

Socat — Professional Cheat Sheet

One-page professional reference for **socat** — multipurpose bidirectional data relay. Practical flags, examples (port forwarding, proxying, file/pty bridging, TLS), patterns, and operational notes for pentesters, devops, and network engineers.

1) At-a-glance

- **Tool:** `socat` (SOcket CAT) — a powerful networking utility that establishes two bidirectional byte streams and transfers data between them. More flexible than `netcat` / `ncat` / `ncat`, with many address types (TCP, UDP, SSL, PTY, UNIX sockets, files, exec, SYSTEM).
 - **Primary uses:** Port forwarding, TCP/UDP proxying, SSL/TLS wrapping, serial-to-network bridging, pseudo-tty shells, local service exposing, and scripted transport transformations.
-

2) Install / check

```
# Debian/Ubuntu
sudo apt update && sudo apt install socat

# Arch
sudo pacman -S socat

# macOS (Homebrew)
brew install socat

# Check version
socat -V
```

3) Basic syntax

```
socat [options] <address1> <address2>
# Example: echo server to client
socat -v TCP-LISTEN:8080,reuseaddr,fork STDOUT
```

- Addresses describe endpoints: `TCP:<host>:<port>`, `TCP-LISTEN:<port>`, `UDP-LISTEN:<port>`, `OPEN:<file>`, `PTY`, `EXEC:'command'`, `UNIX-LISTEN:/path`, `SSL:<host>:<port>`, etc. -
Common options: `fork` (handle multiple clients), `reuseaddr`, `crnl`, `nofork` (default), `delay`, `ignoreeof`, `pty,raw,echo=0` for interactive shells.

4) Common address types & options

- `TCP-LISTEN:port[,options]` — listen TCP server (use `fork` for multiple clients).
 - `TCP:host:port` — connect TCP client.
 - `UDP-LISTEN:port / UDP:host:port` — UDP endpoints.
 - `SSL:host:port,verify=0` — connect with TLS; `SSL-LISTEN:port,cert=cert.pem,key=key.pem` for server.
 - `PTY` — create pseudo-tty (useful for interactive shells or serial devices).
 - `EXEC:'cmd',pty,stderr` — execute program; combine with `PTY` for interactive shells.
 - `SYSTEM:'command'` — run command through system shell.
 - `OPEN:filename,creat,ronly` — file access.
 - `UNIX-LISTEN:/path / UNIX-CONNECT:/path` — Unix domain sockets.
-

5) Practical examples

5.1 Simple TCP connect

```
# Client: connect to remote TCP service and print
socat - TCP:example.com:80
```

5.2 TCP listener (echo)

```
# Echo server: accept connections and echo to stdout (single client)
socat -v TCP-LISTEN:9000,reuseaddr STDOUT

# With fork: handle multiple clients
socat -v TCP-LISTEN:9000,reuseaddr,fork EXEC:'/bin/cat'
```

5.3 Forward local port to remote (local forward)

```
# forward local 8080 to remote:80
socat TCP-LISTEN:8080,reuseaddr,fork TCP:www.example.com:80
```

5.4 Reverse proxy / remote port exposure (remote forward)

```
# On intermediary: listen and forward to internal host
socat TCP-LISTEN:2222,reuseaddr,fork TCP:10.0.0.5:22
```

5.5 Encrypted TLS wrapper (wrap legacy service in TLS)

```
# Server: accept TLS and forward to internal HTTP
socat SSL-LISTEN:443,cert=cert.pem,key=key.pem,reuseaddr,fork TCP:127.0.0.1:80

# Client: connect to TLS-wrapped service (no verification)
socat - SSL:server.example.com:443,verify=0
```

5.6 Simple SOCKS proxy use with socat (connect via SOCKS5)

```
# Use proxychains-like flow (example):
# connect through socks5 proxy at 127.0.0.1:1080 to target
socat - PROXY:127.0.0.1:target:80,proxyport=1080
```

5.7 PTY + shell (remote interactive shell)

```
# Target (listener) – spawn a shell into PTY for incoming connections
socat TCP-LISTEN:4444,reuseaddr,fork EXEC:'/bin/
bash',pty,stderr,setsid,sigint,sane

# Attacker connects
socat - TCP:target:4444
```

5.8 Serial to network bridge (USB/serial device over TCP)

```
socat TCP-LISTEN:2000,reuseaddr,fork FILE:/dev/ttyUSB0,raw,echo=0,nonblock
```

5.9 File transfer

```
# Send file
socat -u FILE:local.bin TCP:remote:9000
# Receive file
socat -u TCP-LISTEN:9000,reuseaddr FILE:received.bin,creat,truncate
```

5.10 Systemd socket example (simple unit)

```
# /etc/systemd/system/socat-echo.service
[Unit]
Description=Socat echo server
[Service]
```

```
ExecStart=/usr/bin/socat -v TCP-LISTEN:9000,reuseaddr,fork EXEC:'/bin/cat'  
Restart=on-failure
```

6) Advanced patterns

- **TLS termination + backend routing:** SSL-LISTEN -> TCP backend (wrap insecure services).
- **Transparent proxying:** combine `IPPROTO` and `SOCKET` options for advanced use (requires root).
- **Chaining socat instances:** chain multiple socat processes for complex piping or protocol translation.
- **Scripting:** use `bash` / `systemd` wrappers to supervise long-running forwards and restart on failure.

7) Troubleshooting & tips

- **Permission errors:** binding privileged ports (<1024) or accessing devices requires root.
- **Stability:** prefer `fork` for concurrent clients; `-u` for unidirectional transfers when appropriate.
- **Binary data / nulls:** use `-u` (unidirectional) or `STDIO` carefully; use `-b` (binary) or `raw` options on endpoints that support it.
- **Quote address strings** when they contain commas or special chars (e.g., `EXEC:'/bin/bash, -l'`).
- **Debugging:** `-d -d -v` increases verbosity for debugging.
- **Timeouts & keepalive:** use `tcpkeepalive` or wrapper scripts to detect and restart dead tunnels.

8) Security & OPSEC

- Avoid exposing interactive shells on Internet-facing hosts.
- Use TLS (`SSL-LISTEN/SSL`) and certificate verification for confidentiality and basic authentication.
- Restrict access via firewall rules and `listenaddr=127.0.0.1` for local-only forwards.
- Monitor for unauthorized socat processes — often used for covert tunnels/backdoors.

9) One-liners (copy-paste)

```
# Local port forward  
socat TCP-LISTEN:8080,reuseaddr,fork TCP:www.example.com:80  
# TLS wrapper  
socat SSL-LISTEN:443,cert=cert.pem,key=key.pem,reuseaddr,fork TCP:127.0.0.1:80  
# Remote shell (target)  
socat TCP-LISTEN:4444,reuseaddr,fork EXEC:'/bin/  
bash',pty,stderr,setsid,sigint,sane
```

```
# Serial to TCP bridge
socat TCP-LISTEN:2000,reuseaddr,fork FILE:/dev/ttyUSB0,raw,echo=0,nonblock
# Send file
socat -u FILE:local.bin TCP:remote:9000
# Receive file
socat -u TCP-LISTEN:9000,reuseaddr FILE:received.bin,creat,truncate
```

10) Alternatives & complements

- **ncat / netcat** — lighter-weight for quick ad-hoc tasks.
 - **ssh -L / -R** — secure port forwarding with authentication.
 - **socat + stunnel** — combine for advanced TLS setups; `stunnel` may be easier for certificates management in some environments.
-

This cheat sheet is intended for authorized testing, diagnostics, and operations. Use `socat` only on systems you own or have explicit permission to use.