On the fly saving of few useful (?) tech tips
Sakana.fr – A sysadmin's blog
Home
Licence

RSS

Follow me !

Tech@Sakana on Facebook
Like 28

Tech@Sakana on Facebook

FOLLOW ME ON TWITTER

Search this site

Search

Newsletter

Get latest posts by email (No spam, only posts):

Enter your email address:

Subscribe

Delivered by FeedBurner

Categories

- Books (2)
- Dev (23)
- IT (1)
- ITIL (2)
- Misc (1)
- Movies (1)
- Networks (10)
- Security (5)
- Software (24)
- Systems (73)
- ThisBlog (9)
- Uncategorized (2)
- Web (18)

Tags

asterisk cell phone cfengine cluster cross site scripting gentoo gsm hal home automation internet IT linux mobile networking openbsd

opensolaris openssh performance perl phonebook rsync script scripts Security shell sms Solaris ssh symlinks sysadmin Systems tip tips tutorial ubuntu unix VirtualBox virtualization Web windows wordpress Xen xorg xss zfs

Meta

- Log in
- Entries RSS
- Comments RSS
- WordPress.org

Monthly archives

May 2008

M	T	W	T	F	S	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

« Apr Jun »

Securing automated rsync over SSH

Quoting the RSYNC homepage : "rsync is an open source utility that provides fast incremental file transfer."

To make rsync both secure and automated (i.e : non-interactive), you can use SSH as the transport and set up a key pair. This is what will be discussed in this post, along with a few improvements.

Basic rsync + ssh

Let's first ensure that rsync works correctly over ssh :

```
spaghetti% rsync -avz -e ssh --delete Documents prodigy:/tmp
Password:
building file list ... done
Documents/
Documents/Letters/
Documents/Letters/Santa.odt
[...]
spaghetti%
```

As for the options : **-avz** is for the verbose archive gzip compressed mode. This transfers your files and directories recursively, preserving most of their attributes (date, owner, group, and so on). **–delete** will make rsync to delete the files in the target directory if they don't exist anymore in the source directory. All in all, you should end up with the target and source directories synchronized.

As we are specifying **-e ssh**, all the data are transfered over a secured ciphered SSH session.

Notice that it did ask for the password which is unsuitable for automation/scripting purposes. Let's take care of that.

Setting up an SSH key pair

First let's create the key pair :

```
spaghetti% ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kattoo/.ssh/id_rsa): testRsync
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in testRsync.
```

```
Your public key has been saved in testRsync.pub.
The key fingerprint is:
7a:7f:16:dd:99:06:02:3f:d8:cb:ac:10:91:7b:f5:79 kattoo@spaghetti
spaghetti%
```

Remember to keep the passphrase **empty**, otherwise you'll have to type in that passphrase anytime you'll want to use that key pair, which defeats the automation goal.

We now have the 2 files testRsync which is the private key, and testRsync.pub which is the public key.

To be able to connect with SSH to the remote host using this key pair, we need to add the public key in the ~/.ssh /authorized_keys file on the remote host. We can use the ssh-copy-id utility for this purpose :

```
spaghetti% ssh-copy-id -i testRsync prodigy
Password:
Now try logging into the machine, with "ssh 'prodigy'", and check in:
  .ssh/authorized_keys
to make sure we haven't added extra keys that you weren't expecting.
spaghetti%
```

Let's give it a try :

```
spaghetti% ssh -i ~/.ssh/testRsync prodigy
Last login: Wed May  7 21:41:04 2008 from spaghetti.sakan
Sun Microsystems Inc.   SunOS 5.10      Generic January 2005
$
```

All right, ssh is able to connect without any password to the remote host.

And now, let's glue this back to rsync :

```
spaghetti% rsync -avz -e "ssh -i /home/kattoo/.ssh/testRsync" --delete Documents prodigy:/tmp
building file list ... done
[...]
spaghetti%
```

No password was asked, which is good for our automation purposes, but not so great on a security standpoint. Let's improve this.

Securing this automated rsync over ssh

The major problem so far is that if your account is compromised on the local machine, so is your account on the remote machine, since the malicious user could connect there without having to guess any password.

Fortunately SSH offers the possibility to limit the use of a key pair. Let's have a look at the authorized_keys that we have previously configured on the remote host :

```
$ cat /home/kattoo/.ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAzS6C[...]== kattoo@spaghetti
$
```

This actually allows this key pair to be used to connect from the local host to the remote host without any limitation, when all we want this key pair to do is the rsync transfer.

Let's have a more detailed look at what happens (ssh-wise) when we do the rsync :

```
spaghetti% rsync -avz -e "ssh -vi /home/kattoo/.ssh/testRsync" --delete Documents prodigy:/tmp
OpenSSH_4.7p1 Debian-8ubuntu1, OpenSSL 0.9.8g 19 Oct 2007
[...]
debug1: Offering public key: /home/kattoo/.ssh/testRsync
[...]
debug1: Authentication succeeded (publickey).
[...]
debug1: Sending command: rsync --server -vlogDtprz --delete . /tmp
[...]
building file list ... done
[...]
```

```
spaghetti%
```

Notice that I've added the flag **-v** to ssh, to add verbose logging (I've removed most of the output lines of SSH to keep only what we are interested in).

What we see is that we first connect with the key pair we have installed, and then a command (`rsync --server -vlogDtprz --delete . /tmp`) is executed on the remote host.

If this key pair was only able to launch that very command and nothing else, then we'd be fairly secure. Let's do that, and edit the ~/.ssh/authorized_keys to make it look like this :

```
prodigy% cat /home/kattoo/.ssh/authorized_keys
command="/usr/local/bin/rsync --server -vlogDtprz --delete . /tmp" ssh-rsa AAAAB3NzaC1yc2EAAA[...] ka
prodigy%
```

By adding the `command=` part, we now restrict this key pair to only execute this specific command. No risk that a malicious user could use this unprotected key to gain a shell access to the remote computer or to execute another command.

You can go farther and add "no-pty", "no-agent-forwarding", "no-port-forwarding" to further limit the key pair like this :

```
prodigy% cat /home/kattoo/.ssh/authorized_keys
command="/usr/local/bin/rsync --server -vlogDtprz --delete . /tmp",no-pty,no-agent-forwarding,no-port
prodigy%
```

(note : this is supposed to be 1 single line)

Let's go back to rsync with a verbose ssh to see how it now looks like :

```
spaghetti% rsync -avz -e "ssh -vi /home/kattoo/.ssh/testRsync" --delete Documents prodigy:/tmp
OpenSSH_4.7p1 Debian-8ubuntu1, OpenSSL 0.9.8g 19 Oct 2007
[...]
debug1: Offering public key: /home/kattoo/.ssh/testRsync
debug1: Remote: Forced command: /usr/local/bin/rsync --server -vlogDtprz --delete . /tmp
debug1: Remote: Pty allocation disabled.
debug1: Remote: Agent forwarding disabled.
debug1: Remote: Port forwarding disabled.
[...]
debug1: Remote: Forced command: /usr/local/bin/rsync --server -vlogDtprz --delete . /tmp
debug1: Remote: Pty allocation disabled.
debug1: Remote: Agent forwarding disabled.
debug1: Remote: Port forwarding disabled.
debug1: Authentication succeeded (publickey).
[...]
debug1: Sending command: rsync --server -vlogDtprz --delete . /tmp
debug1: Remote: Missing locale support for LANG=en_US.UTF-8
building file list ... done
[...]
spaghetti%
```

Rsync now runs passwordless and there's no risk that this can be exploited to get access to the remote host beyond rsync.

Bookmark to:

May 7, 2008 at 10:54 pm by Stephane Kattoor
Category: Security

Tags: authorized_keys, linux, rsync, Security, ssh, unix

4 Comments - "Securing automated rsync over SSH"

1. **Backups : a personnal implementation - Tech@Sakana - A sysadmin's blog** wrote on July 5, 2008 at 11:03 pm

   [...] you've been following my blog for a while, you might have seen posts about SSH, RSYNC, ZFS Snapshots and so on. This article aims at describing the big picture, and to explain how [...]

---

2. **Securing rsync « 0ddn1x: tricks with *nix** wrote on July 8, 2008 at 10:05 pm

   [...] Securing rsync Filed under: Linux, Security, Solaris — 0ddn1x @ 2008-07-08 20:05:10 +0000 http://www.sakana.fr/blog/2008/05/07/securing-automated-rsync-over-ssh/ [...]

---

3. **Klaus** wrote on October 15, 2009 at 1:30 pm

   

   Wery nice "nofat" howto.
   Thanks.

---

4. **TurboTad** wrote on August 20, 2010 at 10:01 pm

   

   Definitely a good no-fat howto. Exactly answered my questions on how to lock down authorized-keys rsync transfers.

Write a Comment

Name (required)

eMail (required) (will not be displayed)

Website

Type the two words:

post your comment

Who am I ?

blogroll

- It's write now
- Lifehacker
- Pearsonified
- Slacker Manager
- Terminally Incoherent

Fellow sysadmins

- My SysAd Blog — UNIX
- Standalone Sysadmin

friends

- Aashiyana – Santanu & Pamela
- Another Home Page Blog
- Les Carottes sont crues !
- man women
- Schyzophrenia !

Links

- cfengine
- MisterHouse
- OpenBSD
- OpenSolaris
- SubVersioN
- Unix Tutorials
- Zsh home page