**Using Netcat for File Transfers**

Netcat is like a swiss army knife for geeks. It can be used for just about anything involving TCP or UDP. One of its most practical uses is to transfer files. Non \*nix people usually don't have SSH setup, and it is much faster to transfer stuff with netcat then setup SSH. netcat is just a single executable, and works across all platforms (Windows, Mac OS X, Linux).

**On the receiving end running**

nc -l -p 1234 > out.file

will begin listening on port 1234.

**On the sending end running**

nc -w 3 [destination] 1234 < out.file

will connect to the receiver and begin sending file.

**For faster transfers if both sender and receiver has some basic \*nix tools installed, you can compress the file during sending process.**

**On the receiving end**

nc -l -p 1234 | uncompress -c | tar xvfp -

**On the sending end**

tar cfp - /some/dir | compress -c | nc -w 3 [destination] 1234

A much cooler but less useful use of netcat is, it can transfer an image of the whole hard drive over the wire using a command called dd.

**On the sender end run**

dd if=/dev/hda3 | gzip -9 | nc -l 3333

**On the receiver end**

nc [destination] 3333 | pv -b > hdImage.img.gz

**Be warned that file transfers using netcat are not encrypted, anyone on the network can grab what you are sending, so use this only on trusted networks.**

**How to transfer files over the network using Netcat**

Introduction

Netcat is a helpful tool when you are in a hurry to transfer files between machines. Learning this tool is a must have skill for any devops, it or software developer. Usually people transfer files using SCP, FTP, SMB but sometimes you don't want to waste time configuring a service.

**IMPORTANT** netcat uses plain transfer. if you want to send confidential information you should encrypt your data before sending it to the network.

Using Netcat

You will have one netcat that will send data and the other one for receiving.

**Receiving side**

nc -l -p 9999 > received\_file.txt

Netcat will start to listen on the port 9999 and the result will be save to the received\_file.txt. If you don't redirect stdout the data received will be printed on screen.

**Sending side**

nc 192.168.0.1 9999 < received\_file.txt

When you are sending you need to specify the address (192.168.0.1) and the port (9999). We redirected the content of the file to netcat, note the order of the "<" is the inverse of the receiving side. You can use -w paramter of the **nc** command to specify a timeout.

Compressing

You can use Linux pipes to compress and decompress

**Receiving side**

The receiving side will first xz, the -c flag is writting to the stdout and -d for decompress. Since uncompress writes to the stdout, then tar will finally unpack the uncompressed data.

nc -l -p 9999 | xz -dc | tar xvf -

**Sending side**

On the sending side we first use tar to pack the directory and then we pipe to xz, when data is compressed we redirect compressed stdout to nc command.

tar cvf - . | xz -c | nc 192.168.0.1 9999

Cool tips

**Use netcat to transfer the whole disk**

On the sending side we use dd and compress with xz:

dd if=/dev/hda3 | xz -c | nc -l 9595

For receiving:

nc 192.168.0.1 9595 | pv -b > hardrive\_backup.img.xz

The pv command allows a user to see the progress of data through a pipeline. We save the hard drive image to a file.