Меняем ssh-agent на gpg-agent

2017-12-13 23:28

#linux #ssh #gnupg #macosx

SSH Agent (SSH Агент) хранит в памяти компьютера приватные SSH-ключи. Когда программа `/usr/bin/ssh` (или другая подобная) пытается подключиться к серверу при помощи ключа, она сначала делает запрос к SSH Agent и просит у него приватный ключ, если у агента он есть, то ключ отдаётся программе и она его использует для подключения.

В linux и macos по умолчанию стоит `/usr/bin/ssh-agent` из openssh, однако он малофунциональный и сложный в настройке. Вместо него можно использовать агент из комплекта gnupg (`gpg-agent`), он умеет не только PGP-ключами управлять, но и выполнять функцию SSH Агента. У gpg-agent есть два преимущества: он настраивается, а также позволяет задавать пароли для шифрования ключей в памяти.

Включение в Linux (debian/ubuntu)¶

Сначала отключаем автоматический старт ssh-agent, для этого открываем редактором файл `/etc/X11/Xsession.options` и комментируем/удаляем строчку:

```
use-ssh-agent
```

Устанавливаем нужные пакеты:

```
$ sudo apt install gpg-agent pinentry-qt
```

Выбираем GUI pinentry (это диалоги, которые вы будете видеть при работе с ключами) вместо дефолтного консольного приложения:

```
$ sudo update-alternatives --set pinentry /usr/bin/pinentry-qt
```

gpg-agent стартует автоматически в X-сессии, дополнительно никакие скрипты писать для старта не нужно. Однако по умолчанию в GPG Agent поддержка SSH-ключей выключена, для включения нужно сделать так:

```
$ echo enable-ssh-support >> ~/.gnupg/gpg-agent.conf
```

Всё, теперь после рестарта X-сессии у вас будет использоваться gpg-agent вместо ssh-agent.

Включение в Macos¶

> Речь идёт о MacOS 10.12 (Sierra).

Для начала вы должны поставить GPG Suite, он устанавливает в `/usr/local/bin`все необходимые программы: gpg, gpg-agent, pinentry. Весь дальнейший текст подразумевает, что вы установили именно GPG Suite, а не gnupg из brew, например. В случае gnupg из brew нужно переписать пути бинарников в некоторых файлах (в gpg-agent.daemon.plist).

gpg-agent автоматически не стартует, поэтому нужно добавить в систему его автозапуск, я написал конфиг для launchd, вот команда, которая его скачивает в корректный каталог:

```
$ curl -L https://goo.gl/6aKUCN -o ~/Library/LaunchAgents/gpg-agent.daemon.plist
```

Теперь можно запустить:

```
$ launchctl load ~/Library/LaunchAgents/gpg-agent.daemon.plist
```

И убедиться, что действительно работает:

```
$ ps ax|grep gpg-agent
...
48479   ??  Ss     0:00.63 gpg-agent --homedir /Users/sigsergv/.gnupg --use-standard-socket --daemon
...
```

Поддержка SSH включается точно так же, как и для linux:

```
$ echo enable-ssh-support >> ~/.gnupg/gpg-agent.conf
```

И вот на этом простые шаги заканчиваются.

Чтобы агент работал для всех приложений, необходимо установить системную переменную окружения SSH_AUTH_SOCK, именно её используют программы для получения пути к сокету. Однако в macos эта переменная контролируется менеджером launchd из конфига запуска ssh-agent и глобально её изменить или удалить штатными средствами нельзя.

Если вы будете пользоваться программой ssh только из терминала, то переменную можно переопределить в стартовом скрипте `~/.profile`, `~/.bashrc`, `~/.zshrc` и так далее. Например, так:

```
$ echo 'export SSH_AUTH_SOCK="${HOME}/.gnupg/S.gpg-agent.ssh"' >> ~/.zshenv
```

После этого в терминале для всех операций с агентом будет использоваться gpg-agent.

Если вы хотите использовать gpg-agent глобально, то вам придётся полностью выключить оригинальный ssh-agent, так как именно в его конфиге запуска прописана мешающая нам переменная. Простым способом этого сделать нельзя, так как конфиг защищён от записи системой SIP (System Integrity Protection). Поэтому нужно SIP отключить, изменить конфиг, а потом снова SIP включить.

1. Отключение SIP¶

Процесс описывать в этой статье не буду, можно почитать, например, здесь http://osxh.ru/elcappitan/sip.

2. Выключение ssh-agent¶

На этом этапе SIP должен быть уже отключен, при этом система загружена в нормальном режиме, а не в режиме восстановления. Выключаем (сообщения об ошибках игнорируем, если они не содержат упоминания SIP):

```
$ launchctl unload -w  /System/Library/LaunchAgents/com.openssh.ssh-agent.plist
$ sudo launchctl unload -w  /System/Library/LaunchAgents/com.openssh.ssh-agent.plist
$ sudo mv /System/Library/LaunchAgents/com.openssh.ssh-agent.plist /System/Library/LaunchAgents/com.open
```

Также убедитесь, что у вас нигде в стартовых скриптах не прописана переменная SSH_AUTH_SOCK, иначе вы не сможете понять, работает описываемый метод или нет.

3. Включение SIP¶

Снова читаем http://osxh.ru/elcappitan/sip, выполняем команду, перезагружаемся.

4. Установка переменной окружения¶

Выше я показывал, как добавить выставление переменной окружения для терминальных программ. Для GUI программ этот метод не работает, так как они не читают `~/.bashrc` и прочие подобные скрипты. Поэтому если вам нужно использовать ssh в GUI-приложениях, следуйте инструкции ниже:

Установка корректной переменной окружения производится также через конфигlaunchd:

```
$ curl -L https://goo.gl/deJ6eb -o ~/Library/LaunchAgents/gpg-agent.env.plist
```

Для гарантии перезагружаемся.

По умолчанию *GPG Suite* сохраняет пароль к расшифрованному ключу в системном Keychain, это можно отключить в настройках: *System Preferences → GPG Suite → Store in macOS Keychain*.

Особенности gpg-agent¶

Отличия gpg-agent от ssh-agent:

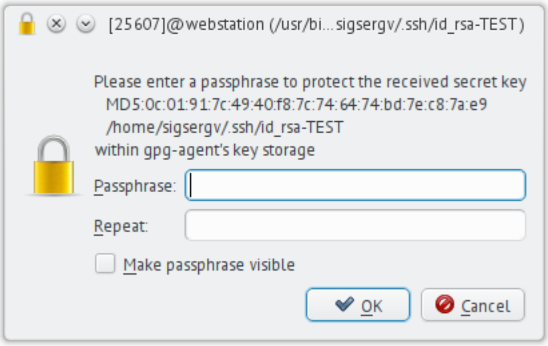
- можно выбирать, какие конкретно ключи хранить в агенте и они там будут храниться после перезагрузки;
- для каждого ключа можно отдельно задать время его кеширования и дополнительное подтверждение на использование;
- у gpg-agent есть дополнительный уровень защиты — пароль на дешифрованные ключи в памяти;
- сразу есть GUI для контроля доступа без необходимости колдовать с переменными окружения и сторонними пакетами.

Если у вас в `/etc/ssh/ssh_config` или `~/.ssh/config` не включена опция **AddKeysToAgent**, то ключи сами в агент не попадут. Идентификаторы одобренных для агента ключей сохраняются в `~/.gnupg/sshcontrol`. Их можно туда добавлять вручную или через `ssh-add`.

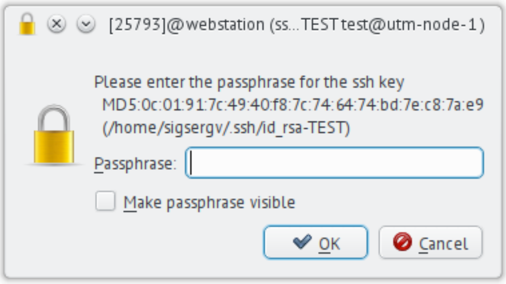
Например, если вы хотите для ключа `~/.ssh/id_rsa-TEST` использовать агент, то можно сделать так:

```
$ /usr/bin/ssh-add ~/.ssh/id_rsa-TEST
```

Сначала ssh спросит парольную фразу для ключа, после чего появится диалог, где нужно выбрать пароль для шифрования ключа в агенте, обычно это простая короткая фраза (НЕ пароль к оригинальному ключу!):



Этот пароль (а не оригинальную парольную фразу!) нужно будет ввести, когда сторонняя программа затребует ключ:



Идентификатор добавленного ключа сохраняется в файле `~/.gnupg/sshcontrol` в виде блока такого формата:

```
# RSA key added on: 2017-12-13 14:07:27
# Fingerprints:  MD5:0c:01:91:7c:49:40:f8:7c:74:64:74:bd:7e:c8:7a:e9
#                SHA256:hnOuTZzfe2ak10Jd6bl60kE9yAUqw34KinJ3H4opNbs
3BC01B6F0043256039006294F76C45139B703DBF 0
```

`~/.gnupg/sshcontrol` — это текстовый файл. Строки с # в начале считаются комментариями (то есть игнорируются системой). Все остальные строки должны иметь такой формат:

```
                ┌──────────────────────────── идентификатор ключа
                │                          ┌─ TTL кеширования в секундах
────────────────┴───────────────────────   ┴
3BC01B6F0043256039006294F76C45139B703DBF 0
```

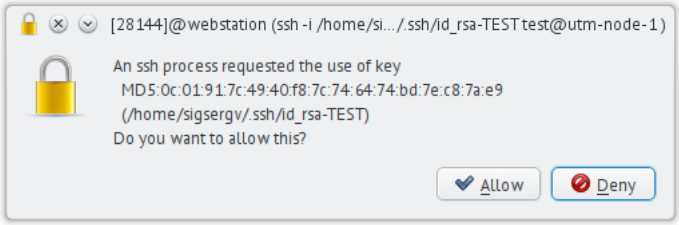
TTL кеширование — это время в секундах после расшифровки, в течение которого запрос ключа не потребует пароля. Если указать 0, будет использоваться TTL по умолчанию (он задаётся аргументом `default-cache-ttl-ssh` в конфиге `~/.gnupg/gpg-agent.conf`).

Перед идентификатором можно указать восклицательный знак (!) такой ключ будет выключен, это по сути то же самое, что и комментирование этой строчки через #.

Также можно для ключа указать набор флагов (третьим полем после TTL, оно опциональное):

```
                ┌────────────────────────────── идентификатор ключа
                │                          ┌─── TTL кеширования в секундах (0 — не кешировать)
                │                          │ ┌─ флаги (пока только confirm)
────────────────┴───────────────────────   ┴ ┴───
3BC01B6F0043256039006294F76C45139B703DBF 0 confirm
```

На данный момент поддерживается только один флаг — `confirm`, если он выставлен, то перед каждым обращением к ключу (даже если оно в пределах действующего TTL) будет показываться дополнительный диалог с подтверждением:



Базовые сценарии и операции¶

> Я очень советую не использовать автоматическое добавление ключей в агент. Для `/usr/bin/ssh` и других программ, использующих `/usr/bin/ssh` (это git, например), рекомендую посмотреть файлы `/etc/ssh/ssh_config` и `~/.ssh/config` удалить оттуда упоминания параметра **AddKeysToAgent**. Дальше все нужные ключи добавлять только вручную.

Программа `ssh-add` из комплекта `openssh-client` обращается к текущему агенту по стандартному протоколу, поэтому её можно использовать вместе с `gpg-agent`. Так что примеры ниже могут пригодиться и для ssh-agent тоже.

Также помните, что в некоторых системах менеджер паролей может запомнить пароль для расшифровки ключа, поэтому я настоятельно советую эту фичу отключить, не ставить галочку «Запомнить пароль», ну или как минимум самые критичные ключи добавлять только с флагом confirm, чтобы вы явно разрешали каждое использование ключа.

Как остановить/запустить/перезапустить агент¶

gpg-agent работает как демон, поэтому для его запуска или остановки нужно использовать специальные команды. Это демон текущей сессии, а не системы в целом.

Для линукса:

```
# остановить
# запустить
# перезапустить
```

Для макоси (подразумевается, что агент настроен по этому руководству):

```
# остановить
$ launchctl unload ~/Library/LaunchAgents/gpg-agent.daemon.plist

# запустить
$ launchctl load ~/Library/LaunchAgents/gpg-agent.daemon.plist

# перезапустить
$ launchctl unload ~/Library/LaunchAgents/gpg-agent.daemon.plist
$ launchctl load ~/Library/LaunchAgents/gpg-agent.daemon.plist
```

Если вы обновили *GPG Suite*, то нужно будет запустить агент снова, так как в процессе апгрейда, процесс агента прибивается.

Просмотр списка ключей в агенте¶

```
$ /usr/bin/ssh-add -l
2048 SHA256:hnOuTZzfe2ak10Jd6bl60kE9yAUqw34KinJ3H4opNbs /home/sigsergv/.ssh/id_rsa-TEST (RSA)
2048 SHA256:OV0fErCIpclIvjl8hCaFmdgfJmGYmdYNOhUus1i7GUg /home/sigsergv/.ssh/id_rsa-cc (RSA)
```

Добавление ключа в агент¶

```
$ /usr/bin/ssh-add ~/.ssh/id_rsa-TEST
Enter passphrase for /home/sigsergv/.ssh/id_rsa-TEST:
Identity added: /home/sigsergv/.ssh/id_rsa-TEST (/home/sigsergv/.ssh/id_rsa-TEST)
```

Добавление ключа с TTL=60 секунд¶

```
$ /usr/bin/ssh-add -t 60 ~/.ssh/id_rsa-TEST
Enter passphrase for /home/sigsergv/.ssh/id_rsa-TEST:
Identity added: /home/sigsergv/.ssh/id_rsa-TEST (/home/sigsergv/.ssh/id_rsa-TEST)
Lifetime set to 60 seconds
```

Данный вызов не модифицирует существующий ключ в `~/.gnupg/sshcontrol`! Имейте это в виду.

Добавление ключа с включенным подтверждением¶

```
$ /usr/bin/ssh-add -c ~/.ssh/id_rsa-TEST
Enter passphrase for /home/sigsergv/.ssh/id_rsa-TEST (will confirm each use):
Identity added: /home/sigsergv/.ssh/id_rsa-TEST (/home/sigsergv/.ssh/id_rsa-TEST)
The user must confirm each use of the key
```

Данный вызов не модифицирует существующий ключ в `~/.gnupg/sshcontrol`! Имейте это в виду.

Удаление ключа из агента¶

В ssh-add есть параметр -d, однако для gpg-agent он не сработает, то есть вы можете выполнить команду, но она ничего не сделает. Для полного удаления нужно пользоваться командами агента. К сожалению, нормальной процедуры или GUI для этого нет и нужно пользоваться несколькими консольными командами.

Сначала посмотрим список хранимых ключей:

```
$ ssh-add -l
2048 SHA256:hnOuTZzfe2ak10Jd6bl60kE9yAUqw34KinJ3H4opNbs /Users/sigsergv/.ssh/id_rsa-TEST (RSA)
2048 SHA256:MApXMQ8qUqBd4yfbA07RSh+VLT+ARsg5k8fbY72LVCc /Users/sigsergv/.ssh/id_rsa-XYZ (RSA)
2048 SHA256:8X/7LtWAFl962369KYW9ADgNCiPDUE7ilwI9QWXrBxk /Users/sigsergv/.ssh/id_rsa-SECURE (RSA)
```

Хотим удалить первый, для него ssh-add показывает SHA256 равный hnOuTZzfe2ak10Jd6bl60kE9yAUqw34KinJ3H4opNbs, открываем файл `~/.gnupg/sshcontrol` и находим его там:

```
# RSA key added on: 2017-12-23 21:04:21
# Fingerprints:  MD5:0c:01:91:7c:49:40:f8:7c:74:64:74:bd:7e:c8:7a:e9
#                SHA256:hnOuTZzfe2ak10Jd6bl60kE9yAUqw34KinJ3H4opNbs
3BC01B6F0043256039006294F76C45139B703DBF 600 confirm
```

Агент использует *keygrip* ключа для его идентификации, это тоже хеш-сумма, но другая. Для нашего ключа keygrip — это 3BC01B6F0043256039006294F76C45139B703DBF. Именно эту строку будем использовать для удаления (команда может несколько раз запросить подтверждение действия):

```
$ gpg-connect-agent 'DELETE_KEY 3BC01B6F0043256039006294F76C45139B703DBF' /bye
OK
```

Убедимся, что всё получилось:

```
$ ssh-add -l
2048 SHA256:MApXMQ8qUqBd4yfbA07RSh+VLT+ARsg5k8fbY72LVCc /Users/sigsergv/.ssh/id_rsa-TEST2 (RSA)
2048 SHA256:8X/7LtWAFl962369KYW9ADgNCiPDUE7ilwI9QWXrBxk /Users/sigsergv/.ssh/id_rsa-INSECURE (RSA)
```

Ключа в списке нет, всё нормально. И дальше вручную удалите упоминания ключа из `~/.gnupg/sshcontrol`.

Очистить закешированные пароли¶

Чтобы очистить закешированные пароли к ключам, можно перезагрузить агент такой командой:

```
$ gpg-connect-agent reloadagent /bye
```

Модификация параметров ключа в агенте¶

Откройте файл `~/.gnupg/sshcontrol` в любом текстовом редакторе и поправьте что нужно, описание формата я давал выше. Вы можете изменить TTL, для отдельных ключей, включить/выключить подтверждение, удалить ключ.

Модификация TTL по умолчанию¶

Откройте файл `~/.gnupg/gpg-agent.conf` и поменяйте значение параметра `default-cache-ttl` (в секундах).