Набор тегов определен статически. Они действуют на поведение программы. К примеру **NOPASSWD**: разрешает выполнять команду без ввода пароля

**commands** - команда или алиас. Можно перечислять команды через запятую. Можно указывать команду с флагами - разрешение будет по четкому совпадению текста в консоли и того, что в конфиге sudo.

ПРИМЕР:

**test1 ALL = (test2) NOPASSWD: /usr/bin/mc**

Тут мы пользователю **test1** на любом компьютере (ALL) разрешаем запустить **mc** от имени пользователя **test2** без ввода пароля ( тег **NOPASSWD:**)

**test2 192.168.88.166=(ALL) NOPASSWD: /sbin/shutdown,/sbin/halt**

Тут мы разрешаем пользователю **test2** завершать работу компьютера на узле **192.168.88.166**.

У тебя может возникнуть вопрос - а как же мы у себя разрешаем и на узле **192.168.88.166** сможем что-то сделать? Нет. Не сможем.

Этот конфиг должен лежать на узле **192.168.88.166**. Пользователь **test2** на этом узле сможет перезагружать и выключать компьютер через sudo и без пароля.

Почему так? Давным давно, когда системы для массового управления конфигурациями еще даже не то что написаны, а даже придуманы не были надо было как-то поддерживать порядок на серверах. И разработчики sudo рассудили так: если мы сделаем один файл **sudoers** и раскидаем на все сервера разом, то это будет оптимально. Не надо следить за изменениями кучи конфигов - можно менять один и разливать по всем серверам. Эдакая политика безопасности.

А если еще использовать сквозную авторизацию с помощью **NIS** так вообще одно сплошное удобство.

Ну и раз уж я завел разговор про упрощение управления политикой безопаности в рамках sudo, то есть тут и еще кой какие удобства - алиасы.

**User\_Alias**

Здесь можно указать имя пользователя, либо перечислить имена пользователей через запятую.

К примеру:

**User\_Alias ADMIN = bobr, vitaly, eduard**

**User\_Alias WEBMASTER = evgen**

**User\_Alias BUH=alla, nata, birukova\_s**

Имя алиаса не обязательно с большой буквы писать. Просто так удобнее выделять его в тексте.

**Cmnd\_Alias**

Вот уж где поле для веселья. Синтаксис тот же. Но думаю стоит заострить внимание на том как записывать команды.

Во-первых необходимо прописывать полный путь до команды. Узнать можно с помощью утилиты **which**

Во-вторых важна степень детализации. Можно просто прописать команду и тогда она будет доступна без ограничений. А можно прописать ее с определенными параметрами и тогда доступ будет только до команды именно с этими параметрами. С другой стороны можно указать путь к папке и тогда все исполняемые файлы в ней будут доступны к запуску.

ПРИМЕР:

**Cmnd\_Alias POWEROFF = /sbin/halt, /sbin/reboot, /sbin/shutdown**

**Cmnd\_Alias FW\_SHOW = /sbin/iptables -L**

**Cmnd\_Alias SERVICES = /etc/init.d/**

**Host\_Alias**.

Тут указывается как имя компьютера, так и ip-адрес или подсеть. При этом, если маска не указана явно, то sudo ее додумает самостоятельно. Поэтому надо быть внимательным при заполнении. Особенно если есть подсети вида 10.5.4.0/24. **Sudo** автоматически подставит маску /8, что может открыть неплохую дырку в защите сервера.

Также надо иметь ввиду, если пишем в Host\_Alias имя, то оно должно указывать на внешний ip, а не на loopback. И локально оно должно резольвиться.

ПРИМЕР:

**Host\_Alias BUH = buh1, buh2, anjela**

**Host\_Alias LAN = 192.168.3.0/24**

Как видишь, хоть параметров не так много, но инструмент получился достаточно гибким.

И для закрепления предлагаю немного практики. Лучше конечно воссоздать всю конфигурацию по ТЗ но на крайний случай хотя бы заведи указанных пользователей на своем компьютере и испытай на них все настройки =).

Итак. Пусть будет такое ТЗ.

Подсеть в локалке 192.168.1.0/24

Имеющиеся компьютеры:

- **server** - единственный сервер в организации. 192.168.1.1

- **manager1** - компьютер менеджера Васи

- **manager2** - компьютер менеджера Пети

- **ingeneer** - компьютер сервис-инженера Артема

- **krevedko** - компьютер сисадмина Виталия

- **boss** - компьютер директора Геннадия

имена пользователей созвучны их именам.

И фантастическим образом у всех стоит Linux и все довольны и работают.

Нужно написать политику **sudo** чтобы соблюсти следующие условия:

1. Все имеют возможность выключать и перезагружать компьютеры

2. Все имеют возможность монтировать устройства (cdrom, флешка, переносной жесткий диск и т.д.)

3. Сисадмин может все.

4. Сервис-инженер в отсутствии сисадмина может перезагружать сервисы на сервере.

5. Босс бывший телефонист и любит иногда покрутить ip-телефонию.

Определям пользвателей в группы:

**User\_Alias USERS = vasya, petya, vitaly, artem, gena**

**User\_Alias ADMINS = ingeneer, krevedko**

Теперь выделяем группы команд, которые будем раздавать группам.

**Cmnd\_Alias POWEROFF = /sbin/shutdown,/sbin/reboot,/sbin/halt**

**Cmnd\_Alias PHONE = /usr/sbin/asterisk -r, /usr/sbin/service asterisk restart, cat /var/log/asterisk/full.log, tail /var/log/asterisk/full.log**

**Cmnd\_Alias SERVICES = /etc/init.d/**

**Cmnd\_Alias MOUNTS = /sbin/mount,/sbin/fdisk**

Сеть:

**Host\_Alias SERVER = 192.168.1.1**

**Host\_Alias LAN = 192.168.1.0/24**

И вот, когда все переменные нашей политики проинициализированы смотри как легко будет перенести задачу в конфигурационный файл.

**USERS LAN, !SERVER = (ALL) NOPASSWD: POWEROFF, MOUNTS**

**ADMINS LAN = (ALL) NOPASSWD: SERVICES**

**director SERVER = (ALL) NOPASSWD: PHONE**

**krevedko ALL = (ALL) ALL**

Вот и все. Конфигурация готова.