DigitalOcean

Subscribe Share Contents

How To Use Netcat to Establish and Test TCP and UDP Connections on a VPS

Updated February 20, 2018 1m NETWORKING SYSTEM TOOLS

Introduction

Linux is known for having a great number of mature, useful command line utilities available out of the bo most distributions. Skilled system administrators can do much of their work using the built-in tools withoı having to install additional software.

In this guide, we will discuss how to use the **netcat** utility. Often referred to as a Swiss army knife of networking tools, this versatile command can assist you in monitoring, testing, and sending data across network connections.

We will be exploring this on an Ubuntu 12.04 VPS, but netcat should be available on almost any modern Linux distribution. Ubuntu ships with the BSD variant of netcat, and this is what we will be using in this guide. Other versions may operate differently or provide other options.

General Syntax

By default, netcat operates by initiating a TCP connection to a remote host.

The most basic syntax is:

```
$ netcat [options] host port
```

This will attempt to initiate a TCP to the defined host on the port number specified. This is basically functions similarly to the old Linux `telnet` command. Keep in mind that your connection is entirely unencrypted.

If you would like to send a UDP packet instead of initiating a TCP connection, you can use the `-u` optioı

```
$ netcat -u host port
```

You can specify a range of ports by placing a dash between the first and last:

```
$ netcat host startport-endport
```

This is generally used with some additional flags.

On most systems, we can use either `netcat` or `nc` interchangeably. They are aliases for the same command.

How To Use Netcat for Port Scanning

One of the most common uses for netcat is as a port scanner.

Although netcat is probably not the most sophisticated tool for the job (nmap is a better choice in most cases), it can perform simple port scans to easily identify open ports.

We do this by specifying a range of ports to scan, as we did above, along with the `-z` option to perform scan instead of attempting to initiate a connection.

For instance, we can scan all ports up to 1000 by issuing this command:

```
$ netcat -z -v domain.com 1-1000
```

Along with the `-z` option, we have also specified the `-v` option to tell netcat to provide more verbose information.

The output will look like this:

```
output
nc: connect to domain.com port 1 (tcp) failed: Connection refused
nc: connect to domain.com port 2 (tcp) failed: Connection refused
nc: connect to domain.com port 3 (tcp) failed: Connection refused
nc: connect to domain.com port 4 (tcp) failed: Connection refused
nc: connect to domain.com port 5 (tcp) failed: Connection refused
nc: connect to domain.com port 6 (tcp) failed: Connection refused
```

```
nc: connect to domain.com port 7 (tcp) failed: Connection refused
. . .
Connection to domain.com 22 port [tcp/ssh] succeeded!
. . .
```

As you can see, this provides a lot of information and will tell you for each port whether a scan was successful or not.

If you are actually using a domain name, this is the form you will have to use.

However, your scan will go much faster if you know the IP address that you need. You can then use the flag to specify that you do not need to resolve the IP address using DNS:

```
$ netcat -z -n -v 198.51.100.0 1-1000
```

The messages returned are actually sent to standard error (see our I/O redirection article for more info). can send the standard error messages to standard out, which will allow us to filter the results easier.

We will redirect standard error to standard output using the `2>&1` bash syntax. We will then filter the results with `grep`:

```
$ netcat -z -n -v 198.51.100.0 1-1000 2>&1 | grep succeeded
```

```
output
Connection to 198.51.100.0 22 port [tcp/*] succeeded!
```

Here, we can see that the only port open in the range of 1-1000 on the remote computer is port 22, the traditional SSH port.

How To Communicate through Netcat

Netcat is not restricted to sending TCP and UDP packets. It also can listen on a port for connections and packets. This gives us the opportunity to connect two instances of netcat in a client-server relationship.

Which computer is the server and which is the client is only a relevant distinction during the initial configuration. After the connection is established, communication is exactly the same in both directions.

On one machine, you can tell netcat to listen to a specific port for connections. We can do this by provid the `-l` parameter and choosing a port:

```
$ netcat -l 4444
```

This will tell netcat to listen for TCP connections on port 4444. As a regular (non-root) user, you will not b able to open any ports under 1000, as a security measure.

On a second server, we can connect to the first machine on the port number we choose. We do this the same way we've been establishing connections previously:

```
$ netcat domain.com 4444
```

It will look as if nothing has happened. However, you can now send messages on either side of the connection and they will be seen on either end.

Type a message and press `ENTER`. It will appear on both the local and remote screen. This works in the opposite direction as well.

When you are finished passing messages, you can press `CTRL-D` to close the TCP connection.

How To Send Files through Netcat

Building off of the previous example, we can accomplish more useful tasks.

Because we are establishing a regular TCP connection, we can transmit just about any kind of informatic over that connection. It is not limited to chat messages that are typed in by a user. We can use this knowledge to turn netcat into a file transfer program.

Once again, we need to choose one end of the connection to listen for connections. However, instead o printing information onto the screen, as we did in the last example, we will place all of the information straight into a file:

```
$ netcat -l 4444 > received_file
```

On the second computer, create a simple text file by typing:

```
$ echo "Hello, this is a file" > original_file
```

We can now use this file as an input for the netcat connection we will establish to the listening computer The file will be transmitted just as if we had typed it interactively:

```
$ netcat domain.com 4444 < original_file
```

We can see on the computer that was awaiting a connection, that we now have a new file called "received_file" with the contents of the file we typed on the other computer:

```
$ cat received_file
```

```
output
Hello, this is a file
```

As you can see, by piping things, we can easily take advantage of this connection to transfer all kinds of things.

For instance, we can transfer the contents of an entire directory by creating an unnamed tarball on-the-f transferring it to the remote system, and unpacking it into the remote directory.

On the receiving end, we can anticipate a file coming over that will need to be unzipped and extracted b typing:

```
$ netcat -l 4444 | tar xzvf -
```

The ending dash (-) means that tar will operate on standard input, which is being piped from netcat acro the network when a connection is made.

On the side with the directory contents we want to transfer, we can pack them into a tarball and then se them to the remote computer through netcat:

```
$ tar -czf - * | netcat domain.com 4444
```

This time, the dash in the tar command means to tar and zip the contents of the current directory (as specified by the * wildcard), and write the result to standard output.

This is then written directly to the TCP connection, which is then received at the other end and decompressed into the current directory of the remote computer.

This is just one example of transferring more complex data from one computer to another. Another common idea is to use the `dd` command to image a disk on one side and transfer it to a remote comput
We won't be covering this here though.

How To Use Netcat as a Simple Web Server

We've been configuring netcat to listen for connections in order to communicate and transfer files. We c
use this same concept to operate netcat as a very simple web server. This can be useful for testing page
that you are creating.

First, let's make a simple HTML file on one server:

```
$ nano index.html
```

Here is some simple HTML that you can use in your file:

index.html

```
<html>
        <head>
                <title>Test Page</title>
        </head>
        <body>
                <h1>Level 1 header</h1>
                <h2>Subheading</h2>
                <p>Normal text here</p>
        </body>
</html>
```

Save and close the file.

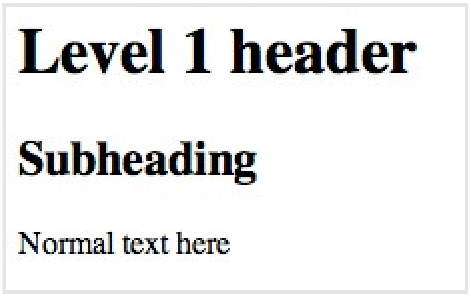
Without root privileges, you cannot serve this file on the default web port, port 80. We can choose port 8888 as a regular user.

If you just want to serve this page one time to check how it renders, you can run the following command

```
$ printf 'HTTP/1.1 200 OK\n\n%s' "$(cat index.html)" | netcat -l 8888
```

Now, in your browser, you can access the content by visiting:

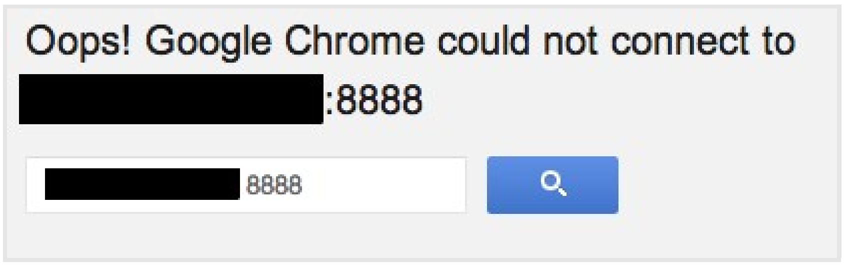
```
http://server_IP:8888
```


Level 1 header

Subheading

Normal text here

This will serve the page, and then the netcat connection will close. If you attempt to refresh the page, it \
be gone:


Oops! Google Chrome could not connect to [redacted]:8888

[redacted] 8888

We can have netcat serve the page indefinitely by wrapping the last command in an infinite loop, like thi

```
$ while true; do printf 'HTTP/1.1 200 OK\n\n%s' "$(cat index.html)" | netcat -l 8888
```

This will allow it to continue to receive connections after the first connection closes.

We can stop the loop by typing `CTRL-C` on the server.

This allows you to see how a page renders in a browser, but it doesn't provide much more functionality.
should never use this for serving actual websites. There is no security and simple things like links do not
even work correctly.

Conclusion

You should now have a pretty good idea as to what netcat can be used for. It is a versatile tool that can b useful to diagnose problems and verify that base-level functionality is working correctly with TCP/UDP connections.

Using netcat, you can communicate between different computers very easily for quick interactions. Netc attempts to make network interactions transparent between computers by taking the complexity out of forming connections.

Upvote (15) Subscribe Share

Introducing Spaces: Object Storage on DigitalOcean

A simple and cost-effective way to store, serve, backup, and archive a virtually infinite amount c media, content, images, and static files for your apps.

TRY FREE FOR 2 MONTHS

Related Tutorials

How To Use Traceroute and MTR to Diagnose Network Issues

How To Mirror Local and Remote Directories on a VPS with lsyncd

How To Create a High Availability Setup with Heartbeat and Floating IPs on Ubuntu 16.04

How To Create a Point-To-Point VPN with WireGuard on Ubuntu 16.04

How to Set Up and Use LXD on Ubuntu 16.04

6 Comments

Leave a comment...

Log In to Comment

127wexfordroad *July 22, 2014*

1 Amazing. I'd heard of netcat and wanted to get started with it but did not know where to begin. This is a perfect intro, thanks!

helder.mc *August 7, 2014*

0 Good article, but you have formatting issues from somewhere in "How To Send Files through Netcat".

kamaln7 **MOD** *August 7, 2014*

0 @helder.nc: Thanks for catching that! I've fixed it.

jasonrhaas *September 14, 2015*

0 great intro to `netcat`, thanks! I use `nmap` for doing port scanning, but its nice to be able to do it with `netcat` as well, since `nmap` is often not included in the base linux packages.

SureshDH *August 26, 2016*

0 nice article you covered lot of variants, can you please also show an example to listen(nc -lu <port_num>) specific udp port. That could be helpful for the first readers.

jemstar89 *March 25, 2017*

Nice article. One point worth mentioning.

0 When using netcat as a simple web server, you have this command written:
netcat -l 8888 < index.html

You actually need to specify the port in the command, or else it won't work. Try this instead.
netcat -lp 8888 < index.html
I found this solution to work.

Keep up the good work!

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Copyright © 2018 DigitalOcean™ Inc.