SSH, аутентификация по ключам, ssh-keygen, ssh-agent (FreeBSD)

Опубликовано: 04.10.2014 | Автор: Николай

В нескольких предыдущих статьях данного раздела, мы более-менее раскрыли тему протокола **SSH**, настройку и использование *SSH* сервера и *SSH* клиента в операционной системе FreeBSD. В данной статье хотелось-бы рассказать об *SSH* аутентификации на основе пар ключей, заодно рассмотреть остальные программы из пакета программного обеспечения, протокола *SSH*, под FreeBSD.

Итак, нужно это в первую очередь для удобства, при удаленном администрировании серверов , не нужно вводить пароль при каждом подключении, и в определенной степени более безопасно, нежели подключаться к удаленной машине только по паролю.
Общий принцип для аутентификации на основе публичного ключа, в протоколе *SSH*, таков:

- С помощью программы **ssh-keygen**, должна быть сгенерирована пара ключей, публичный ключ (*public key*) и приватный ключ (*private key*)
- Секретный ключ, всегда остается у клиента и никому никогда не показывается.
- Публичный ключ копируется на удаленный *SSH* сервер (говорим опять-же в контексте операционной системы FreeBSD) и кладется в специальный файл, известный *SSH* серверу. По-умолчанию, для хранения публичных ключей, используется файл *~/.ssh/authorized_keys*. Файл для хранения ключей назначается в файле конфигурации *SSH* сервера, директивой **AuthorizedKeysFile**
- Клиент, отправляет *SSH* серверу свой публичный ключ и запрашивает аутентификацию по данному ключу.
- Сервер проверяет файл *~/.ssh/authorized_keys*, если такой ключ найден, *SSH*сервер отправляет клиенту сообщение, зашифрованное найденным публичным ключом пользователя.
- Клиент должен расшифровать сообщение с помощью своего приватного ключа, если приватный ключ защищен паролем (а так и должно быть всегда, в

целях безопасности), программа *ssh*, попросит пользователя ввести пароль, что-бы сначала расшифровать сам ключ.

- Если сообщение расшифровано, правильность публичного и приватного, ключей, считается подтвержденной и пользователю предоставляется доступ в систему.

Весь процесс аутентификации можно посмотреть, с помощью опции **-v** (*verbose*), программы *ssh*, очень полезная штука, особенно на стадии настройки серверной и клиентской частей протокола *SSH*.

Генерация ключей с помощью программы ssh-keygen.

Для создания и управления ключами, предназначена программа *ssh-keygen*, так-же входящая в пакет программного обеспечения *OpenSSH*. Полный список опций можно как всегда посмотреть командой *man ssh-keygen*. Здесь приведу лишь несколько из них:

-t type
ssh-keygen, работает с тремя типами ключей. Возможные значения:
RSA 1 — для протокола SSH версии 1.
RSA — для протокола SSH версии 2.
DSA — для протокола SSH версии 2.

-b
Длина ключа в битах.
RSA — минимальная длина, 768 бит, длина ключа по-умолчанию, 2048 бит.
DSA — длина 1024 бита.

-i
Данная опция используется для импорта ключей из одного формата (например ключи сгенерированные программой *PuTTYgen*, для Windows), в формат OpenSSH.

-l
Посмотреть отпечаток секретного ключа (fingerprint).

-p
Изменить секретную фразу приватного ключа.

Сгенерируем пару *RSA* ключей, это рекомендуемый формат, как наиболее устойчивый к взлому. По-умолчанию, ключи, сохраняются в домашнюю директорию пользователя, в файлы *~/.ssh/id_rsa* — приватный (секретный) ключ, и *~/.ssh/id_rsa.pub* — публичный ключ.

```
vds-admin /root# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Здесь предлагается ввести путь для сохранения файлов ключей. Имейте
в виду, если просто указать свое имя файла, например: mynew_id_rsa,
то ключи будут сохранены в корне домашней директории пользователя,
в моем случае в директории /root, с именами: mynew_id_rsa и
mynew_id_rsa.pub.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Здесь вас попросят ввести секретную фразу, настоятельно рекомендую
делать это всегда. )
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
56:79:b5:61:ea:19:70:13:a4:67:a2:af:15:11:db:b5 root@vds-admin.
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|E                |
| .               |
|  o o ..         |
| . B +..So       |
|  o =.. o .      |
| . o. .o         |
|  =..+o          |
| o.+*o           |
+-----------------+
```

Вот собственно и все, сгенерирована пара ключей *RSA*, с длиной 4096 бит и сохранены в файлы */root/.ssh/id_rsa* и */root/.ssh/id_rsa.pub*.

Настройка SSH сервера на аутентификацию по открытому ключу.

SSH сервер естественно должен быть настроен на аутентификацию по ключам, приведу кусок, касающийся аутентификации, своего файла конфигурации, *SSH* сервера. Все что закомментировано в файле конфигурации *SSH*, отсюда убрал, для простоты восприятия:

> PermitRootLogin yes
> Данная директива нужна, если вы планируете работать под учетной записью root.
> PubkeyAuthentication yes
> AuthorizedKeysFile .ssh/authorized_keys
> RhostsRSAAuthentication no
> HostbasedAuthentication no
> PermitEmptyPasswords no
> UseLogin no

Теперь копируем публичный ключ на удаленный *SSH* сервер:

```
vds-admin /root/.ssh# cat id_rsa.pub | ssh 192.168.50.50 "cat >>
~/.ssh/authorized_keys"
```

или так

```
cat ~/.ssh/id_rsa.pub | ssh user@machine "mkdir ~/.ssh; cat >>
~/.ssh/authorized_keys"
```

Так как для копирования вы подключаетесь к *SSH* серверу, он запросит пароль, авторизацию по ключам-то мы еще не настроили.) Я работал под учетной записью *root*, а без явного указания имени пользователя в командной строке или в конфигурационном файле *SSH* клиента, подключение происходит с именем текущего пользователя, то есть мне нужно было ввести пароль пользователя root, удаленной системы. После ввода пароля, публичный ключ будет добавлен в конец файла *~/.ssh/authorized_keys*, так как мы подключаемся пользователем root, то путь *~/.ssh/authorized_keys* указывает на директорию */root/.ssh/authorized_keys*.

Ключи сгенерированы и скопированы на *SSH* сервер, сервер соответствующим образом настроен, пробуем подключится:

```
vds-admin /root/.ssh# ssh 192.168.50.50
Enter passphrase for key '/root/.ssh/id_rsa':  Здесь вводим нашу
секретную фразу, указанную при генерации ключей.
```

Если пароль на ключ введен верно, получаем доступ в удаленную систему.

> Обратите внимание на следующий момент, с приведенным выше вариантом конфигурации *SSH* сервера, при неудачной аутентификации по ключам, например если неправильно ввести секретную фразу ключей, будет предложена аутентификация по паролю. Что-бы изменить это поведение и например вообще не пускать пользователя root иначе, как по ключам, можно изменить в конфигурационном файле сервера, значение директивы **PermitRootLogin** с *yes* на *without-password*.

Использование программы ssh-agent

Как было сказано выше, в целях безопасности, приватный ключ, всегда должен быть защищен секретной фразой (паролем), но это вызывает некоторые неудобства, вам придется вводить секретную фразу, каждый раз когда вы подключаетесь к удаленному серверу по ключу, было-бы гораздо проще ввести пароль на ключ один раз и пользоваться им сколько потребуется. На этот случай в пакете *OpenSSH*, существуют специальные программы *ssh-agent* и *ssh-add*, в общем-то вторая является дополнением первой.

Как это работает. Поле запуска программы *ssh-agent*, в нее добавляются расшифрованные ключи, то есть при добавлении она запросит секретную фразу ключа, для его дешифровки, и далее *ssh-agent*, будет выдавать уже расшифрованные ключи по запросу, например программе SSH.

Запускать *ssh-agent*, можно двумя способами, со специальной опцией, говорящей, какой тип оболочки используется, или с помощью команды *eval*. Принципиальной разницы как его запускать,

нет, просто в случае с опцией, вы должны точно знать, какую оболочку вы используете.

ssh-agent -c
Если в качестве оболочки используется C — Shell

ssh-agent -s
Если в качестве оболочки используется Bourne Shell

eval `ssh-agent`
В таком варианте запущенный ssh-agent, будет передан команде eval, которая выполнит его в текущей оболочке. Обратите внимание, используются **обратные кавычки** а не обычные !

Итак, запускаем *ssh-agent*:

```
vds-admin /root/.ssh# eval `ssh-agent`
Agent pid 1982
```

При запуске, *ssh-agent* создает переменные окружения, проверим какие:

```
vds-admin /root/.ssh# env | grep SSH_A
SSH_AUTH_SOCK=/tmp/ssh-7EeitdI5mr/agent.1981
В этой переменной хранится сокет, через который программа ssh,
будет связываться с ssh-agent.
SSH_AGENT_PID=1982
Это PID процесса ssh-agent
```

Теперь нужно поместить в него расшифрованные ключи, делается это с помощью программы *ssh-add*.

Если запустить ее без аргументов, будут добавлены все ключи, найденные в стандартных местах их обитания, то есть будут просканированы следующие файлы, *~/.ssh/identify*, *~/.ssh/id_rsa* и */.ssh/id_dsa*. Если ключ защищен парольной фразой, программа попросит ввести ее, что-бы расшифровать ключ и загрузить уже готовый к применению.

```
vds-admin /root/.ssh# ssh-add
Enter passphrase for /root/.ssh/id_rsa: Запрос пароля на
расшифровку
Identity added: /root/.ssh/id_rsa (/root/.ssh/id_rsa)
```

Теперь пробуем подключиться к удаленному SSH серверу:

```
vds-admin /root/.ssh# ssh 192.168.50.50
Last login: Tue Jul 7 18:45:27 2009 from .host.
Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
The Regents of the University of California. All rights reserved.

FreeBSD 7.1-STABLE (SMP_KERNEL) #1: Tue Mar 10 18:14:59 UTC 2009
Welcome to FreeBSD!
```

Как видите пароль у нас больше никто не спрашивает, программа *SSH*, получает уже расшифрованный ключ от *ssh-agent* и мы успешно подключаемся к удаленному *SSH* серверу.

Посмотреть отпечатки загруженных в *ssh-agent* ключей, можно той-же командой *ssh-add* с опцией **-l**, или целиком ключи, опцией **-L**.

```
vds-admin /root/.ssh# ssh-add -l
4096 56:79:b5:61:ea:19:70:13:a4:67:a2:af:15:11:db:b5
/root/.ssh/id_rsa (RSA)
vds-admin /root/.ssh# ssh-add -L
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAgEArL0hIMmhw8rXeg0p72+EJXnC4iAY2XTkPAdTb3L
nQb9bc0E5wvd
cwCdNEtLlDIDCH+z0I1FaP3TfpvgVkv59X15TaNIeoB7uydqXvlLMOxpOJkfbc3eiA6
a07PvZHMKXcIA0ZZ9+j12u
l+HsGOK2qMQ5g52mOc6BOF1PVuoHfTR1C9nExv5UCA6h7e/v2wxq79pMW07nx7nshB5
/1n5Gnyx+toQEzRiFbf
zOJBB1ry/9NUF1DiBwOhKJVdEJBTUi0hyh/e77UAmVtkguEtjrsDEdxJ31sV21SL97E
ZHymMjRPjwU2nWjRkHf0Pi7
dlXBoCKRj3dQps38kwFd3m9Tu4+hXSnsF8FdxkX5y9XmN8Uz8UWR6O2zslr7xZubkDR
3aCq1dtcbu2nkvC4+Vy
TOxEdnaNqDlC6U6G6aUVKFc0Rb5dcPnqpKqUHWE8MlXq/obKMRjuSz+GOr1VgRe/wZM
7/0GoO1Xrv2MDMhS+
S1uR+XkHkQr/EjTSxPiDZ92snZhtiyPIzTUZDOmclWHbe4gyvxDtU3Lxqzl3t1+Murg
4sN1NrkZIHefMq2xeCOS8P
bI89b3zJG2PJ3i2PSsOMviqIBOL3BBskGSWksJKi/YvvKwrlKaSM10wMZTbXHomgu+6
jRd7cZtUOmU/FO0IoKejB
MwuYbcPC+TCWBks0phU= /root/.ssh/id_rsa
```

Загружен один ключ, по которому мы подключались к удаленной машине. Кроме этого, при запуске *ssh-add*, можно указать путь до конкретного ключа, который необходимо загрузить, например:

```
vds-admin /root/.ssh# ssh-add /root/.ssh/id_rsa.new1
Enter passphrase for /root/.ssh/id_rsa.new1:
Identity added: /root/.ssh/id_rsa.new1 (/root/.ssh/id_rsa.new1)
Проверяем, что у нас теперь в ssh-agent:
vds-admin /root/.ssh# ssh-add -l
4096 56:79:b5:61:ea:19:70:13:a4:67:a2:af:15:11:db:b5
/root/.ssh/id_rsa (RSA)
2048 68:81:38:fe:66:e8:05:88:8b:49:80:d2:d1:8b:bf:99
/root/.ssh/id_rsa.new1 (RSA)
Загружено уже 2 ключа
```

Удаляются ключи из *ssh-agent*, так-же просто как и добавляются, для этого используется опция **-d**, без параметров, для удаления стандартных ключей, опция **-d** *файл_ключа*, если нужно удалить конкретный ключ, или опция **-D**, для удаления всех ключей, например:

```
vds-admin /root/.ssh# ssh-add -l
4096 56:79:b5:61:ea:19:70:13:a4:67:a2:af:15:11:db:b5 id_rsa (RSA)
2048 68:81:38:fe:66:e8:05:88:8b:49:80:d2:d1:8b:bf:99 id_rsa.new1
(RSA)
2048 c7:9f:b1:3b:c1:d0:61:15:38:27:d1:36:a7:49:55:cd id_rsa.new2
(RSA)

vds-admin /root/.ssh# ssh-add -d id_rsa.new2
Identity removed: id_rsa.new2 (id_rsa.new2.pub)

vds-admin /root/.ssh# ssh-add -l
4096 56:79:b5:61:ea:19:70:13:a4:67:a2:af:15:11:db:b5 id_rsa (RSA)
2048 68:81:38:fe:66:e8:05:88:8b:49:80:d2:d1:8b:bf:99 id_rsa.new1
(RSA)

vds-admin /root/.ssh# ssh-add -D
All identities removed.
vds-admin /root/.ssh# ssh-add -l
The agent has no identities.
```

Приведу список самых используемых опций программы *ssh-add*:

ssh-add

Без опций, добавляются стандартные ключи

ssh-add имя файла
Добавляются конкретный ключ

-l
Показывает отпечатки всех загруженных в данный момент ключей

-L
Посмотреть список самих ключей

-D
Из ssh-agent, будут удалены все ключи

-d имя файла
Удаляет конкретный ключ

-t
Установить время жизни ключей, через данный промежуток времени ключи будут выгружены.

-x
Заблокировать *ssh-agent* паролем

-X
Разблокировать *ssh-agent*

Что-бы закрыть *ssh-agent*, можно вызвать его с опцией **-k**, ну или на крайний случай прибить сигналом, например *kill -QUIT PID*, но это крайняя мера и при корректном запуске, это не потребуется:

```
vds-admin /root/.ssh# ssh-agent -k
unsetenv SSH_AUTH_SOCK;
unsetenv SSH_AGENT_PID;
echo Agent pid 1982 killed;
```

Как видите произошел обратный процесс, переменные очищены, процесс убит.

Форвардинг ssh-agent

Форвардинг агента включается в файле конфигурации клиента *SSH*, директивой **ForwardAgent yes**. Как это работает. Вы запускаете *ssh-agent* на локальной машине, загружаете ключи, подключаетесь к удаленному *SSH* серверу, сервер создает обратное

перенаправление через созданный *SSH* туннель к вашему *ssh-agent* и вы можете использовать загруженные в него ключи для последующих соединений.
Для примера, с локального хоста, **Local_host**, подключаемся к удаленной машине **Remote_host**, по каким-то причинам, нам понадобилось что-то посмотреть на еще одном хосте, **Next_remote_host**, что происходит в таком случае:

- Клиент *ssh* с **Local_host**, подключается к *SSH* серверу, **Remote_host**, и запрашивает форвардинг для *ssh-agent*
- Сервер *SSH*, */usr/sbin/sshd*, хоста **Remote_host**, создает сокет в */tmp/ssh-XXXXXXX/agent.#####* и устанавливает переменную окружения *SSH_AUTH_SOCK*, присваивая ей путь к сокету.
- Когда нам понадобится подключиться к следующему серверу, (мы сейчас на сервере **Remote_host**), *SSH* клиент хоста **Remote_host**, обращается по пути, лежащему в переменной *SSH_AUTH_SOCK*, то есть к сокету.
- *SSH* сервер, находящийся на другом конце сокета */tmp/ssh-XXXXXXX/agent.#####*, передает данные из *ssh*, сервера **Remote_host**, на *ssh-agent*, запущенный на хосте **Local_host**. Вся работа с ключами происходит на машине **Local_host** а не на машинах, на которых вы регистрируетесь в процессе работы.
- Теперь с хоста **Remote_host**, вы можете подключиться к хосту **Next_remote_host**, используя ключи, загруженные в *ssh-agent*, на хосте **Local_host**.

Это только на первый взгляд сложно выглядит, вся эта схема работает абсолютно прозрачно для пользователя, от него требуется только соответствующим образом настроить */etc/ssh/ssh_config* а дплее все просто. Собственно тут даже показывать нечего в качестве примера.