

Automating Configurations And Backups With Expect

Michael J. Martin

This month we look at automating router configuration changes and configuration backups. Even the most static of routing environments requires periodic configuration changes; SNMP community strings, TACACS/RADIUS security keys, quality of service (QoS) policies, and access control lists (ACLs) are the most common culprits. Aside from convenience, using automated scripts provides certain advantages over making changes by hand. A big one is the elimination of fat finger mistakes. Automating allows the changes to be consistent across the environment, because all routers are configured from the same config file. Automated configuration backups also allow you to keep track of changes that occur on routers between configuration runs. At the end of the change run, comparisons of current and previous configurations can be made by the configuration tool and reported via e-mail.

The solution we will examine has three components. First is a Trivial File Server (TFTP) to provide file exchange services. Second is a Bash (Borne Again Shell) script called *config-rover.sh*, which is executed as a daily or weekly *Cron* or *At* job that is responsible for file management, process control and reporting functions. And last is an Expect script called *config-engine.exp*, which handles the command interaction with the router, uploading changes and backing up the router's running configuration. The scripts can be run from any system that can support TFTP, Bash and Expect -- in other words, any Windows 98/NT/2000 system running the CYGWIN GNU tools environment for Windows, available at <http://www.cygwin.com/>. Any Unix system loaded with the appropriate software can also run them. As Unix is a primary platform for all of these tools, our examples will be based on a Linux system running Red Hat 6.2. First, we will examine TFTP and the configuration of a Linux TFTP server and operation of the Unix command *chmod*, which is used for setting file access permissions. The second part of the article consists of a brief overview of installing Expect and configuring Cron jobs to run the tool, followed by the complete scripts that document logic functionality and operation.

Trivial File Transfer Protocol Services

The Trivial File Transfer Protocol defines a simple client and server model to provide simple file transfer services. Originally developed to provide an easy method of downloading bootstrap information to diskless workstations, it has become the preferred method for uploading/downloading configuration information and operating software for network devices with limited operating systems. It is designed to work in fairly stable networking environments; it uses UDP to provide unreliable transport service with the server listening on port 69.

TFTP follows a client/server model with separate applications for each function and typically has no provision for user authentication. In traditional (Unix-based) implementations, files for uploading and downloading are located within a "root server" directory that is readable and writeable to all system users. The TFTP server "root server" directory is set as a configuration parameter. The "files" themselves are configured to be world-readable or world-writable depending on their use. To copy a file from the server, it must be world-readable. To write a file to the server, it must already exist within the server tree and be world-writable. It is important to keep in mind that TFTP provides no file locking or version tracking, so world-writable files can be overwritten without any notice to the user. TFTP implementations on Windows and Macintosh systems do not operate under the same file permissions construct and so permit the creation of new files when writes to the server are made and no restrictions are placed on downloads.

With its lack of any access security or file control, TFTP is quite dangerous to operate and represents a security risk when running. For this reason, it usually needs to be added to a Unix system, or if it is

Automating Configurations And Backups With Expect

Michael J. Martin

installed as part of the OS install it is un-configured. TFTP security can be improved by using tools like TCP Wrappers and IP Chains to restrict access to the server. The very insecure Windows and Macintosh implementations should only be run when needed and never operated on a continual basis.

Configuring TFTP

There are a number of Macintosh and Windows TFTP server implementations. Cisco provides a Windows implementation on its Web site at <http://www.cisco.com/pcgi-bin/tablebuild.pl/TFTP>. One of the most recent additions to the number of TFTP servers for the Macintosh platform is available at Mac Technologies at <http://www.mactechnologies.com/pages/downld.html>.

Configuring TFTP on the Unix platform is a bit of a chore, due largely to the file permission requirements. You may complain, but these permissions represent what minimal security TFTP offers. If the permissions are not set correctly, access is denied.

The TFTP server runs as a service daemon, controlled by the Unix network super-daemon, *inetd*. It is not installed as part of the standard Red Hat installation and needs to be added after *inetd* has been installed. The TFTP implementation for Linux has the server `ftp://ftp.rutgers.edu/pub/redhat/redhat-6.2-en/os/i386/RedHat/RPMS/TFTP-server-0.16-5.i386.rpm` and the client `ftp://ftp.rutgers.edu/pub/redhat/redhat-6.2-en/os/i386/RedHat/RPMS/TFTP-0.16-5.i386.rpm` as two separate components. (The RPMs are located on the install disk in the `/RedHat/RPMS/` directory.) Both are installed as root using the command `<rpm -ihv {package name}>`. Here is a command line example:

```
[root@parsafal /root]# rpm -ihv TFTP-server-0.16-5.i386.rpm
TFTP-server #####
[root@parsafal /root]#
```

The service daemon is located in `/usr/sbin/in.tftpd`, the same location for *inetd*. Once installed, the service must be configured in `/etc/inetd.conf`. The *inetd.conf* file that comes with Red Hat has a default entry for *in.TFTPd* configured to use TCP Wrappers (*tcpd*), but it is commented out. To enable *inetd* to launch *in.TFTPd*, remove the "#," set the server "root" directory, (if you wish to run *in.TFTPd* without TCP Wrapper support change server path from `/usr/sbin/tcpd` to `/usr/sbin/in.TFTPd`) and restart *inetd*. Here is an example of the *inetd.conf* file entry:

```
# TFTP service is provided primarily for booting. Most sites
# run this only on machines acting as "boot servers." Do not uncomment
# this unless you *need* it.
#
```

```
TFTP dgram udp wait root /usr/sbin/in.TFTPd in.TFTPd /TFTPboot
#bootps dgram udp wait root /usr/sbin/tcpd bootpd
```

To restart *inetd*, use the `<kill>` command. Here is an example of the full command:

```
[root@parsafal /etc]# kill -HUP `cat /var/run/inetd.pid`
```

Automating Configurations And Backups With Expect

Michael J. Martin

To verify that the TFTP service is running, you can use a combination of the `<netstat>` and `<grep>` commands.

```
[root@parsafal /etc]# netstat -a | grep udp
udp 0 0 *:syslog *.*
udp 0 0 *:TFTP *.* <- Here is the service entry
udp 0 0 orion.core.outland.:ntp *.*
udp 0 0 parsafal:ntp *.*
udp 0 0 *:ntp *.*
[root@parsafal /etc]#
```

Assigning file access permissions

Once the server is up and running, it is time to create the readable and writeable placeholder files within the "root server" tree the using the `<touch>` command. The final task is to set the file access permissions correctly using the `<chmod>` command. Chmod permissions can be set using either octal or mnemonic syntax.

Table 1

Access Rights	Owner (U)	Group (G)	Other (O)
No Privileges	0	0	0
Execute (X)	1	1	1
Write (W)	2	2	2
Write & Execute (WX)	3	3	3
Read (R)	4	4	4
Read & Execute (RX)	5	5	5
Read & Write (RW)	6	6	6
Read, Write & Execute (RWX)	7	7	7

Table 1 lists the mnemonic statements (in brackets) and the octal statements in relation to the three levels of file access permissions provided for under Unix, defined in Table 2.

Table 2

Owner	The user who created the file or assigned using the <code><chown></code> command.
Group	The group ownership designation assigned to file usually inherited from the files owner or assigned using the <code><chgrp></code> command.
Other	The access rights granted to all other users and groups on the system i.e., world-readable, world-writable and world-executable.

Mnemonic permissions are assigned using "+" to add, "=" to set an equivalent, and "-" to remove a privilege. Following a command line format of `<chmod UGO +/-= RWX>`, if specific privileges need to be assigned to each level, the level/privilege settings can be defined by separating each level/privilege grouping with a comma. For example, say you want to create a temp file to be written via TFTP. The file is first created using the `<touch>` command. Then the access privileges for user and group are removed, and read and write are granted to other. The permissions are then checked

Automating Configurations And Backups With Expect

Michael J. Martin

using the `</s>` command with the `"-l"` flag. Permissions are indicated in mnemonic form reading from left to right: user, group, and other.

```
[root@parsafal /TFTPboot]# touch norway-gw.conf
[root@parsafal /TFTPboot]# chmod ug-rwx,o=rw norway-gw.conf
[root@parsafal /TFTPboot]# ls -la norway-gw-201.6.56.1.conf
-----rw- 1 root root 0 May 5 16:46 norway-gw.conf
[root@parsafal /TFTPboot]#
```

Octal permissions are assigned by setting a integer value from 0 to 7 representing the desired privileges for each access level in a left to right order for user, group, and other. Here is the command output for the previous example using octal instead of mnemonic encoding:

```
[root@parsafal /TFTPboot]# chmod 006 norway-gw.conf
[root@parsafal /TFTPboot]# ls -la norway-gw.conf
-----rw- 1 root root 0 May 5 16:46 norway-gw.conf
[root@parsafal /TFTPboot]#
```

Which is the better way: octal or mnemonic? The answer is whichever way you feel most comfortable. The advantage of mnemonic is its granularity. And the advantage for octal is that, for many users, it is easier to remember, because the access rights are additive, starting with 0 for none and ending with 7 for full access.

Getting Started with Expect

Expect is a Tool Control Language (TCL) derived tool for scripting interactive sessions. It was created by Don Libes in 1991. TCL, pronounced "tickle," was created by John Ousterhout and his students at University Of California at Berkeley in the late 1980's. TCL and its companion graphic package Tk were created to control and extend existing applications through interactive scripts (TCL) and a scriptable GUI provided by Tk. Both Expect and TCL are easy to learn and are great tools for "non-programmers" because they handle many of the more complicated programming tasks associated with creating applications from scratch with C. TCL is available for most Unix platforms, Windows and Macintosh at <http://tcl.activestate.com/software/tcltk/>.

Expect for Unix is available at <http://expect.nist.gov/>, and a port for Windows is available at <http://bmc.berkeley.edu/people/chaffee/tcltk.html>. If you are interested in learning more about these applications, John K. Ousterhout has written an excellent book titled "Tcl and the Tk Toolkit," published by Addison-Wesley. Don Libes has also written a book on Expect, titled "Exploring Expect," published by O'Reilly.

The reader is not required to have any knowledge of TCL or Expect to run the following scripts, aside from installing the applications properly. But if you find this tool useful you may want to create some of your own.

Running Config-Rover and Config-Engine

Continuing with our Red Hat TFTP server example, we need to load Expect and TCL. Both rpm's are on the install CD in `/RedHat/ RPMS`, or you can download them using the links

Automating Configurations And Backups With Expect

Michael J. Martin

ftp://ftp.rutgers.edu/pub/redhat/redhat-6.2-en/os/i386/RedHat/RPMS/expect-5.28-35.i386.rpm and ftp://ftp.rutgers.edu/pub/redhat/redhat-6.2-en/os/i386/RedHat/RPMS/tcl-8.0.5-35.i386.rpm. The packages can be installed using `<rpm -ihv {package}>`. Expect is dependent on TCL; this requires TCL to be installed prior to installing Expect. After the package install, the Expect binary is located in `/usr/bin/expect`.

To use the scripts, copy the text between the `<start copy>` and `<end copy>` markers and paste the text into a text file using your favorite text editor. The files can be located anywhere; `/usr/local/etc` is the location defined in the scripts but an alternative location can be used -- just change the definitions in the scripts. Once the files have been created, set the permissions using `chmod`.

```
[root@parsafal /root]# chmod u=rwx,go-rwx config-rover.sh
[root@parsafal /root]# ls -la config-rover.sh
-rwx----- 1 root root 1728 May 6 00:39 config-rover.sh
[root@parsafal /root]# chmod u=rwx,go-rwx config-eng.exp
[root@parsafal /root]# ls -la config-eng.exp
-rwx----- 1 root root 1440 May 5 21:41 config-eng.exp
[root@parsafal /root]#
```

With the scripts installed, we need to get them running as automated jobs using Cron. Red Hat provides default Cron configuration that provides command execution on an hourly, daily, weekly and monthly basis. Alternatively, a crontab file for root can be created and jobs can be entered for timely execution. See the man page for Cron for more information on setting up your own crontab file.

Admittedly, using the default Cron implementation is a little more of a pain than using a custom crontab file, but is the easiest method to manage for novice Unix users. The default implementation is modeled to support standalone script execution, so to use this model a "wrapper" script needs to be created for each router we want to configure and backup. The wrapper is a simple single line script that contains the command and command flags we wish to run and directs their standard and error event output to `/dev/null`. Here is an example:

```
#!/bin/sh
/usr/local/etc/config-rover.sh 172.30.71.1 outland-gw > /dev/null 2>&1
```

Once the wrapper is complete, set the appropriate permissions, and install the script in the appropriate Cron job directory. Here is an example where the wrapper script "outland-gw" is installed to run on a daily basis.

```
[root@parsafal /root]# chmod u=rwx,go-rwx outland-gw
[root@parsafal /root]# mv outland-gw /etc/cron.daily/
```

Now, all we are left with is the scripts. The documentation for their operation and each of the functions they perform is within the script text. They are usable under the GNU open source license. They are made available without warranty but have been tested on FreeBSD, Linux 6.2 and 7.x. I hope you find them useful -- enjoy and good luck.

Automating Configurations And Backups With Expect

Michael J. Martin

```
Config-Rover.sh
<start copy>
#!/bin/sh
#
# The Config-Rover script is command and control wrapper for an Expect
# script, called config-engine.exp. These tools in combination perform
router
# configuration updates and backups and report these activities via syslog
# or via SMTP mail.
#
# The config-rover/config-engine tool is designed to run from Cron on a
host
# running a TFTP server. As long as the TFTP server is local, the script(s)
# can run as a command line tool.
#
# The rover script requires two command-line values <host IP address> and
the
# short <hostname>. The ip address tells the expect script what address to
# open a VTY session with. The hostname variable is used for naming the
TFTP
# server directory and the configuration backup filenames. The fully
# qualified domain name can be used but will result in long TFTP directory
# and file names.
#
# In addition to the cli variables, two script variables "TFTP=<Serv IP
# Addr>"and the "TROOT=<dir>" need to be defined.
#
# The "TFTP=" variable should be the ip address of the TFTP server. In
order
# for the file comparison and reporting functions to work the rover/config-
# eng tools should be run from the same host that is functioning as the
TFTP
# server. If these functions are not required, comment out the SYSLOG
report
# and Mail Report sections.
#
# command syntax rover <host ip addr> <hostname>
#
# User Defined Config Variables

# TFTP Server Address
TFTP=172.30.71.103
# TFTP Server root directory (as defined in inetd.conf)
TROOT=/TFTPboot
# User or Users addresses that receive event reports via SMTP mail
RTP="admins@outland.net,mmartin@outland.net"
```

Automating Configurations And Backups With Expect

Michael J. Martin

```
# Under normal operating conditions, i.e., run from the host functioning as  
# the TFTP server. No changes are required beyond this point.
```

```
# Display GNU Banner
```

```
echo "Configuration-Rover, Copyright (C) 2002 Michael J Martin.
```

```
Config-Rover is free software; you can redistribute it and/or  
modify it under the terms of the GNU General Public License  
as published by the Free Software Foundation; either version 2  
of the License, or (at your option) any later version.
```

```
This program is distributed in the hope that it will be useful,  
but WITHOUT ANY WARRANTY; without even the implied warranty of  
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  
GNU General Public License for more details.
```

```
This is free software, released under the terms defined under the  
GNU General Public License and you are welcome to redistribute  
it under the terms defined within the license.
```

```
You should have received a copy of the GNU General Public License  
along with this program; if not, write to the Free Software  
Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  
02111-1307, USA."
```

```
# Test for command line variables:  
# Was a target address provided?
```

```
if [ "$1" = "" ]  
then  
echo "Target IP address is required, run command with -h to see required  
variables";exit  
elif [ "$1" = "-h" ]  
then  
echo "syntax <Target IP Address> <Target Hostname>";exit  
fi
```

```
# Was a hostname provided?
```

```
if [ "$2" = "" ]  
then  
echo "Hostname is required";exit  
fi
```

```
# Create TFTP directory and placeholder file
```

Automating Configurations And Backups With Expect

Michael J. Martin

```
# Event Stamp Format
TS=`date +%b-%d`
# TFTP Directory-Name format
TFTPD="$TROOT/$2-$1"
# TFTP Filename
TFTPF="$TROOT/$2-$1"/$2-$TS.conf

# Setting TFTP Server Paramaters
mkdir $TFTPD > /dev/null 2>&1
chmod 777 $TFTPD
touch $TFTPF
chmod 006 $TFTPF

# Create a backup of current saved configuration, once the config-eng runs
# the configuration backup file will be overwritten. If mail event
# reporting
# is used this file will be compared to newly created configuration backup
# after the config-eng.exp script has run.

cp $TFTPF $TFTPD/check.conf > /dev/null 2>&1

#
# Check to see if a router configuration file exists, if it does not create
# a
# placeholder file. The config-engine script reads this file and uploads
# the router configuration commands to the router
#

if [ `ls /TFTPboot/config.conf` = "/TFTPboot/config.conf" ]
then
echo "config file exists"
else
touch /TFTPboot/config.conf;echo "Created Placeholder config.conf"
fi
chmod 006 /TFTPboot/config.conf

# Run Configuration-Engine

# Define where the expect binary resides
expect=/usr/bin/expect

# Define where the Configuration-Engine script resides
confeng=/usr/local/etc/config-eng.exp

# Run Configuration-Engine script, the target address and other variables
are
```

Automating Configurations And Backups With Expect

Michael J. Martin

```
# passed to configuration-engine as command line variables

expect config-eng.exp $1 $TFTP $TFTP $TROOT > /dev/null 2>&1

#
# Reporting Functions
#
# Syslog Report

logger -p local1.notice $2 configuration complete

# Mail Report
# This function needs to be performed on the host running the TFTP server.
# If the server is not local uncomment the following commands.
#
touch /var/tmp/$2-tempfile > /dev/null 2>&1
diff $TFTP/check.conf $TFTP > /var/tmp/$2-tempfile
#
touch /var/tmp/place.txt > /dev/null 2>&1
echo "`date` Changes on $2" > /var/tmp/place.txt
cat /var/tmp/$2-tempfile >> /var/tmp/place.txt
#
mail -v $RTP -s "Change Report for $2" < /var/tmp/place.txt

# Clean up
rm /var/tmp/$2-tempfile
rm /var/tmp/place.txt
rm $TFTP/check.conf
#
<end copy>
Config-Engine.exp
<start copy>
#!/usr/bin/expect
# Copyright (C) 2002 Michael J Martin
# Made available for use under the GNU General Public License
#
# This Expect script performs configuration uploads and backups on Cisco
# routers using TFTP. It can operate as a standalone script or be executed
# by config-rover.sh a bash script written to manage TFTP control and event
# reporting.
#
# command syntax <host ip addr> <TFTP server addr> <BKU-filename> <TFTP
dir>
#
#
# Operation is dependent on four cli passed variables, router ip, TFTP
server
```

Automating Configurations And Backups With Expect

Michael J. Martin

```
# address, backup file name, and the TFTP server directory where the backup
# file should be saved. When used with config-rover.sh all of these
variables
# are passed to the script.
#
set host [lindex $argv 0]
set TFTP_S [lindex $argv 1]
set TFTP_F [lindex $argv 2]
set TFTP_D [lindex $argv 3]
#
# The script will run on routers configured for local and user based
# authentication.
#
# In order to log into the router the a user/exec password and enable
# password must be defined in the script:
#
# define the enable password here:
set ENPASS suuser

# define the username here ; required if user based authentication is used
set USER c0nflg

# define the local or user account password here
set PASS emmetis1

#
# Start the VTY session, uncomment the appropriate method telnet or ssh
#

spawn telnet $host

# spawn ssh -l $USER $host

# Login to the router
#
expect "Username:" {send "$USERr"}
"Password" {send "$PASSr"}
"refused" exit
#
expect "Password" {send "$PASSr"}
">" {send "enabler"}

#
# Enter into Enable Mode
#
expect ">" {send "enabler"}
"Password:" {send "$ENPASSr"}
```

Automating Configurations And Backups With Expect

Michael J. Martin

```
#
expect "Password:" {send "$ENPASSr"}
"#" {send "r"}
#
expect "#"

# Backup Running Configuration
#

send "copy run TFTPPr"
expect "host"

# Send IP address of TFTP Server
send "$TFTPSr"
expect "filename"

# Send Destination File Name
send "$TFTPPr"

# Configure Router
#

expect "#"

if [ file readable $TFTPD/config.conf ] {
send "copy TFTP://$TFTPS/config.conf runr"
} else {
send logoutr
}
#
expect "]"?
send "r"
expect "#"

## Exit out of the System
send "exitr"
#
<end copy>
```