NAME

nc − arbitrary TCP and UDP connections and listens

SYNOPSIS

nc [−e *command*] [−g *intermediates*] [−G *hopcount*] [−i *interval*] [−lnrtuvz] [−o *filename*] [−p *source port*] [−s *ip address*] [−w *timeout*] [*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 source routing. Unlike `telnet`(1), nc scripts nicely, and separates error messages onto standard error instead of sending them to standard output, as `telnet`(1) does with some.

Destination ports can be single integers, names as listed in `services`(5), or ranges. Ranges are in the form nn-mm, and several separate ports and/or ranges may be specified on the command line.

Common uses include:

- simple TCP proxies
- shell−script based HTTP clients and servers
- network daemon testing
- source routing based connectivity testing
- and much, much more

The options are as follows:

−e *command*
: Execute the specified command, using data from the network for stdin, and sending stdout and stderr to the network. This option is only present if nc was compiled with the GAPING_SECURITY_HOLE compile time option, since it allows users to make arbitrary programs available to anyone on the network.

−g *intermediate-host*
: Specifies a hop along a loose source routed path. Can be used more than once to build a chain of hop points.

−G *pointer*
: Positions the "hop counter" within the list of machines in the path of a source routed packet. Must be a multiple of 4.

−i *seconds*
: Specifies a delay time interval between lines of text sent and received. Also causes a delay time between connections to multiple ports.

−l
: Is used to specify that nc should listen for an incoming connection, rather than initiate a connection to a remote host. Any hostname/IP address and port arguments restrict the source of inbound connections to only that address and source port.

−n
: Do not do DNS lookups on any of the specified addresses or hostnames, or names of port numbers from /etc/services.

−o *filename*
: Create a hexadecimal log of data transferred in the specified file. Each line begins with “ <” or “>”. “<” means "from the net" and “>” means "to the net".

−p *port*
: Specifies the source port nc should use, subject to privilege restrictions and availability.

−r
: Specifies that source and/or destination ports should be chosen semi-randomly instead of sequentially within a range or in the order that the system assigns.

−s *hostname/ip-address*
: Specifies the IP of the interface which is used to send the packets. On some platforms, this can be used for UDP spoofing by using `ifconfig`(8) to bring up a dummy interface with the desired source IP address.

−t
: Causes nc to send RFC854 DON'T and WON'T responses to RFC854 DO and WILL requests. This makes it possible to use nc to script telnet sessions. The presence of this option can be enabled or disabled as a compile-time option.

−u
: Use UDP instead of TCP. On most platforms, nc will behave as if a connection is established until it receives an ICMP packet indicating that there is no program listening to what it sends.

−v
: Verbose. Cause nc to display connection information. Using −v more than once will cause nc to become even more verbose.

−w *timeout*
: Specifies the number of seconds nc should wait before deciding that an attempt to establish a connection is hopeless. Also used to specify how long to wait for more network data after standard input closes.

−z
: Specifies that nc should just scan for listening daemons, without sending any data to them. Diagnostic messages about refused connections will not be displayed unless −v is specified twice.

EXAMPLES

`nc`
: Wait for the user to type what would normally be command-line arguments in at stdin.

`nc example.host 42`
: Open a TCP connection to port 42 of example.host. If the connection fails, do not display any error messages, but simply exit.

`nc -p 31337 example.host 42`
: Open a TCP connection to port 42 of example.host, and use port 31337 as the source port.

`nc -w 5 example.host 42`
: Open a TCP connection to port 42 of example.host, and time out after five seconds while attempting to connect.

`nc -u example.host 53`
: Send any data from stdin to UDP port 53 of example.host, and display any data returned.

`nc -s 10.1.2.3 example.host 42`
: Open a TCP connection to port 42 of example.host using 10.1.2.3 as the IP for the local end of the connection.

`nc -v example.host 42`
: Open a TCP connection to port 42 of example.host, displaying some diagnostic messages on stderr.

`nc -v -v example.host 42`
Open a TCP connection to port 42 of example.host, displaying all diagnostic messages on stderr.

`nc -v -z example.host 20-30`
Attempt to open TCP connections to ports 20 through 30 of example.host, and report which ones nc was able to connect to.

`nc -v -u -z -w 3 example.host 20-30`
Send UDP packets to ports 20-30 of example.host, and report which ones did not respond with an ICMP packet after three seconds.

`nc -l -p 3000`
Listen on TCP port 3000, and once there is a connection, send stdin to the remote host, and send data from the remote host to stdout.

`echo foobar | nc example.host 1000`
Connect to port 1000 of example.host, send the string `"foobar"` followed by a newline, and move data from port 1000 of example.host to stdout until example.host closes the connection.

SEE ALSO

`cat`(1), `telnet`(1)

The netcat `README`.

AUTHOR

∗Hobbit∗ [hobbit@avian.org]