SSH-AGENT - SSH FOR THE PARANOID

We have seen how to setup an ssh login so that you are not asked for a password (see Passwordless SSH), which is tremendously useful. But for those paranoid few of us, we rebel against the idea of something having no password, I mean there is no password! But how do we add a password but still allow for the capacity to do batch processing with ssh? Luckily the people who made ssh must have also been a little paranoid, because there is a little utility called *ssh-agent* to help us.

Getting Started
All you need is ssh installed really and the need or desire to use a passworded certificate in a batch processing environment. I'll assume you do not already have a certificate (you will not be able to see it but I will be using the very original password of *bigpassword*)..

```
[root@lowsend .ssh]# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
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:
e8:4e:30:47:8e:97:68:e4:c4:13:01:8b:20:45:65:19 root@lowsend
```

Now we send it across to the target server so it can be added to the *authorized_keys2* file..

```
[root@lowsend .ssh]# scp /root/.ssh/id_rsa.pub 192.168.10.80:/root/.ssh/
test.pub
root@192.168.10.80's password:
id_rsa.pub                                                    100%
222     0.0KB/s   00:00
```

Once the key has been added, lets try to ssh to our target machine (remember my very original password has to used)..

```
[root@lowsend .ssh]# ssh 192.168.10.80
Enter passphrase for key '/root/.ssh/id_rsa':
Last login: Sat Aug 13 21:17:21 2005 from 10.0.0.50
[root@localhost root]#
```

Paranoid and Automatic
As you saw our certificate has a password but since it asks for it, we can use it in an non-interactive script. Or at least we cannot without a little bit more work. Remember I mentioned a useful little tool called *ssh-agent*? Lets take a look..

```
[root@lowsend .ssh]# ssh-agent
SSH_AUTH_SOCK=/tmp/ssh-Gkg16457/agent.16457; export SSH_AUTH_SOCK;
SSH_AGENT_PID=16458; export SSH_AGENT_PID;
echo Agent pid 16458;
```

When you run it, you will have to take the output it produces and run it at the command line, this can done by typing or by redirecting the output into a file, making it executable and then running it. Whichever way, you should now see a *ssh-agent* process in your process listing. Now lets add the key we want to use..

```
[root@lowsend .ssh]# ssh-add /root/.ssh/id_rsa
Enter passphrase for /root/.ssh/id_rsa:
Identity added: /root/.ssh/id_rsa (/root/.ssh/id_rsa)
```

When you run *ssh-add* against your private keyit will ask you to enter in your passphrase, but after that you should be able to ssh to the target server with out any prompting. Lets try..

```
[root@lowsend .ssh]# ssh 192.168.10.80
Last login: Sat Aug 13 21:18:58 2005 from 10.0.0.50
[root@localhost root]#
```

And there you have it. As long as the *ssh-agent* process is running you will be able to login to the target server without any passphrase prompting. One thing to remember though, if for whatever reason the *ssh-agent* process stops (a reboot, a mistyped kill command, etc) then you will need to redo this process to allow for automatic logins again, but for the paranoid of us that's a small price to pay.

Final Words

There you go, another way to make ssh more secure, but this is also a very good example of the fact that security always costs ease-of-use. We have made our batch processes more secure, but we have also made a little more work for ourselves if something happens to the server. You need to choose how much ease-of-use you're willing to give up for security. Anyway, as always have fun and learn.