

Networking

Networking

- When it comes to networking, there is probably nothing that cannot be done with Linux.
- Linux is used to build all sorts of networking systems and appliances, including firewalls, routers, name servers, network-attached storage (NAS) boxes.
- This chapter will cover the following commands:
 - ping – Send an ICMP ECHO_REQUEST to network hosts
 - traceroute – Print the route packets trace to a network host
 - ip – Show / manipulate routing, devices, policy routing and tunnels
 - netstat – Print network connections, routing tables, interface statistics, masquerade connections, and multicast memberships
 - ftp – Internet file transfer program
 - curl - Transfer a URL
 - wget – Non-interactive network downloader
 - ssh – OpenSSH SSH client (remote login program)

Examining and Monitoring a Network

- Ping: The most basic network command is ping. The ping command sends a special network packet called an ICMP ECHO_REQUEST to a specified host. Most network devices receiving this packet will reply to it, allowing the network connection to be verified.

`ping linuxcommand.org`

```
[me@linuxbox ~]$ ping linuxcommand.org
PING linuxcommand.org (66.35.250.210) 56(84) bytes of data.
64 bytes from vhost.sourceforge.net (66.35.250.210): icmp_seq=1
ttl=43 time=107 ms
64 bytes from vhost.sourceforge.net (66.35.250.210): icmp_seq=2
ttl=43 time=108 ms
64 bytes from vhost.sourceforge.net (66.35.250.210): icmp_seq=3
ttl=43 time=106 ms
64 bytes from vhost.sourceforge.net (66.35.250.210): icmp_seq=4
ttl=43 time=106 ms
64 bytes from vhost.sourceforge.net (66.35.250.210): icmp_seq=5
ttl=43 time=105 ms
64 bytes from vhost.sourceforge.net (66.35.250.210): icmp_seq=6
ttl=43 time=107 ms

--- linuxcommand.org ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 6010ms
rtt min/avg/max/mdev = 105.647/107.052/108.118/0.824 ms
```

traceroute

- The traceroute program (some systems use the similar tracepath program instead) lists all the “hops” network traffic takes to get from the local system to a specified host.
- For example, to see the route taken to reach slashdot.org, we would do this:

```
traceroute slashdot.org
```

```
traceroute to slashdot.org (216.34.181.45), 30 hops max, 40 byte
packets
 1  ipcop.localdomain (192.168.1.1)  1.066 ms  1.366 ms  1.720 ms
 2  * * *
 3  ge-4-13-ur01.rockville.md.bad.comcast.net (68.87.130.9)  14.622
ms  14.885 ms  15.169 ms
 4  po-30-ur02.rockville.md.bad.comcast.net (68.87.129.154)  17.634
ms  17.626 ms  17.899 ms
 5  po-60-ur03.rockville.md.bad.comcast.net (68.87.129.158)  15.992
ms  15.983 ms  16.256 ms
 6  po-30-ar01.howardcounty.md.bad.comcast.net (68.87.136.5)  22.835
ms  14.233 ms  14.405 ms
 7  po-10-ar02.whitemarsh.md.bad.comcast.net (68.87.129.34)  16.154
ms  13.600 ms  18.867 ms
 8  te-0-3-0-1-cr01.philadelphia.pa.ibone.comcast.net (68.86.90.77)
21.951 ms  21.073 ms  21.557 ms
```

- In the output, we can see that connecting from our test system to `slashdot.org` requires traversing 16 routers.
- For routers that provided identifying information, we see their hostnames, IP addresses, and performance data, which includes three samples of round-trip time from the local system to the router.
- For routers that do not provide identifying information (because of router configuration, network congestion, firewalls, etc.), we see asterisks as in the line for hop number 2.
- In cases where routing information is blocked, we can sometimes overcome this by adding either the `-T` or `-I` option to the `traceroute` command.

```

9 pos-0-8-0-0-cr01.newyork.ny.ibone.comcast.net (68.86.85.10)
22.917 ms 21.884 ms 22.126 ms
10 204.70.144.1 (204.70.144.1) 43.110 ms 21.248 ms 21.264 ms
11 cr1-pos-0-7-3-1.newyork.savvis.net (204.70.195.93) 21.857 ms
cr2-pos-0-0-3-1.newyork.savvis.net (204.70.204.238) 19.556 ms cr1-
pos-0-7-3-1.newyork.savvis.net (204.70.195.93) 19.634 ms
12 cr2-pos-0-7-3-0.chicago.savvis.net (204.70.192.109) 41.586 ms
42.843 ms cr2-tengig-0-0-2-0.chicago.savvis.net (204.70.196.242)
43.115 ms
13 hr2-tengigabitethernet-12-1.elkgrovech3.savvis.net
(204.70.195.122) 44.215 ms 41.833 ms 45.658 ms
14 csr1-ve241.elkgrovech3.savvis.net (216.64.194.42) 46.840 ms
43.372 ms 47.041 ms
15 64.27.160.194 (64.27.160.194) 56.137 ms 55.887 ms 52.810 ms
16 slashdot.org (216.34.181.45) 42.727 ms 42.016 ms 41.437 ms

```

ip

- The `ip` program is a multi-purpose network configuration tool that makes use of the full range of networking features available in modern Linux kernels.
- It replaces the earlier and now deprecated `ifconfig` program.
- The `ip` program is used to examine various network settings and statistics. Through the use of its many options, we can look at a variety of features in our network setup. With `ip`, we can examine a system's network interfaces and routing table.
- In the example above, we see that our test system has three network interfaces.
- The first, called `lo`, is the loopback interface, a virtual interface that the system uses to “talk to itself”, the second, called `enp1s0`, is an Ethernet interface (`en` = Ethernet), and the third called `wlp2s0` is a wireless interface (`wl` = wireless)..

```
[me@linuxbox ~]$ ip address show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    group default qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host noprefixroute
            valid_lft forever preferred_lft forever
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
    group default qlen 1000
        link/ether 00:26:6c:26:67:bf brd ff:ff:ff:ff:ff:ff
        inet 192.168.1.223/24 brd 192.168.50.255 scope global dynamic
            noprefixroute enp1s0
            valid_lft 82366sec preferred_lft 82366sec
        inet6 fe80::226:6cff:fe26:67bf/64 scope link noprefixroute
            valid_lft forever preferred_lft forever
3: wlp2s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq
    state DOWN group default qlen 1000
        link/ether 06:8a:59:f4:f6:d3 brd ff:ff:ff:ff:ff:ff permaddr
        24:ec:99:46:aa:1e
```

- When performing casual network diagnostics, the important things to look for are the presence of the phrase `state UP` in the first line for the interface, indicating that it is enabled, and the presence of a valid IP address in the `inet` field on the third line.
- For systems using Dynamic Host Configuration Protocol (DHCP), a valid IP address in this field will verify that the DHCP is working.
- The routing table,

```
[me@linuxbox ~]$ ip route show
default via 192.168.1.1 dev enp1s0 proto dhcp src 192.168.50.223
metric 100
169.254.0.0/16 dev enp1s0 scope link metric 1000
192.168.1.0/24 dev enp1s0 proto kernel scope link src 192.168.1.223
metric 100
```

- In this simple example, we see a typical routing table for a client machine on a local area network (LAN) behind a firewall/router. IP addresses that end in zero refer to networks rather than individual hosts, so this destination means any host on the LAN can be reached directly. We see two networks listed: 169.254.0.0 and 192.168.1.0.
- Now, the 192.168.1.0 network is our LAN, but what is 169.254.0.0? Well, that's a piece of network trickery called Automatic Private IP Addressing (APIPA). It is used to automatically assign an IP address when a DHCP server is unavailable.

- The first line contains the destination default. This means send any traffic destined for a network that is not otherwise listed in the table to this address. In our example, we see that the default gateway is defined as a router with the address of 192.168.1.1, (a typical address for a home router) which presumably knows what to do with the destination traffic.
- The ip command is a complicated program with many options and commands. The command syntax consists of:
`ip [-options] object [command]`
- In the previous examples we used the objects address and route with the command show. As a convenience, ip allows object names to be shortened to a single character and since the default command is show, we can shorten the commands to ip a and ip r and get identical results.

Transporting Files Over a Network

ftp

- One of the true “classic” programs, ftp gets its name from the protocol it uses, the File Transfer Protocol. FTP was once the most widely used method of downloading files over the Internet. Some web browsers still support it, and we may see URLs starting with the protocol `ftp://`.
- Before there were web browsers, there was the `ftp` program. `ftp` is used to communicate with FTP servers, machines that contain files that can be uploaded and downloaded over a network.
- FTP (in its original form) is not secure because it sends account names and passwords in cleartext. This means they are not encrypted and anyone sniffing the network can see them.
- Because of this, almost all FTP done over the Internet is done by anonymous FTP servers. An anonymous server allows anyone to log in using the login name “anonymous” and a meaningless password.

- In the example below, we show an imaginary session with the **ftp** program downloading an Ubuntu ISO image located in the **/pub/cd_images/Ubuntu-24.04** directory of the anonymous FTP server **fileserver**:

```
[me@linuxbox ~]$ ftp fileserver
Connected to fileserver.localdomain.
220 (vsFTPd 2.0.1)
Name (fileserver:me): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd pub/cd_images/Ubuntu-24.04
250 Directory successfully changed.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-rw-r--    1 500      500    733079552 Apr 25 03:53 ubuntu-
24.04-desktop-amd64.iso
226 Directory send OK.
ftp> lcd Desktop
Local directory now /home/me/Desktop
ftp> get ubuntu-24.04-desktop-amd64.iso
local: ubuntu-24.04-desktop-amd64.iso remote: ubuntu-24.04-desktop-
amd64.iso
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for ubuntu-24.04-desktop-
amd64.iso (733079552 bytes).
226 File send OK.
733079552 bytes received in 68.56 secs (10441.5 kB/s)
ftp> bye
```

- The table provides an explanation of the commands entered during this session.
- Typing `help` at the `ftp>` prompt will display a list of the supported commands.
- Using `ftp` on a server where sufficient permissions have been granted, it is possible to perform many ordinary file management tasks

Command	Meaning
<code>ftp fileserver</code>	Invoke the <code>ftp</code> program and have it connect to the FTP server <code>fileserver</code> .
<code>anonymous</code>	Login name. After the login prompt, a password prompt will appear. Some servers will accept a blank password; others will require a password in the form of an email address. In that case, try something like <code>user@example.com</code> .
<code>cd pub/cd_images/Ubuntu-24.04</code>	Change to the directory on the remote system containing the desired file. Note that on most anonymous FTP servers, the files for public downloading are found somewhere under the <code>pub</code> directory.
<code>ls</code>	List the directory on the remote system.
<code>lcd Desktop</code>	Change the directory on the local system to <code>~/Desktop</code> . In the example, the <code>ftp</code> program was invoked when the working directory was <code>~</code> . This command changes the working directory to <code>~/Desktop</code> .
<code>get ubuntu-24.04-desktop-amd64.iso</code>	Tell the remote system to transfer the file <code>ubuntu-24.04-desktop-amd64.iso</code> to the local system. Since the working directory on the local system was changed to <code>~/Desktop</code> , the file will be downloaded there.
<code>bye</code>	Log off the remote server and end the <code>ftp</code> program session. The commands <code>quit</code> and <code>exit</code> may also be used.

lftp – A Better ftp

- ftp is not the only command-line FTP client. In fact, there are many. One of the better (and more popular) ones is lftp by Alexander Lukyanov.
- It works much like the traditional ftp program but has many additional convenience features including multiple-protocol support (including HTTP), automatic retry on failed downloads, background processes, tab completion of path names, and many more.

curl - Transfer a URL

- Another popular file transfer program is `curl`. Its most basic usage works like this:
`curl https://linuxcommand.org`
- We specify a URL and `curl` downloads the first page of the URL and outputs it to standard output. Multiple URLs can be specified.
- `curl` supports most network protocols including HTTP, HTTPS, FTP, IMAP, POP3, SFTP, SMB, and others. The table lists a few of the many options that `curl` supports.

Option	Description
<code>-o, --output <i>file</i></code>	Send output to the specified <i>file</i> rather than standard output.
<code>-O, --remote-name</code>	Like <code>-o</code> but name local file the same as the name of the remote file.
<code>-s, --silent</code>	Suppress the progress meter and error messages.
<code>-u, --proxy-user <i>user:password</i></code>	Specify a user name/password combination.
<code>-v, --verbose</code>	Display verbose messages as it executes.

wget - Non-interactive network downloader

- It is useful for downloading content from both web and FTP sites. Single files, multiple files, and even entire sites can be downloaded. To download the first page of linuxcommand.org we could do this:

```
[me@linuxbox ~]$ wget http://linuxcommand.org/index.php
--11:02:51--  http://linuxcommand.org/index.php
              => `index.php'
Resolving linuxcommand.org... 66.35.250.210
Connecting to linuxcommand.org|66.35.250.210|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]

[ <=>                               ] 3,120          --.-K/s

11:02:51 (161.75 MB/s) - `index.php' saved [3120]
```

Secure Communication with Remote Hosts

- In the early days, before the general adoption of the Internet, there were a couple of popular programs used to log in to remote hosts.
- These were the `rlogin` and `telnet` programs. These programs, however, suffer from the same fatal flaw that the `ftp` program does; they transmit all their communications (including login names and passwords) in cleartext.
- This makes them wholly inappropriate for use in the Internet age.

ssh - Secure Shell (SSH)

- To address this problem, a new protocol called Secure Shell (SSH) was developed. SSH solves the two basic problems of secure communication with a remote host.
 1. It authenticates that the remote host is who it says it is (thus preventing so-called man-in-the-middle attacks).
 2. It encrypts all of the communications between the local and remote hosts.
- SSH consists of two parts. An SSH server runs on the remote host, listening for incoming connections, by default, on port 22, while an SSH client is used on the local system to communicate with the remote server.
- Most Linux distributions ship an implementation of SSH called OpenSSH from the OpenBSD project. Some distributions include both the client and the server packages by default, while others only supply the client.
- To enable a system to receive remote connections, it must have the `OpenSSH-server` package installed, configured and running, and (if the system either is running or is behind a firewall) it must allow incoming network connections on TCP port 22.

- To connect to a remote host named `remote-sys`, we would use the `ssh` client program like so:

```
[me@linuxbox ~]$ ssh remote-sys
The authenticity of host 'remote-sys (192.168.1.4)' can't be
established.
RSA key fingerprint is
41:ed:7a:df:23:19:bf:3c:a5:17:bc:61:b3:7f:d9:bb.
Are you sure you want to continue connecting (yes/no)?
```

- The first time the connection is attempted, a message is displayed indicating that the authenticity of the remote host cannot be established. This is because the client program has never seen this remote host before.
- To accept the credentials of the remote host, enter “yes” when prompted. Once the connection is established, the user is prompted for a password:

```
Warning: Permanently added 'remote-sys,192.168.1.4' (RSA) to the list
of known hosts.
me@remote-sys's password:
```

- After the password is successfully entered, we receive the shell prompt from the remote system

```
Last login: Sat Aug 30 13:00:48 2024
[me@remote-sys ~]$
```

- The remote shell session continues until the user enters the exit command at the remote shell prompt, thereby closing the remote connection. At this point, the local shell session resumes, and the local shell prompt reappears.
- It is also possible to connect to remote systems using a different username. For example, if the local user “me” had an account named “bob” on a remote system, user me could log in to the account bob on the remote system as follows:

```
[me@linuxbox ~]$ ssh bob@remote-sys
bob@remote-sys's password:
Last login: Sat Aug 30 13:03:21 2024
[bob@remote-sys ~]$
```

As stated earlier, ssh verifies the authenticity of the remote host. If the remote host does not successfully authenticate, the following message appears:

- This message is caused by one of two possible situations. First, an attacker may be attempting a man-in-the-middle attack.
 - This is rare, since everybody knows that ssh alerts the user to this. The more likely culprit is that the remote system has been changed somehow; for example, its operating system or SSH server has been reinstalled. In the interests of security and safety, however, the first possibility should not be dismissed out of hand.
 - Always check with the administrator of the remote system when this message occurs.

```
[me@linuxbox ~]$ ssh remote-sys
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!
@ IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle
attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ECDSA key sent by the remote host is
SHA256:9uh0XHzPz0AjIs0ZDanNYfv7ksU04Mjy5pR4kUTSKkA.
Please contact your system administrator.
Add correct host key in /home/me/.ssh/known_hosts to get rid of this
message.
Offending ECDSA key in /home/me/.ssh/known_hosts:42
  remove with:
    ssh-keygen -f "/home/me/.ssh/known_hosts" -R "remote-sys"
Host key for remote-sys has changed and you have requested strict
checking.
Host key verification failed.
```

- After it has been determined that the message is because of a benign cause, it is safe to correct the problem on the client side. This is done by using the suggestion provided by the warning message:

```
ssh-keygen -f "/home/me/.ssh/known_hosts" -R "remote-sys"
```

- Failing that, we can use a text editor (vim perhaps) to remove the obsolete key from the `~/.ssh/known_hosts` file.
- In the example message above, we see this:

```
offending key in /home/me/.ssh/known_hosts:42
```

- This means that the 42nd line of the `known_hosts` file contains the offending key.
- Delete this line from the file, and the ssh program will be able to accept new authentication credentials from the remote system.

- Besides opening a shell session on a remote system, ssh allows us to execute a single command on a remote system.
- For example, to execute the free command on a remote host named remote-sys and have the results displayed on the local system, use this:

```
[me@linuxbox ~]$ ssh remote-sys free
me@twin4's password:
              total        used        free      shared  buffers  cached
Mem:       775536      507184      268352          0     110068    154596
          -/+ buffers/cache:  242520      533016
Swap:      1572856          0     1572856
[me@linuxbox ~]$
```

- It's possible to use this technique in more interesting ways, such as the following example in which we perform an ls on the remote system and redirect the output to a file on the local system:

```
[me@linuxbox ~]$ ssh remote-sys 'ls *' > dirlist.txt
me@twin4's password:
[me@linuxbox ~]$
```

- Notice the use of the single quotes in the command above. This is done because we do not want the pathname expansion performed on the local machine; rather, we want it to be performed on the remote system. Likewise, if we had wanted the output redirected to a file on the remote machine, we could have placed the redirection operator and the file-name within the single quotes.

```
[me@linuxbox ~]$ ssh remote-sys 'ls * > dirlist.txt'
```

scp and sftp

- The OpenSSH package also includes two programs that can make use of an SSH-encrypted tunnel to copy files across the network.
- The first, scp (secure copy) is used much like the familiar cp program to copy files.
- The most notable difference is that the source or destination pathnames may be preceded with the name of a remote host, followed by a colon character.
- For example, if we wanted to copy a document named document.txt from our home directory on the remote system, remote-sys, to the current working directory on our local system, we could do this:

```
[me@linuxbox ~]$ scp remote-sys:document.txt .
me@remote-sys's password:
document.txt                                100% 5581      5.5KB/s   00:00
[me@linuxbox ~]$
```

- As with ssh, you may apply a username to the beginning of the remote host's name if the desired remote host account name does not match that of the local system.
`scp bob@remote-sys:document.txt .`
- The second SSH file-copying program is sftp which, as its name implies, is a secure replacement for the ftp program. sftp works much like the original ftp program that we used earlier; however, instead of transmitting everything in cleartext, it uses an SSH encrypted tunnel.
- sftp has an important advantage over conventional ftp in that it does not require an FTP server to be running on the remote host. It requires only the SSH server.
- This means that any remote machine that can connect with the SSH client can also be used as an FTP-like server. Here is a sample session:

```
[me@linuxbox ~]$ sftp remote-sys
Connecting to remote-sys...
me@remote-sys's password:
sftp> ls
ubuntu-8.04-desktop-i386.iso
sftp> lcd Desktop
sftp> get ubuntu-8.04-desktop-i386.iso
Fetching /home/me/ubuntu-8.04-desktop-i386.iso to ubuntu-8.04-
desktop-i386.iso
/home/me/ubuntu-8.04-desktop-i386.iso 100% 699MB 7.4MB/s 01:35
sftp> bye
```

An SSH Client for Windows?

- Let's say you are sitting at a Windows machine but you need to log in to your Linux server and get some real work done; what do you do? Get an SSH client program for your Windows box, of course!
- The most popular one is probably PuTTY by Simon Tatham and his team.
- The PuTTY program displays a terminal window and allows a Windows user to open an SSH (or telnet) session on a remote host. The program also provides analogs for the scp and sftp programs.
- PuTTY is available at
<http://www.chiark.greenend.org.uk/~sgtatham/putty/>