POOR IT GUY BACKUPS

Backups are a good thing, being able to restore to good data if need be is a good thing. But making sure it happens - properly, securely and easily can cost a lot of money. Now being the person I am (stingy, underfunded and possessed of many other things to spend my money on) I tried to see if I could not make these things happen without spending a cent. I have gotten it working for me, so lets take a trip and see if it helps you..

The backup server
Our story begins with a backup server. You will need this, otherwise you have a lot more problems then you think. I am assuming that the server has the necessary disk space needed and is a linux box. Now I knew I needed to backup files securely so *ssh* was a must but I also wanted my backup server to connect to all the other servers and do the grunt work. This was so that I only had to go to one machine to add backup jobs, test backups, schedule backups, etc. So the first thing I did was setup certificate logins (see here) to the other servers which needed to be backed up, this meant that I could run my backup jobs in batch mode. Then I was trying to figure out how to tar stuff on a remote server over ssh if the backup server logs onto the remote server, I bypassed this problem with the funky *sshfs* which allows you to mount filesystems over ssh - very cool (you will also need the *fuse* programs). Installation of these programs is very easy. So now I could mount the filesystems of the remote servers via ssh, and those ssh logins could happen without user intervention, we are on our way.

The backup script
Now when I started looking at the script I realized I was lazy, I wanted the script to be able to set itself up as much as possible, and the one script should work for all servers. Now my script lives in */admin/bin*, I put the configuration files in */admin/conf* and the backup partition is */data01*. The script (which is here) takes two command line arguments - the first is the ip address of the server which is going to be backed up, the second is a numerical flag which tells the script if it is a full backup or a partial backup (*0* = a full backup and *1* = a partial backup). The script uses configuration files found in the */admin/conf* folder, the filename format of these configuration files is *<backup type>-<server ip address>.bkup*, so an example would be *system-10.0.0.10.bkup*. Contained in these files is a list of files or folders which need to be backed up, so an example would be..

```
/etc/passwd
/etc/group
/var/log
```

When you specify a full backup, the script backups up all the files and folders. When you specify a partial backup the script uses md5deep to check if a file has changed, if it has it then backs it up. The idea is that once a week you run a full backup, and then each other night you run a partial backup. The script also cleans out old backups, you can change the timing for these clean outs by editing the following lines in the script..

```
find $FLL -type f -ctime +21 -exec rm -rf {} \;
find $PTL -type f -ctime +32 -exec rm -rf {} \;
```

The first line checks the full backups and the second line checks the partial backups. The script writes all backup data to a swap folder on the backup partition (for me */data01*) it then moves it into a folder which is named the same as the server ip address being backed up, and then into either a *full* or *partial* folder. So when you run the script you would execute something like..

```
/admin/bin/backup 10.0.0.10 1
```

The script will check to see if the mount point and backup folders exist, if they do not it will create them. This way the only things you need before you can run the backup script against a server is a configuration file (with the properly formatted filename) and the ability to do a passwordless ssh login to that box. I like that, less work for me. Now as with all my scripts, I do not claim that they are perfect or even nicely done, just that they work for me, please feel free to use it if you want, change, amend, whatever.

Backing up servers
So that was all my linux boxes sorted, and life was good. Then I remembered that I had one or two windows boxes on my network - damn I always try not to think of those too much. I really did not want to create a new

backup process just for them. Good news is I did not have to. Enter *copssh*, a openssh server for windows. Installation is simple, point - click - next -next - finish. Creates a service, starts the service. Once the install is finished you need to activate a user so you can logon. Once thats done you just go through the normal process of setting up a certificate login and creating your configuration files with the list of files and folders which need to be backed up. One gotcha is that *copssh* sets up symlinks in it's root folder to the full drives, so your configuration file will look something like..

```
/cygdrive/c/admin/custom
/cygdrive/c/admin/dw
```

Other then that everything else remains the same. So my linux server with it's one backup script backs up all my server no matter the OS - nice.

Final Words

So at the end of the day I have a central backup server with a single script and all backups occur over an encrypted channel. But bear in mind that this does not cater for off site storage, this is a backup solution I use, not a DRP solution. If you want to make changes to my script, please feel free, and as always, have fun and learn.