Using Expect to Transfer SSH Keys

Finnbarr P. Murphy

(fpm@fpmurphy.com)

TCL (Tool Command Language, typically pronounced as “tickle”) is a scripting language created by John Ousterhout in 1988 while working at the University of California, Berkeley. To provide additional functionality, TCL supports extensions. One of the most popular extensions is Expect which is a tool developed by Don Libes for automating and testing interactive applications such as *ftp*. *telnet* and *ssh*.

I assume you know how to generate SSH keys using *ssh-keygen* and understand that if you want to be able to *ssh* to a remote system without entering a password you need to create the keys without a *passphrase*. Once the private and public keys are generated the public key have to be copied to the the user account on the remote system that you wish to have access to and added to the remote users *authorized_keys* file. Typically *scp* or *ftp* is used for that task followed by ssh’ing to the remote system and then doing something like the following:

```
$ mkdir .ssh
$ chmod 700 .ssh
$ cd .ssh
$ touch authorized_keys
$ chmod 600 authorized_keys
$ cat ../id_dsa.pub >> authorized_keys
$ rm ../id_dsa.pub
```

Alternatively you can use the *ssh-copy-id* script to do all the above work for you. This is an interactive script that requires you to the enter the remote user’s password and possibly enter *yes* to add your local host to the remote user’s *known_hosts* file.

Here is a simple shell script which automates the generation of DSA keys with no *passphrase* for a user and transfers the public DSA key to a remote system using expect. Basically it is a wrapper around *ssh-keygen* and *ssh-copy-id*.

```
#!/bin/bash
#
# Copyright (c) Finnbarr P. Murphy 2006
#
# ---- start configurables ----
PATH=/usr/sbin:/usr/bin:/sbin:/bin
LOCALHOMEDIR=/home
# ----- end configurables -----
function usage {
    printf "Usage: setupkeys -U remote-username -P remote-password -H remote-host -u local
-username\n"
    exit 2
}
# --- script starts here
echo
(( $# == 0 )) &amp;&amp; usage
(( $EUID != 0 )) &amp;&amp; {
    echo "ERROR: You must be root to run this script."
    exit 1
}
username=""
password=""
```



```
host=""
localuser=""
while getopts "u:P:U:H:" OPTION
do
    case $OPTION in
        U)  username="$OPTARG";;
        P)  password="$OPTARG";;
        H)  host="$OPTARG";;
        u)  localuser="$OPTARG";;
      esac
done
# --- basic argument checking
if [[ -z "$username" ]]; then
    echo "ERROR - No username entered."
    exit 1
fi
if [[ -z "$password" ]]; then
    echo "ERROR - No passed entered."
    exit 1
fi
if [[ -z "$host" ]]; then
    echo "ERROR - No host entered."
    exit 1
fi
if [[ -z "$localuser" ]]; then
    echo "ERROR - No localuser entered."
    exit 1
fi
# --- do some sanity checking here
echo -n "Checking if $localuser in /etc/passwd. "
grep "^$localuser:" /etc/passwd > /dev/null 2>&amp;1
RESULT=$?
if (( RESULT == 1 )); then
    echo; echo "ERROR - $localuser not found in /etc/passwd."
    exit 1
fi
echo "Yes"
echo -n "Checking connectivity with $host. "
/bin/ping -q -c 2 $host > /dev/null 2>&amp;1
RESULT=$?
if (( RESULT == 1 )); then
    echo; echo "ERROR - could not ping $host."
    exit 1
fi
echo "System is alive."
# --- check for $localuser public and private ssh keys
# --- we need to be $localuser here when using ssh-* utilities
echo -n "Checking for $localuser ssh key files. "
SSH_KEYS_FOUND=0
if [[ -d $LOCALHOMEDIR/$localuser ]]; then
   if [[ -s $LOCALHOMEDIR/$localuser/.ssh/id_dsa &amp;&amp; -s $LOCALHOMEDIR/$localuser/.s
sh/id_dsa.pub ]]; then
      sudo -u $localuser -- /usr/bin/ssh-keygen -e -f $LOCALHOMEDIR/$localuser/.ssh/id_dsa.
pub |  grep "1024-bit DSA"  > /dev/null 2>&amp;1
      RESULT=$?
      if (( RESULT == 0 )); then
          SSH_KEYS_FOUND=1
      fi
   fi
fi
if (( SSH_KEYS_FOUND == 1 )); then
   echo "Found"
else
   echo "Not found"
   rm -rf $LOCALHOMEDIR/$localuser/.ssh
   mkdir $LOCALHOMEDIR/$localuser/.ssh
   chmod 700 $LOCALHOMEDIR/$localuser/.ssh
   chown -R $localuser:$localuser $LOCALHOMEDIR/$localuser/.ssh
```



```
   sudo -u $localuser -- /usr/bin/ssh-keygen -q -t dsa -N "" -f $LOCALHOMEDIR/$localuser/.
ssh/id_dsa
   echo "New ssh key files generated (DSA protocol)"
fi
# --- add $localname's public key to $username@$host authorized keys
TMPUSER=expectscript-user.$$
cat <<EOT > $TMPUSER
#!/usr/bin/expect
if {[llength \$argv] != 4} {
   puts "usage: \$argv0 localuser username password host"
   exit 1
}
log_file -a expectscript-user.log
log_user 0
set localuser [lindex \$argv 0]
set username  [lindex \$argv 1]
set password  [lindex \$argv 2]
set host      [lindex \$argv 3]
set timeout 60
spawn /usr/bin/ssh-copy-id -i \$localuser/.ssh/id_dsa.pub \$username@\$host
expect {
    "assword: " {
        send "$password\n"
        expect {
            "again."     { exit 1 }
            "expecting." { }
            timeout      { exit 1 }
        }
    }
    "(yes/no)? " {
        send "yes\n"
        expect {
            "assword: " {
                send "$password\n"
                expect {
                    "again."     { exit 1 }
                    "expecting." { }
                    timeout      { exit 1 }
                }
            }
        }
    }
}
exit 0
EOT
echo -n "Copying $localuser's public key to $host. "
chmod 755 $TMPUSER
sleep 3
./$TMPUSER $LOCALHOMEDIR/$localuser $username $password $host
RESULT=$?
rm -f $TMPUSER
if (( RESULT == 0 )); then
   echo "Succeeded"
   rm -f expectscript-user.log
else
   echo "Failed"
   echo "ERROR: Check expectscript-user.log"
   exit 1
fi
# we are done
echo "Setup completed. Goodbye."
exit 0
```


You may have to modify it if you use a shell other than *bash* as I have only tested in on Linux systems. If you want to use RSA, it is easy to modify the script to generate RSA keys.

