Contents:

Introduction:

With the release of Nmap 5.00, the new Ncat tool has been officially released. Ncat is meant to be a modern implementation of Netcat using Nmap's mature networking libraries, combining the best features of the various Netcat derivatives into one new tool. While Ncat is an extremely versatile tool with many amazing new features, it is not quite 100% reverse-compatible with the original Netcat. I decided that it would be helpful to make some kind of chart or guide mapping how to do common Netcat tasks using Ncat, and this is the result.

Some of the new features in Ncat compared to the original Netcat are:

- IPv6 support
- SSL support
- Proxy support (both using a proxy and acting as one!)
- Ability to chain Ncat's together
- Ability to specify specific hosts to allow or deny access to in listen mode
- Connection brokering
- SCTP support

General Tips:

- As of Ncat 5.20, the "`-p`" option can now be used to specify the local port to listen on in listen mode, in order to be reverse-compatible with the original Netcat. This allows for Ncat to be used as a drop-in replacement for most Netcat examples out there! However, the "`-s`" option cannot be used to specify the local IP address to listen on. Instead, simply provide the IP address without preceding it with "`-s`".
- If a host to listen on isn't specified, Ncat listens on all local interfaces. If a port to listen on isn't specified, Ncat listens on port 31337 by default.
- By default, timing arguments are in milliseconds! Append "s", "m", or "h" to the value to specify seconds, minutes, or hours, respectively. Note that this is going to change to seconds in an upcoming version of Ncat (newer than 5.30BETA1).

- By default, Ncat accepts up to 100 simultaneous connections! Use "`-m1`" (that's a numeral one) to limit it to 1 like the original Netcat.
- When using "`-e, --exec`" to execute commands, it is necessary to specify the full path to any executables used. Using "`-c, --sh-exec`" executes via /bin/sh and takes advantage of the PATH enviornment variable.
- UDP mode does not support SSL or connection brokering, likely due to its stateless nature.
- Command execution (`-e, --exec` and `-c, --sh-exec`) cannot be used with connection brokering (`--broker`).
- As of Ncat 5.20, Ncat's command execution options *can* be used with SSL (`--ssl`) and Ncat's output options (`-o` and `-x`)!
- The original stable version of Ncat (5.00) did not properly set the return value based on whether the connection was successful or not. This has been fixed in Ncat 5.20.

Mapping of Netcat options to Ncat options:

Netcat Option	Netcat Description	Ncat Equivilent	Ncat Description
`-d`	[Windows only] detach from console, stealth mode	`[none]`	[DIFFERENT!] `-d, --delay` specifies the wait between read/writes in Ncat
`-e prog`	inbound program to exec [dangerous!!]	`-e, --exec <command>`	Alternatively, use `-c, --sh-exec` to execute via /bin/sh
`-g gateway`	source-routing hop point[s], up to 8	`-g hop1[,hop2,...]`	Loose source routing hop points (8 max)
`-G num`	source-routing pointer: 4, 8, 12, ...	`-G n`	Loose source routing hop pointer (4, 8, 12, ...)
`-h`	this cruft	`-h, --help`	Display this help screen
`-i secs`	delay interval for lines sent, ports scanned	`-d, --delay <time>`	Wait between read/writes

`-l`	listen mode, for inbound connects	`-l, --listen`	By default, Ncat will accept 100 simultaneous connections, instead of 1 like the original netcat. Use `-m` to specify the maximum number of simultaneous connections.
`-L`	[Windows only] listen harder, re-listen on socket close	`-l -k`	With Ncat, use the "`-k`" (`--keep-open`) option along with "`-l`" for the same effect as "`-L`". Also is supported on all platforms with Ncat!
`-n`	numeric-only IP addresses, no DNS	`-n, --nodns`	Do not resolve hostnames via DNS
`-o file`	hex dump of traffic	`-x, --hex-dump`	[DIFFERENT!] With Ncat, "`-o`" saves the text-based data to a file (like running "`ncat [host] [port] | tee [filename]`"; use "`-x`" to save the data in hex. Note the hex output is slightly different from the original netcat!
`-p port`	local port number	`-p, --source-port <port>`	Specify source port for client mode; not required (but works for reverse-compatibility with the original Netcat) in server mode!
`-r`	randomize local and remote ports	`[none]`	No equivilent since this is only used for port scanning (which is what nmap is for)
`-s addr`	local source address	`-s, --source <addr>`	Specify source address to use for client mode; not used in server mode!

`-t`	answer TELNET negotiation	`-t, --telnet`	Answer Telnet negotiations (all platforms)
`-u`	UDP mode	`-u, --udp`	Use UDP instead of default TCP
`-v`	verbose [use twice to be more verbose]	`-v, --verbose`	Issue once (-v) for connection information, twice (-vv) for code debugging information, or three times (-vvv) for both
`-w secs`	timeout for connects and final net reads	`-w, --wait <time>`	Connect timeout
`-z`	zero-I/O mode [used for scanning]	`--send-only --recv-only`	No direct equivilent since this is only used for port scanning (which is what nmap is for), but using --send-only and --recv-only together is the same thing
port numbers can be individual or ranges: m-n [inclusive]		Only individual ports are supported	

New Ncat options:

Ncat option	Description
`-4`	Use IPv4 only (default).
`-6`	Use IPv6 only.
`-C, --crlf`	Use CRLF for EOL sequence (useful for HTTP).
`-c, --sh-exec <command>`	Executes specified command via /bin/sh (allows omission of absolute paths, etc.).
`-m, --max-conns <n>`	Maximum n simultaneous connections (default is 100).

`-k, --keep-open`	Accept multiple connections in listen mode (similar to "`-L`" in original Windows netcat).
`--send-only`	Only send data, ignoring received; quit on EOF.
`--recv-only`	Only receive data, never send anything.
`--allow <host>[,<host>,...]`	Allow specific hosts to connect to Ncat (can use Nmap-style host specification).
`--allow-file <file>`	A file of hosts allowed to connect to Ncat (one per line).
`--deny <host>[,<host>,...]`	Hosts to be denied from connecting to Ncat (can use Nmap-style host specification).
`--deny-file <file>`	A file of hosts denied from connecting to Ncat (one per line).
`--broker`	Enable Ncat's Connection Brokering mode, allowing multiple parties to connect to a centralized Ncat server and communicate with each other (automatically implies listen mode).
`--chat`	Start a simple Ncat chat server using Connection Brokering, giving each user a unique ID and escaping characters that could mess up the terminal.
`--proxy <addr[:port]>`	Specify address of host to proxy through.
`--proxy-type <type>`	Specify proxy type ("http" or "socks4"). Listen mode currently only supports acting as an HTTP proxy, and there is currently no Windows support.
`--proxy-auth <username>:<password>`	Authenticate with HTTP or SOCKS4 proxy server using specified credentials, or require the specified credentials in listen mode.
`--sctp`	Use SCTP instead of default TCP. SCTP support is implemented in TCP compatible mode.
`--ssl`	Connect or listen with SSL. Not supported over UDP.

`--ssl-cert <file>`	Specify SSL certificate file (PEM) for listening or authenticating with a server (one will be generated on the fly if none specified).
`--ssl-key <file>`	Specify SSL private key (PEM) file that goes with the certifcate.
`--ssl-verify`	Verify trust and domain name of certifcates.
`--ssl-trustfile <file>`	PEM file containing trusted SSL certificates.

SSL Support:

Ncat supports SSL, which is useful for many different scenarios, such as securely tranferring files or connecting to SSL-enabled services. For basic file transfer, just add the "`--ssl`" option to both the listener and the client. This will transparently generate a one-time certificate for the connection. To do manual fingerprint verification, use "`-v, --verbose`" to see the SHA-1 fingerprint, or use an already-generated certificate (such as one created with openssl) with the "`--ssl-cert`" and "`--ssl-key`" options.

Pushing a file from a listener to a client over SSL:

```
listener> ncat -v -l --ssl --send-only [LocalPort] < [filename]
client>   ncat -v --ssl --recv-only [RemoteIPaddr] [RemotePort] > [filename]
```

As of version 5.20, Ncat now supports acting as an SSL wrapper around a non-SSL service! One usage case for this feature is setting up an encrypted tunnel for VNC traffic. While there are tons of tutorials on how to set up SSH tunnels for VNC traffic, such a setup can be difficult on Windows systems due to the lack of easy-to-use, well-supported, open-source SSH servers for Windows. However, Ncat is just a simple executable! To set up SSL-enabled VNC, you have to use a listener-to-client relay on both the host running the VNC server and the client running the VNC client. However, there is a bug that prevents VNC from working over Ncat, which seems to be fixed in versions 5.30BETA1 and later.

See the SSL page of the official Ncat guide for more information and examples.

Proxy Support:

Ncat is able to connect through an HTTP or SOCKS proxy, and is able to act as an HTTP proxy in listen mode. While acting as an HTTP proxy, Ncat supports the access control options, as well as `--proxy-auth` to require username/password authentication. For example, here is how to implement a simple HTTP proxy on port 8080 that only accepts connections from a specific network (192.168.1.0/24), also requiring the username "ncatuser" and password "s3cr3tpass":

```
ncat -l -k --proxy-type http --proxy-auth ncatuser:s3cr3tpass --allow
192.168.1.0/24 8080
```

Netcat Relays:

While the following Netcat examples are for a Unix-like system, the Ncat examples will work on all supported platforms, including Windows!

Listener-to-Client Relay:

With the original Netcat, you could implement a listener-to-client relay that sends packets from a local port to a port on a remote system as follows:

```
mknod backpipe p
nc -l -p [LocalPort] 0<backpipe | nc [TargetIPaddr] [port] > backpipe
```

With Ncat's ability to chain multiple Ncat's together, you could replace the above with:

```
ncat -l -m1 [LocalPort] -c "ncat [TargetIPaddr] [port]"
```

Without the "`-m1`" the relay will stay open and allow multiple simultaenous connections! You could also use Ncat's access control functionality to only allow certain systems to access the relay.

Listener-to-Listener Relay:

With the original Netcat, you could implement a listener-to-listener relay that sends packets from any connection on one local port to another local port as follows:

```
mknod backpipe p
nc -l -p [LocalPort_1] 0<backpipe | nc -l -p [LocalPort_2] > backpipe
```

With Ncat, you could use Connection Brokering to achieve the same goal:

```
ncat -l --broker -m2 [LocalPort]
```

With Connection Brokering, multiple clients can connect to a single listening port, and any data sent by any client is sent to all of the other clients. For the purposes of this relay it makes sense to limit Ncat to 2 connections (`-m2`), but this technique could also be used for broadcasting data from one-to-many. It might also make sense to use `--send-only` on the client that is sending data and `--recv-only` on the client that is receiving.

Client-to-Client Relay:

With the original Netcat, you could implement a client-to-client relay that sends packets from a connection to one local port to the connection on another local port as follows:

```
mknod backpipe p
nc [PreviousHopIPaddr] [port] 0<backpipe | nc [NextHopIPaddr] [port2] >
backpipe
```

Using Ncat, the same relay can be implemented as follows:

```
ncat [PreviousHopIPaddr] [port] -c "ncat [NextHopIPaddr] [port2]"
```

This takes advantage of the ability to chain multiple Ncat's together, similar to the listener-to-client relay.

Resources: