

Scanning ports with netcat "nc" command on Linux and UNIX / Checking for firewall filtering between source and destination with nc

Author : admin



netcat_

Netcat (nc) is one of that tools, that is well known in the hacker (script kiddie) communities, but little underestimated in the sysadmin world, due to the fact **nmap (network mapper) - the network exploratoin and security auditing tool** has become like the **standard penetration testing TCP / UDP port tool**.

nc is feature-rich network debugging and investigation tool with tons of built-in capabilities for reading from and writing to network connections using TCP or UDP.

Its Plethora of features includes port listening, port scanning & Transferring files due to which it is often used by Hackers and PenTesters as Backdoor. Netcat was written by a guy we know as the Hobbit .

For a start-up and middle sized companies if nmap is missing on server usually it is okay to install it without risking to open a huge security hole, however in **Corporate world**, due to security policies often **nmap** is not found on the servers but **netcat (nc)** is present on the servers so you have to learn, if you haven't so to use **netcat** for the usual IP range port scans, if you're so used to **nmap**.

There are different implementations of **Netcat**, whether historically netcat was UNIX (BSD) program

with a latest release of March 1996. The [Linux version of NC is GNU Netcat \(official source here\)](#) and is POSIX compatible. The other **netcat in Free Software OS-es** is **OpenBSD's netcat whose ported version is also used in FreeBSD. Mac OS X** also comes with default prebundled netcat on its **Mac OS X from OS X version (10.13)** onwards, on older OS X-es it is installable via **MacPorts package repo**, even **FreeDOS** has a port of it called **NTOOL**.

The (Swiss Army Knife of Embedded Linux) busybox includes a default leightweight version of netcat and **Solaris** has the *OpenBSD netcat version bundled*.

A cryptography enabled version fork exists that supports that supports integrated transport encryption capabilities called **Cryptcat**.

The Nmap suite also has included rewritten version of GNU Netcat named **Neat**, featuring new possibilities such as "Connection Brokering", TCP/UDP Redirection, **SOCKS4** client and server support, ability to "Chain" Neat processes, **HTTP CONNECT proxying (and proxy chaining)**, **SSL connect/listen support and IP address/connection filtering**. Just like Nmap, Neat is cross-platform.

In this small article I'll very briefly explain on basic **netcat** - known as the **TCP Army knife tool port scanning for an IP range of UDP / TCP ports**.

1. Scanning for TCP opened / filtered ports remote Linux / Windows server

Everyone knows scanning of a port is possible with a simple telnet request towards the host, e.g.:

```
telnet SMTP.EMAIL-HOST.COM 25
```

The most basic netcat use that does the same is achiavable with:

```
$ nc SMTP.EMAIL-HOST.COM 25
```

```
220 jeremiah ESMTP Exim 4.92 Thu, 05 Sep 2019 20:39:41 +0300
```

Beside scanning the remote port, using netcat interactively as pointing in above example, if connecting to HTTP Web services, you can request remote side to return a webpage by sending a false *referer*, *source host and headers*, this is also easy doable with *curl* / *wget* and *lynx* but doing it with netcat just like with telnet could be fun, here is for example how to request an INDEX page with spoofed HTTP headers.

```
nc Web-Host.COM 25
```

```
GET / HTTP/1.1
```

```
Host: spoofedhost.com
```

```
Referer: mypage.com
```

```
User-Agent: my-spoofed-browser
```

2. Performing a standard HTTP request with netcat

To do so just type the content with a standard bash integrated **printf function** with the included end of line (the unix one is `\n` but to be OS independent it is better to use `r\n` - the end of line completion character for Windows.

```
printf "GET /index.html HTTP/1.0\r\nHost: www.pc-freak.net\r\n\r\n" | nc www.pc-freak.net 80
```

3. Scanning a range of opened / filtered UDP ports

To scan for lets say opened remote system services on the very common important ports opened from **UDP port 25 till, 1195** - more specifically for:

- *UDP Bind Port 53*
- *Time protocol Port (37)*
- *TFTP (69)*
- *Kerberos (88)*
- *NTP 123*
- *Netbios (137,138,139)*
- *SNMP (161)*
- *LDAP 389*
- *Microsoft-DS (Samba 445)*
- *Route BGP (52)*
- *LDAPS (639)*
- *openvpn (1194)*

```
nc -vzu 192.168.0.1 25 1195
```

UDP tests will show opened, if no some kind of firewall blocking, the **-z** flag is given to scan only for remote listening daemons without sending any data to them.

4. Port Scanning TCP listening ports with Netcat

As prior said using netcat to scan for remote opened HTTP Web Server on port 80 an FTP on Port 23 or a Socks Proxy or MySQL Database on 3306 / PostgreSQL DB on TCP 5432 is very rare case

scenario.

Below is example to scan a Local network situated IP for TCP open ports from port 1 till 7000.

```
# nc -v -n -z -w 5 192.168.1.2 1-7000
```

```
nc: connect to host.example.com 80 (tcp) failed: Connection refused
nc: connect to host.example.com 20 (tcp) failed: Connection refused
Connection to host.example.com port [tcp/ssh] succeeded!
nc: connect to host.example.com 23 (tcp) failed: Connection refused
```

Be informed that scanning with netcat is much more slower, than nmap, so specifying smaller range of ports is always a good idea to reduce annoying waiting ...

The **-w** flag is used to set a timeout to remote connection, usually on a local network situated machines the timeout could be low **-w 1** but for machines across different Data Centers (let say one in Berlin and one in Seattle), use as a minimum **-w 5**.

If you expect remote service to be responsive (as it should always be), it is a nice idea to use netcat with a low timeout (-w) value of 1 below is example:

```
netcat -v -z -n -w 1 scanned-hosts 1-1023
```

5. Port scanning range of IP addresses with netcat

If you have used **Nmap** you know scanning for a network range is as simple as running something like **nmap -sP -P0 192.168.0.*** (to scan from IP range 1-255) **nmap -sP -P0 192.168.0.1-150** (to scan from local IPs ending in 1-150) or giving the network mask of the scanned network, e.g. **nmap -sP 192.168.0.1/24** - for more examples please check my previous article [Checking port security on Linux with nmap \(examples\)](#).

But what if nmap is not there and **want to check a bunch 10 Splunk servers** (software for searching, monitoring, and analyzing machine-generated big data, via a Web-style interface.), **with netcat to find, whether the default Splunk connection port 9997 is opened or not:**

```
for i in `seq 1 10`; do nc -z -w 5 -vv splunk0$i.server-domain.com 9997; done
```

6. Checking whether UDP port traffic is allowed to destination server

Assuring you have access on **Source traffic (service) Host A** and **Host B** (remote destination server where a daemon will be set-upped to listen on UDP port and no firewall in the middle Network router or no traffic control and filtering software HUB is preventing the sent UDP proto traffic, lets say an **ntpd** will be running on its standard 123 port there is done so:

- On host B (the remote machine which will be running ntpd and should be listening on port 123), run netcat to listen for connections

```
# nc -l -u -p 123  
Listening on [0.0.0.0] (family 2, port 123)
```

Make sure there is no **ntpd** service actively running on the server, if so stop it with **/etc/init.d/ntpd stop** and run above command. The command should run as superuser as UDP port 123 is from the so called low ports from 1-1024 and binding services on such requires root privileges.

- On Host A (UDP traffic send host

nc -uv remote-server-host 123

```
root@jeremiah:~# nc -lv -u -p 123
Listening on [0.0.0.0] (family 2, port 123)
Connection from localhost 50568 received!
XXXXX
```

If the remote port is not reachable due to some kind of network filtering, you will get "**connection refused**".

An important note to make is on some newer Linux distributions **netcat might be silently trying to connect by default using IPV6**, bringing false positives of filtered ports due to that. Thus it is generally a good idea, to make sure you're connecting to **IPV6**

\$ nc -uv -4 remote-server-host 123

Another note to make here is netcat's UDP connection takes 2-3 seconds, so make sure you wait at least

4-8 seconds for a very distant located hosts that are accessed over a multitude of routers.

7. Checking whether TCP port traffic allowed to DST remote server

To listen for TCP connections on a specified location (external Internet IP or hostname), it is analogous to listening for UDP connections.

Here is for example how to bind and listen for TCP connections on all available Interface IPs (localhost, eth0, eth1, eth2 etc.)

```
nc -lv 0.0.0.0 12345
```

Then on client host test the connection with

```
nc -vv 192.168.0.103 12345  
Connection to 192.168.0.103 12345 port [tcp/*] succeeded!
```

8. Proxying traffic with netcat

Another famous hackers use of Netcat is its proxying possibility, to proxy anything towards a third party application with UNIX so any content returned be printed out on the listening **nc** spawned daemon like process.

For example one application is traffic SMTP (Mail traffic) with netcat, below is example of how to proxy traffic from **Host B** -> **Host C** (in that case the yandex current mail server **mx.yandex.ru**)


```
linux-srv:~# nc -l 12543 | nc mx.yandex.ru 25
```

Now go to **Host A** or any host that has **TCP/IP** protocol access to **port 12543** on proxy-host **Host B** (**linux-srv**) and connect to it on 12543 with another **netcat** or **telnet**.

to make netcat keep connecting to yandex.ru MX (Mail Exchange) server you can run it in a small **never ending bash shell while loop**, like so:

```
linux-srv:~# while ;; do nc -l 12543 | nc mx.yandex.ru 25; done
```

Below are screenshots of a connection handshake between **Host B** (linux-srv) proxy host and **Host A** (the end client connecting) and **Host C** (*mx.yandex.ru*).

```
root@ [REDACTED] ~# while ;; do nc -l 12543 | nc mx.yandex.ru 25; done
220 mxfront10q.mail.yandex.net (Want to use Yandex.Mail for your domain? Visit http://pdd.yandex.ru)
250 mxfront10q.mail.yandex.net
250 2.1.0 <hipo@mail.com> ok
250 2.1.5 <hip0d@yandex.ru> recipient ok
354 Enter mail, end with "." on a line by itself
451 4.7.1 Sorry, the service is currently unavailable. Please come back later. 1567700751-RD8BVKMTbg-PTxwNbxX
250 mxfront10q.mail.yandex.net
250 2.1.0 <hipopo@mail.com> ok
250 2.1.5 <postmaster@yandex.ru> recipient ok
354 Enter mail, end with "." on a line by itself
451 4.7.1 Sorry, the service is currently unavailable. Please come back later. 1567700810-RD8BVKMTbg-QQxw5TF9
421 4.4.2 mxfront10q.mail.yandex.net Error: timeout exceeded
```

Host B netcat as a (Proxy)

```
hipo@jeremiah:~$ telnet localhost 12543
Trying ::1...
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
HELO localhost
MAIL FROM:<hipo@mail.com>
RCPT TO:<hip0d@yandex.ru>
DATA
Just a simple test
.
HELO localhost
MAIL FROM:<hipopo@mail.com>
RCPT TO:<postmaster@yandex.ru>
DATA
Another test
.
```

that is possible in combination of UNIX and named pipes (for more on Named pipes [check my previous article simple linux logging with named pipes](#)), here is how to run a single netcat version to proxy any traffic in a similar way as the good old **tinypoxy**.

On Proxy host create the pipe and pass the incoming traffic towards **google.com** and write back any output received back in the named pipe.

```
# mkfifo backpipe
# nc -l 8080 0backpipe
```

Other useful netcat proxy set-up is to simulate a network connectivity failures.

For instance, if server:port on TCP 1080 is the normal host application would connect to, you can to set up a forward proxy from port 2080 with

```
nc -L server:1080 2080
```

then set-up and run the application to connect to localhost:2080 (nc proxy port)

```
/path/to/application_bin --server=localhost --port=2080
```

Now application is connected to localhost:2080, which is forwarded to server:1080 through netcat. To simulate a network connectivity failure, just kill the netcat proxy and check the logs of *application_bin*.

Using netcat as a bind shell (make any local program / process listen and deliver via nc)

netcat can be used to make any local program that can receive input and send output to a server, this use is perhaps little known by the junior sysadmin, but a favourite use of *1337 h4x0rs* who use it to spawn shells on remote servers or to make connect back shell. The option to do so is **-e**

-e - option spawns the executable with its input and output redirected via network socket.

One of the **most famous use of binding a local OS program to listen and receive / send content** is by making **netcat** as a *bind server* for local **/bin/bash** shell.

Here is how

```
nc -l -p 4321 -e /bin/sh
```

If necessary specify the bind hostname after -l. Then from any client connect to 4321 (and if it is opened) you will gain a shell with the user with which above netcat command was run. Note that many modern distribution versions such as **Debian / Fedora / SuSE Linux's netcat** binary is compiled without the **-e option** (this works only when compiled with **-DGAPING_SECURITY_HOLE**), removal in this distros is because option is potentially opening a security hole on the system.

If you're interested further on few of the methods how modern hackers bind new backdoor shell or connect back shell, check out [Spawning real tty shells article](#).

For more complex things you might want to check also **socat** (SOcket CAT) - multipurpose relay for bidirectional data transfer under Linux.

socat is a great **Linux Linux / UNIX TCP port forwarder tool** similar holding the same spirit and functionality of netcat plus many, many more.

On some of the many other UNIX operating systems that are lacking netcat or nc / netcat commands can't be invoked a similar utilities that should be checked for and used instead are:

ncat, pnetcat, socat, sock, socket, sbd

To use nmap's ncat to spawn a shell for example that allows up to 3 connections and listens for connects only from 192.168.0.0/24 network on port 8081:

```
ncat --exec "/bin/bash" --max-conns 3 --allow 192.168.0.0/24 -l 8081 --keep-open
```

9. Copying files over network with netcat

Another good hack often used by hackers to copy files between 2 servers *Server1* and *Server2* who doesn't have any kind of **FTP / SCP / SFTP / SSH / SVN / GIT or any kind of Web copy support service** - i.e. servers only used as a Database systems that are behind a paranoid sysadmin firewall is **copying files between two servers with netcat.**

On Server2 (the Machine on which you want to store the file)

```
nc -lp 2323 > files-archive-to-copy.tar.gz
```

On server1 (the Machine from where file is copied) run:

```
nc -w 5 server2.example.com 2323
```

Note that **the downside of such transfers with *netcat* is data transferred is unencrypted** so any one with even a simple network sniffer or packet analyzer such as **iptraf** or **tcpdump** could capture the file, so make sure the file doesn't contain sensitive data such as passwords.

Copying partition images like that is perhaps best way to get disk images from a big server onto a NAS (when you can't plug the NAS into the server).

10. Copying piped archived directory files with netcat

On computer A:

```
export ARIBTRARY_PORT=3232  
nc -l $ARBITRARY_PORT | tar vzf -
```

On Computer B:

```
tar vzf - files_or_directories | nc computer_a $ARBITRARY_PORT
```

11. Creating a one page webserver with netcat and ncat

As netcat could listen to port and print content of a file, it can be set-up with a bit of bash shell scripting to serve as a one page webserver, or even combined with some perl scripting and bash to create a multi-serve page webserver if needed.

To make netcat serve a page to any connected client run in a **screen** / **tmux** session following code:

```
while true; do nc -l -p 80 -q 1
```

Another interesting fun example if you have installed **ncat** (is a small web server that connects current time on server on connect).

```
ncat -l -p 8080 --sh-exec 'echo -ne "HTTP/1.0 200 OK\r\n\r\nThe date is "; date;'
```

12. Cloning Hard disk partitions with netcat

rsync is a common tool used to clone hard disk partitions over network. However if **rsync** is not installed on a server and **netcat** is there you can use it instead, lets say we want to clone **/dev/sdb** from **Server1** to **Server2** assuming (**Server1** has a configured working Local or Internet connection).

On Server2 run:

```
nc -l -p 4321 | dd of=/dev/sdb
```

Following on Server2 to start the Partition / HDD cloning process run

```
dd if=/dev/sdb | nc 192.168.0.88 4321
```

Where **192.168.0.88** is the IP address listen configured on Server2 (in case you don't know it, check the listening IP to access with `/sbin/ifconfig`).

Next you have to wait for some short or long time depending on the partiiton or Hard drive, number of files / directories and allocated disk / partition size.

To clone `/dev/sda` (a main partiiton) from Server1 to Server2 first requirement is that it is not mounted, thus to have it unmounted on a system assuming you have physical access to the host, you can boot some LiveCD Linux distribution such as [Knoppix Live CD](#) on Server1, manually set-up networking with `ifconfig` or grab an IP via **DHCP** from the **central DHCP server** and repeat above example.

Happy netcating :)