NUK Documentation

VNC Desktop

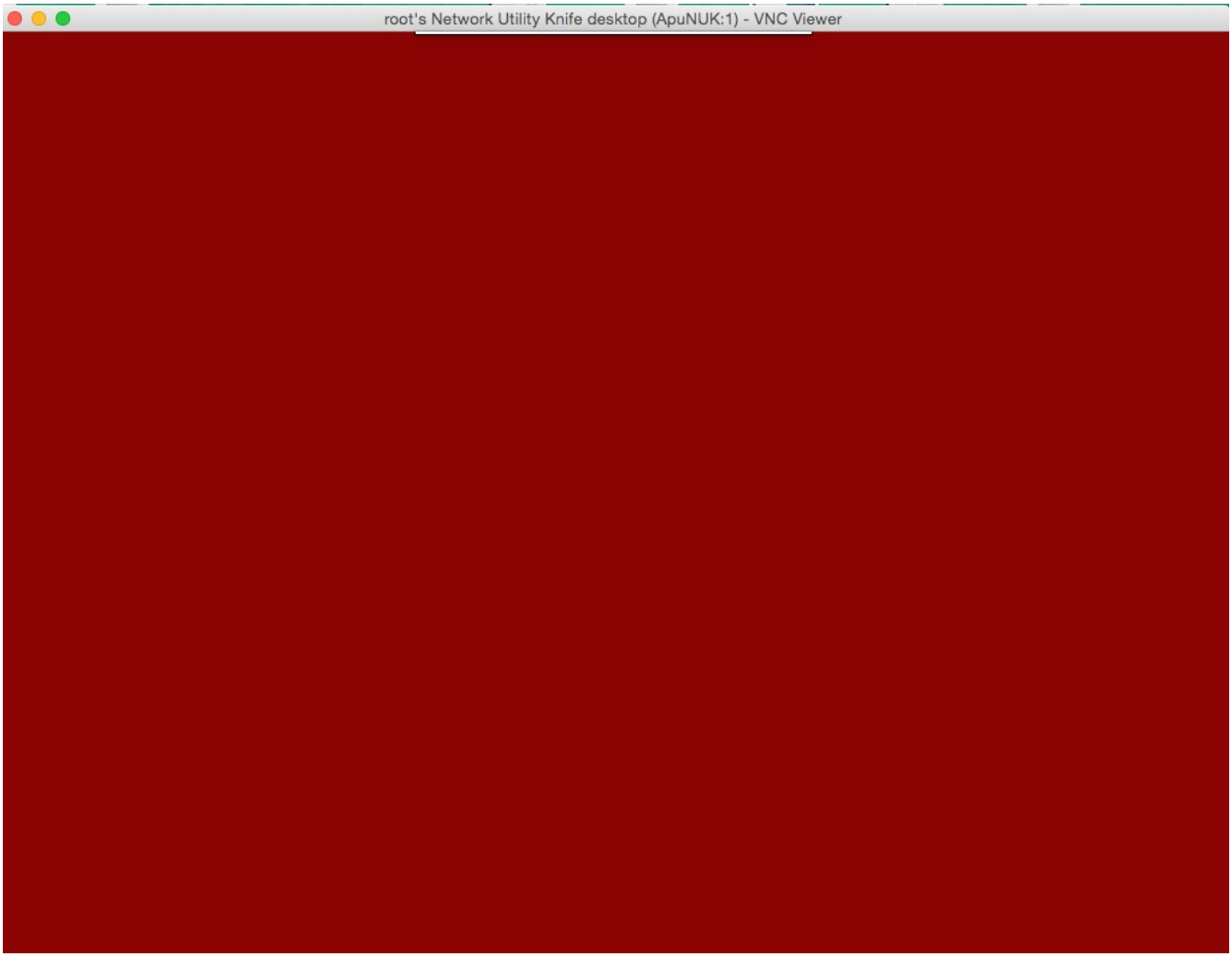
After configuring the NUK with an IP address for your network and a VNC password, the next step is to connect to the VNC desktop. A good VNC client is available at http://www.realvnc.com. You can just use the free one, if you want encryption, use stunnel. By default, the VNC port is not firewalled, but stunnel is configured to tunnel VNC on port 48227. The stunnel configuration to access VNC and the proxy server (tinyproxy) via port 48228. A sample stunnel configuration is below:

```
client = yes

; Service-level configuration

[vnc-nuk]
client = yes
accept  = 5901
connect = <NUK IP>:48227

[proxy-nuk]
client = yes
accept  = 8801
connect = <NUK IP>:48228
```

If you configure stunnel on your computer using the above example, you would then VNC to 127.0.0.1:1. You could access the proxy server by using 127.0.0.1:8801. If you want to use stunnel, you can use the sample firewall configuration (ipf-sample.conf) that's located in the /config directory. It would be a good idea to test it first (**Menu Choice D, submenu choice F**), then if it works to your satisfaction, rename it to ipf.conf and reboot.

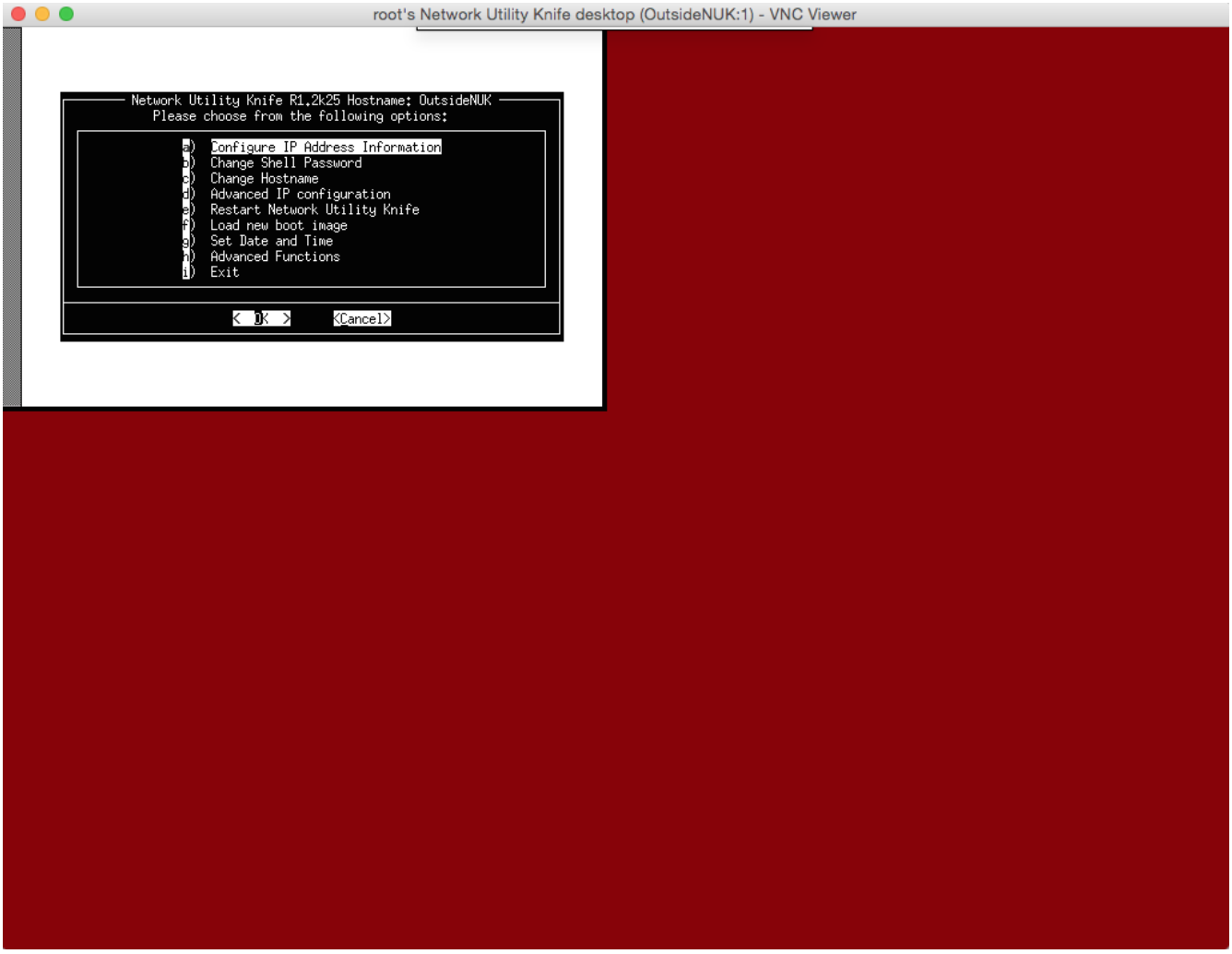
If you want more documentation on the firewall rules, you can check out the IP filter FAQ at `https://www.phildev.net/ipf`

Once you've connected to the VNC desktop, here's what you'll see:



Doesn't look like much does it? To bring up the VNC menu, hold down the Ctrl and Shift keys and hold the left mouse button. Now you'll see the menu:

Once the menu appears you can let go of the Ctrl and Shift keys, but you need to hold down the mouse button. When you let go off the mouse button, whatever's highlighted is selected. So if you let go of the mouse button when Mainmenu is selected, you'll see the familiar menu:



The same menu that you used to configure the device initially now appears in the upper right hand corner. Before we go through the other menu choices, here are a few other tools that you can use to manipulate windows:

First, if you hold down the second mouse button this menu appears:

With the mouse button held down, move and select from the last three choices (the first two don't do anything). This is what they do:

Move

This allows you to move a window. Move the mouse pointer down to Move, let go of the second mouse button. Then the mouse pointer changes to look like a bullseye. ##this:

insert picture##

Then move your mouse to the window you want to move (anywhere in the window is fine), hold down the second mouse button (the mouse pointer will change to a small square) and drag the window (this is counter-intuitive, we know). You will be dragging the window by it's upper-left corner, so it will automatically jump based on where your cursor was when you picked up the window (unless of course, you picked up the window from the upper-left corner).

Delete

Just like move, except when you right click on the window, it closes.

Hide

Just like delete, except the window is hidden. The window is added to the second mouse button menu and by selecting it, it reappears. Here's what the menu looks like with a hidden window.

In this instance we hid iPerf running as a server.

Third Mouse Button

If you have a three button mouse, you can access up to four screens. Hold down the third button (or mouse wheel). ##Then you'll see a menu like this:

##

A menu of four desktops appears. Move the cursor down to whichever desktop you want to see. The desktops are completely independent of one another. Also, using the Hide feature above, you can move a window from one desktop to another.

You can use the hide feature to move a window from one desktop to another. Hide the window you want to move, switch desktops, and un-hide it.

VNC Menu – continued.

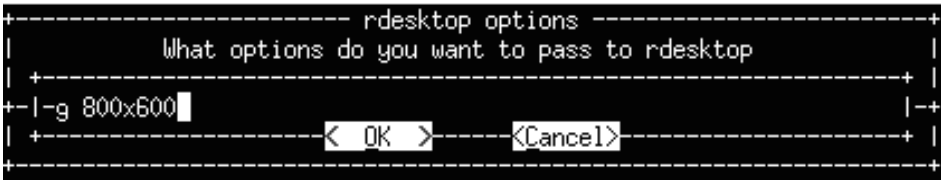
We will now go through the other menu choices.

VNC Menu – Minicom

This option is not available on the current hardware platform.

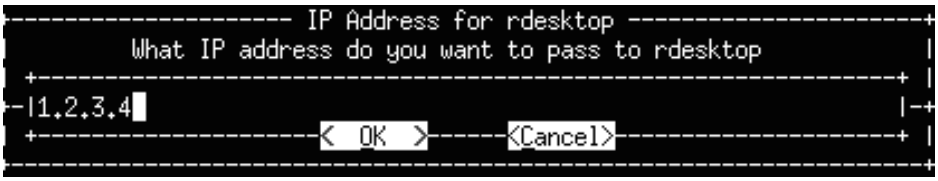
VNC Menu – Rdesktop (Windows Remote Desktop)

After selecting Remote Desktop, two menu choices will appear:

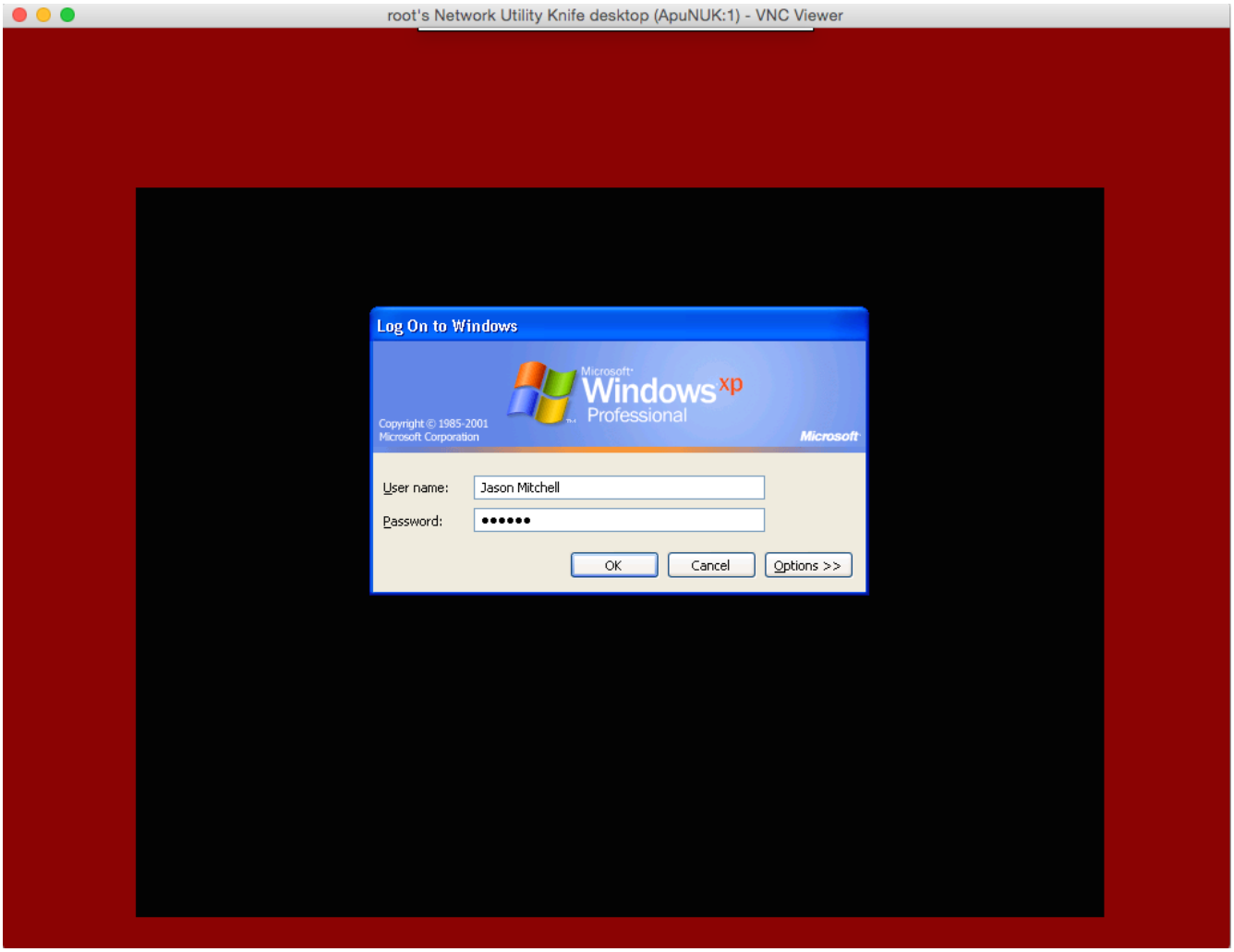


This is the default and it tells Rdesktop to make the window 800x600. You can try making this bigger, but remember that the VNC window itself is only 1024 x 768.

The next option specifies the IP address to connect to:



When Rdesktop makes a connection, it looks like this:



When you log out the window will close. If Rdesktop is unable to make a connection to the computer (usually because the computer is running a newer version of the RDP protocol), you'll see a window flash open, then disappear. In this instance we recommend to use rinetd.

If you have a machine running a newer version of the RDP protocol then our Rdesktop client supports, create a file named rinetd.conf as follows:

0.0.0.0 3389 <Remote Desktop Device IP> 3389

This says listen on port 3389, then when a connection is made, connect to port 3389 on the Remote Desktop Device IP and pass the data between the two connections. It's as if the outside user is connecting directly to the Remote Desktop Device.

Upload this file it to /config if using SFTP (/ (root file system) if using FTP. Start rinetd using Start Services, then Remote Desktop to the NUK IP address.

For VNC, create rinetd.conf as follows:

0.0.0.0 5900 <VNC IP> 5901

(Port 5900 is in use on the NUK)

Upload this file it to /config if using SFTP (/ (root file system) if using FTP. Start rinetd using Start Services, then VNC to the NUK IP address:1 (Desktop 1). The NUK display runs on Desktop 0

For Citrix:

0.0.0.0 1494 <Citrix IP> 1494

And connect with your Citrix client to the NUK IP address.

VNC Menu – Ntop/Nprobe

Nprobe is disabled pending licensing negotiations, but an older text only version of ntop is available. It can be run to listen on either monitor interface and reports basic IP info, such as packets sent and device throughput. For ntop to work, the switch port that monitor 1 or 2 is connected to must be a mirror or monitoring port.

Here's an example of what Ntop looks like:

```
           ntop v.1.1 ST [i386-unknown-netbsdelf1.6.2] listening on re0
4064 Pkts/269.5 Kb [IP 38.1 Kb/Other 231.4 Kb]         Thpt: 2.3 Kbps/4.3 Kbps
 Host                     Act   -Rcvd-      Sent       TCP       UDP     ICMP
 01:80:C2:00:00:00         R   27.2 Kb         0         0         0        0
 <West Coast>              R    5.3 Kb    4.3 Kb       662       451   4.2 Kb
 01:00:0C:CC:CC:CC         R    2.4 Kb         0         0         0        0
 <West Coast 2>            S    2.1 Kb    4.1 Kb         0         0   2.1 Kb
 <West Coast - GW>         S    2.1 Kb   85.5 Kb         0         0   2.1 Kb
 01:80:C2:00:00:0E         R    1.4 Kb         0         0         0        0
 00:26:99:4B:C6:C0         S         0   67.8 Kb         0         0        0
 A4:56:30:B5:49:56         S         0   31.0 Kb         0         0        0
 91.230.47.38              S         0       120         0         0        0
 222.185.236.154           S         0        60         0         0        0
```

Ntop wasn't connected to a monitoring port, so that's why the throughput numbers are so low.

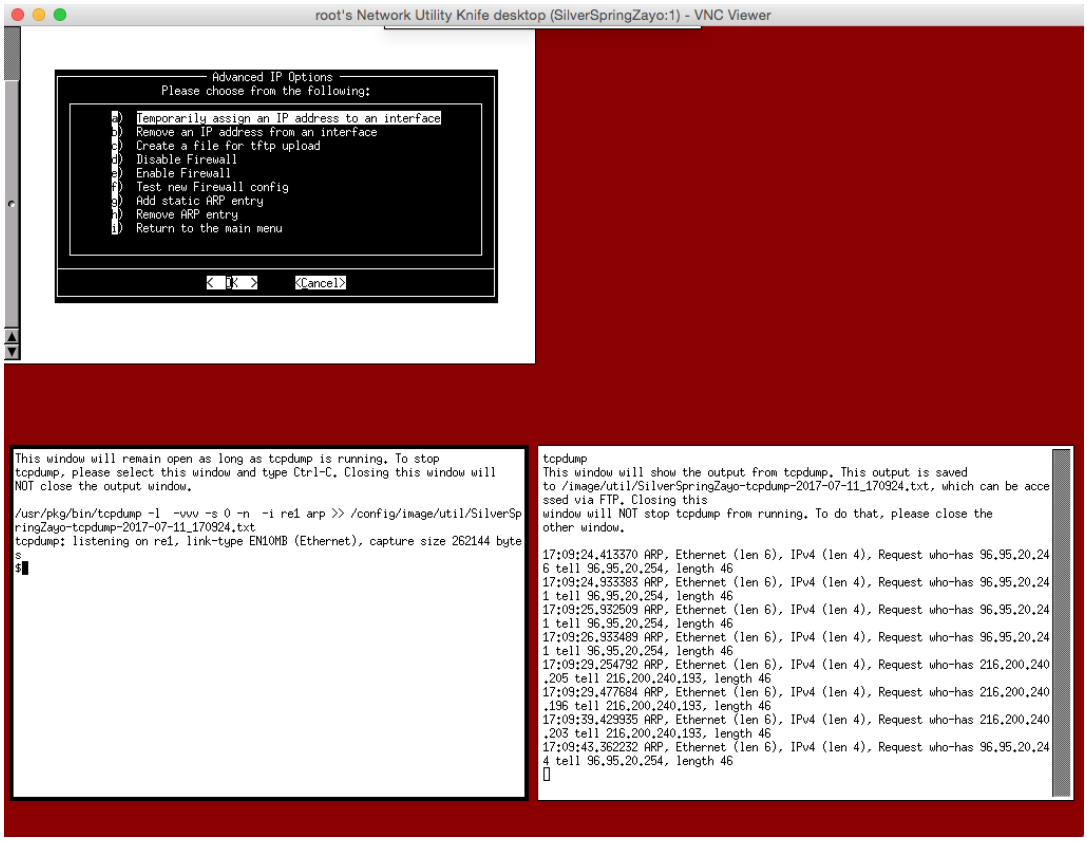
VNC Menu – Net-tools

Net-tools contains a set up packet capturing programs. The most commonly used tools are tcpdump (in live decode mode) and tcpdump, listed as “cap”, (in capture to a file mode).

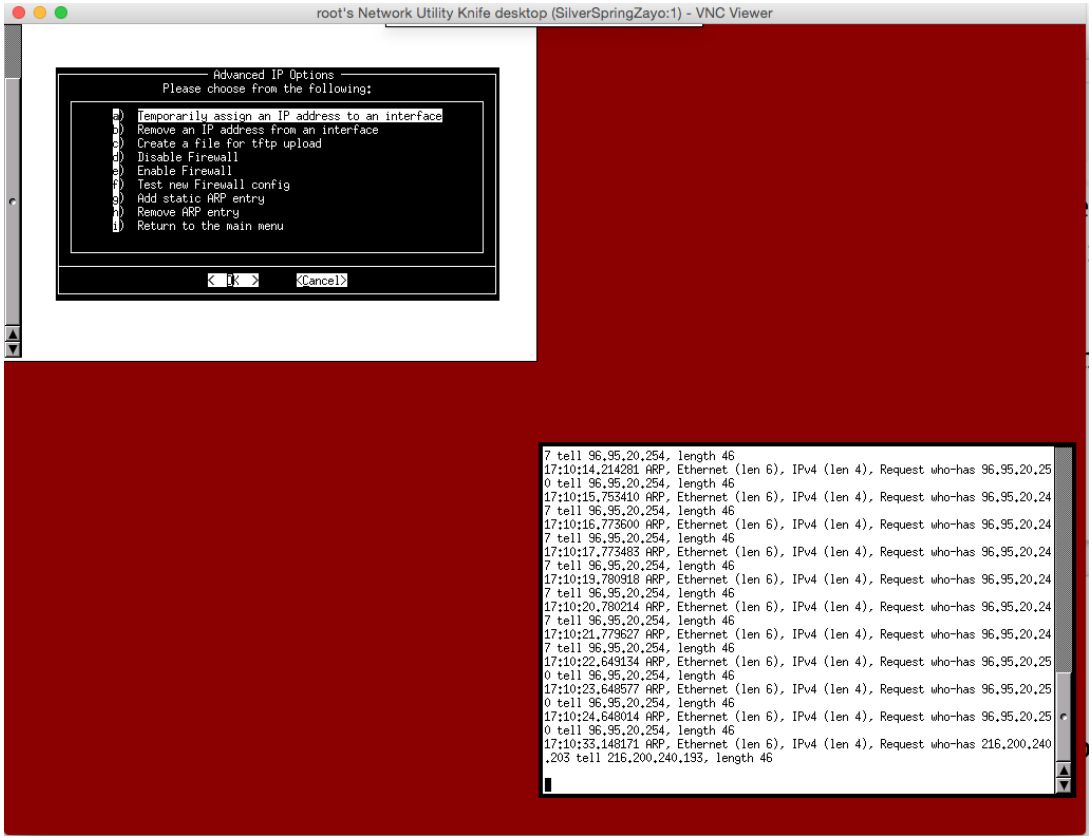
Whatever of the four utilities are used, two windows open up: a control window on the left and an output window on the right. This is so you can stop packet capture while being able to review the captured packets at your leisure.

If you want to stop packet capturing, select the left window (left bottom) with the first mouse button and press Ctrl-C. The window should close. If it doesn't you can use Delete from the second mouse button menu. Now you can review the packet capturing output on the right window (bottom right) without worrying about unnecessary packets being captured. For how ever long you run the packet capture, the output is stored in /config/storage/util/ in a file called %NUK-NAME%-%util-name%-%full-date%.txt

Here's an example of packet capture in progress:



(note the two windows on the bottom). Closing the left window still allows you to review the packet capture on the right:



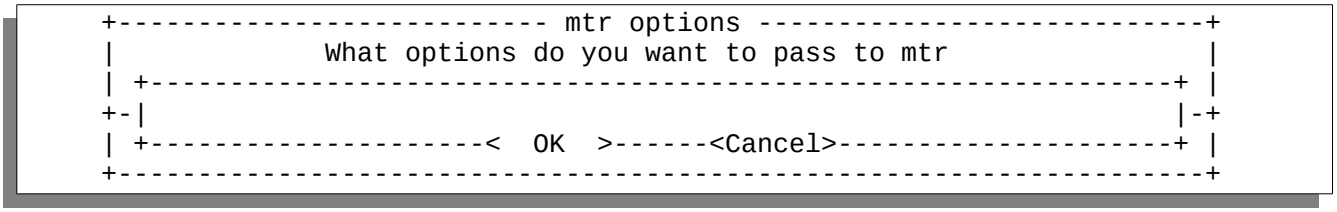
Note the scroll bar on the right, or you can scroll with the scroll wheel on a mouse that has one.

Net-tools2

Net-tools2 (from the VNC menu) is a collection of active utilities that probe network devices, measure available bandwidth, and trace a network path, among other things. All of the output from these utilities is available in /config/storage/util, with the same %NUK-NAME%-%util-name%-%full-date%.txt file naming scheme.

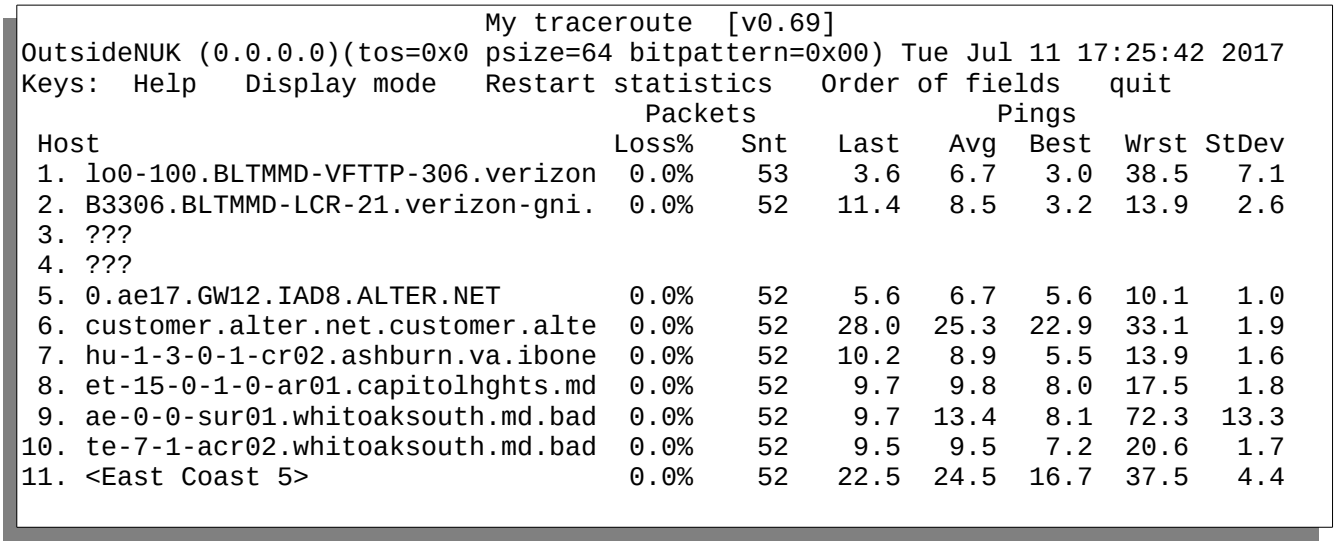
MTR

MTR is a continuously updating traceroute. It can show you what router hops are likely to be dropping packets and what router hops are slow. When you first start MTR you get an options window:



For MTR, you don't need to pass any options. If you want info on all the options available, the output from the man page is included in the appendix.

Here's an example of MTR:



You can see that hop number 6 is the slowest (most likely because it's the place Comcast and Verizon meet. It's the slowest because it has the slowest average time, slowest best time, and slowest worst time. But no router along the path is dropping packets (that's the 0.0% in the Loss column).

NBTSCAN

NBTSCAN is a utility that lists the Netbios names associated with a Windows machine. Unfortunately, Windows firewall prevents this utility from working in many cases. When you start nbtscan, you'll be asked for options. You don't need to enter any. The next dialog box asks for an IP address. You can enter an address or a network range (1.2.3.0/24 will scan from 1.2.3.1 through 1.2.3.254). Here are some examples:

```
cat: /usr/share/options/nbtscan: No such file or directory
Tue Jul 11 20:06:49 GMT 2017
Doing NBT name scan for addresses from <EC4 - FW>.71

IP address       NetBIOS Name     Server    User             MAC address
-------------------------------------------------------------------------------
<EC4 - FW>.71    JM5              <server>  <unknown>        00-0c-29-3a-de-41
Tue Jul 11 20:06:50 GMT 2017
```

```
cat: /usr/share/options/nbtscan: No such file or directory
Tue Jul 11 20:07:32 GMT 2017
Doing NBT name scan for addresses from <EC4 - FW>.73

IP address       NetBIOS Name     Server    User             MAC address
-------------------------------------------------------------------------------
<EC4 - FW>.73    CRAIGFERGUSON    <server>  <unknown>        00-0c-29-17-ed-cb
Tue Jul 11 20:07:33 GMT 2017
```

```
cat: /usr/share/options/nbtscan: No such file or directory
Tue Jul 11 20:10:39 GMT 2017
Doing NBT name scan for addresses from <EC4 - FW>.75

IP address       NetBIOS Name     Server    User             MAC address
-------------------------------------------------------------------------------
<EC4 - FW>.75    JENNAMARBLES     <server>  <unknown>        00-0c-29-b2-82-b8
Tue Jul 11 20:10:40 GMT 2017
Press Ctrl-C to close this window.
```

It does let you know the Netbios name of the machine.

iPerf-server

This is one way to start iPerf in server mode. There are others which will be described later on in this document, but this is the only way you can set options for iPerf-server. However in this version of iPerf there are few meaningful server options except for “-l” which tells iPerf to handle one connection, then quit.

But if you run iPerf from the Net-tools2 menu, you see the iPerf output with a very long delay (minutes), so it's recommended to use “Start Services” or “Startup-opts”, both of which are explained in a later section, to run iPerf server. “-i” might be a useful option for people who want to monitor and log each time iPerf is run as a server.

```
  -p, --port      #         server port to listen on/connect to
  -f, --format    [kmgKMG]  format to report: Kbits, Mbits, KBytes, MBytes
  -i, --interval  #         seconds between periodic bandwidth reports
  -F, --file name           xmit/recv the specified file
  -A, --affinity n/n,m      set CPU affinity
  -B, --bind      <host>    bind to a specific interface
  -V, --verbose             more detailed output
  -J, --json                output in JSON format
  -d, --debug               emit debugging output
  -v, --version             show version information and quit
  -h, --help                show this message and quit
  -D, --daemon              run the server as a daemon
  -1, --one-off             handle one client connection then exit
```

Here's what you'll get if you start iPerf using Start-Services or it starts when the NUK starts because you have it set on Startup-Opts (both discussed later):

```
------------------------------------------------------------
Server listening on 5201
------------------------------------------------------------
```

iPerf-client

In contrast to iPerf server the only way to run iPerf client is through this menu. You'll be asked for options. Here's a list of all of the client options.

```
  -p, --port      #         server port to listen on/connect to
  -f, --format    [kmgKMG]  format to report: Kbits, Mbits, KBytes, MBytes
  -i, --interval  #         seconds between periodic bandwidth reports
  -F, --file name           xmit/recv the specified file
  -A, --affinity n/n,m      set CPU affinity
  -B, --bind      <host>    bind to a specific interface
  -V, --verbose             more detailed output
  -J, --json                output in JSON format
  -d, --debug               emit debugging output
  -v, --version             show version information and quit
  -h, --help                show this message and quit
  -u, --udp                 use UDP rather than TCP
```

```
  -b, --bandwidth #[KMG][/#] target bandwidth in bits/sec (0 for unlimited)
                             (default 1 Mbit/sec for UDP, unlimited for TCP)
                             (optional slash and packet count for burst mode)
  -t, --time      #          time in seconds to transmit for (default 10 secs)
  -n, --bytes     #[KMG]     number of bytes to transmit (instead of -t)
  -k, --blockcount #[KMG]    number of blocks (packets) to transmit (instead of -t
or -n)
  -l, --len       #[KMG]     length of buffer to read or write
                             (default 128 KB for TCP, 8 KB for UDP)
  -P, --parallel  #          number of parallel client streams to run
  -R, --reverse              run in reverse mode (server sends, client receives)
  -w, --window    #[KMG]     set window size / socket buffer size
  -M, --set-mss   #          set TCP maximum segment size (MTU - 40 bytes)
  -N, --nodelay              set TCP no delay, disabling Nagle's Algorithm
  -4, --version4             only use IPv4
  -6, --version6             only use IPv6
  -S, --tos N                set the IP 'type of service'
  -Z, --zerocopy             use a 'zero copy' method of sending data
  -O, --omit N               omit the first n seconds
  -T, --title str            prefix every output line with this string
  --get-server-output        get results from server
```

The most common options are “-t n” which tells iPerf to run for n seconds or “-i n” which tells iPerf to report every n seconds. Also there's “-u “ to run in UDP mode which requires “-b n{KMG}” to be placed after the IP address. This is the bandwidth to send data at. “75M = 75Mbits/sec”, “75k = 75kbits/sec”, & “1G = 1Gbit/sec”

This is covered in more detail in the “Bandwidth Testing” Document, but we'll include a sample here. This is a connection from <East Coast site #4> which is inside a FW to the <East Coast> in TCP mode. This connection is crossing carriers, so there's a reasonable amount of latency, so as you can see the bandwidth varies from interval to interval (an interval is one line and by default is one second):

```
Wed Jun 14 08:21:43 GMT 2017
Connecting to host <East Coast>, port 5201
[  4] local <EC4 - FW>.202 port 65534 connected to <East Coast> port 5201
[ ID] Interval           Transfer     Bandwidth
[  4]   0.00-1.00   sec  1.96 MBytes  16.4 Mbits/sec
[  4]   1.00-2.00   sec  6.11 MBytes  51.2 Mbits/sec
[  4]   2.00-3.00   sec  9.66 MBytes  80.9 Mbits/sec
[  4]   3.00-4.00   sec  5.40 MBytes  45.4 Mbits/sec
[  4]   4.00-5.01   sec  8.81 MBytes  73.4 Mbits/sec
[  4]   5.01-6.00   sec  8.06 MBytes  67.9 Mbits/sec
[  4]   6.00-7.01   sec  8.63 MBytes  72.0 Mbits/sec
[  4]   7.01-8.00   sec  5.98 MBytes  50.6 Mbits/sec
[  4]   8.00-9.00   sec  8.19 MBytes  68.7 Mbits/sec
[  4]   9.00-10.00  sec  8.81 MBytes  73.9 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth
[  4]   0.00-10.00  sec  71.6 MBytes  60.1 Mbits/sec                  sender
[  4]   0.00-10.00  sec  71.5 MBytes  60.0 Mbits/sec                  receiver

iperf Done.
```

```
Wed Jun 14 08:21:53 GMT 2017
```

Nmap

Nmap is a port scanning, ping sweeping application. It can tell you what ports are open on a host or multiple hosts, and it can tell you which machines respond to a ping request. The most common options are: “-n “ – don't do DNS lookups on the IP addresses scanned, “-sP” – do a ping sweep, don't scan any TCP ports”, “-p “, pick which ports to scan, i.e. 21 or 1-65535 (to scan all port numbers), “-P0” or “Pn” scan a machine that is blocking ping probes.

By default Nmap scans it's preset list of “interesting” ports. Here's a list of the help output from NMAp

```
Nmap 6.47 ( http://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -iL <inputfilename>: Input from list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile <exclude_file>: Exclude list from file
HOST DISCOVERY:
  -sL: List Scan - simply list targets to scan
  -sn: Ping Scan - disable port scan
  -Pn: Treat all hosts as online -- skip host discovery
  -PS/PA/PU/PY[portlist]: TCP SYN/ACK, UDP or SCTP discovery to given ports
  -PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
  -PO[protocol list]: IP Protocol Ping
  -n/-R: Never do DNS resolution/Always resolve [default: sometimes]
  --dns-servers <serv1[,serv2],...>: Specify custom DNS servers
  --system-dns: Use OS's DNS resolver
  --traceroute: Trace hop path to each host
SCAN TECHNIQUES:
  -sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans
  -sU: UDP Scan
  -sN/sF/sX: TCP Null, FIN, and Xmas scans
  --scanflags <flags>: Customize TCP scan flags
  -sI <zombie host[:probeport]>: Idle scan
  -sY/sZ: SCTP INIT/COOKIE-ECHO scans
  -sO: IP protocol scan
  -b <FTP relay host>: FTP bounce scan
PORT SPECIFICATION AND SCAN ORDER:
  -p <port ranges>: Only scan specified ports
    Ex: -p22; -p1-65535; -p U:53,111,137,T:21-25,80,139,8080,S:9
  -F: Fast mode - Scan fewer ports than the default scan
  -r: Scan ports consecutively - don't randomize
  --top-ports <number>: Scan <number> most common ports
  --port-ratio <ratio>: Scan ports more common than <ratio>
SERVICE/VERSION DETECTION:
  -sV: Probe open ports to determine service/version info
  --version-intensity <level>: Set from 0 (light) to 9 (try all probes)
  --version-light: Limit to most likely probes (intensity 2)
  --version-all: Try every single probe (intensity 9)
  --version-trace: Show detailed version scan activity (for debugging)
OS DETECTION:
```

```
  -O: Enable OS detection
  --osscan-limit: Limit OS detection to promising targets
  --osscan-guess: Guess OS more aggressively
TIMING AND PERFORMANCE:
  Options which take <time> are in seconds, or append 'ms' (milliseconds),
  's' (seconds), 'm' (minutes), or 'h' (hours) to the value (e.g. 30m).
  -T<0-5>: Set timing template (higher is faster)
  --min-hostgroup/max-hostgroup <size>: Parallel host scan group sizes
  --min-parallelism/max-parallelism <numprobes>: Probe parallelization
  --min-rtt-timeout/max-rtt-timeout/initial-rtt-timeout <time>: Specifies
      probe round trip time.
  --max-retries <tries>: Caps number of port scan probe retransmissions.
  --host-timeout <time>: Give up on target after this long
  --scan-delay/--max-scan-delay <time>: Adjust delay between probes
  --min-rate <number>: Send packets no slower than <number> per second
  --max-rate <number>: Send packets no faster than <number> per second
FIREWALL/IDS EVASION AND SPOOFING:
  -f; --mtu <val>: fragment packets (optionally w/given MTU)
  -D <decoy1,decoy2[,ME],...>: Cloak a scan with decoys
  -S <IP_Address>: Spoof source address
  -e <iface>: Use specified interface
  -g/--source-port <portnum>: Use given port number
  --proxies <url1,[url2],...>: Relay connections through HTTP/SOCKS4 proxies
  --data-length <num>: Append random data to sent packets
  --ip-options <options>: Send packets with specified ip options
  --ttl <val>: Set IP time-to-live field
  --spoof-mac <mac address/prefix/vendor name>: Spoof your MAC address
  --badsum: Send packets with a bogus TCP/UDP/SCTP checksum
OUTPUT:
  -oN/-oX/-oS/-oG <file>: Output scan in normal, XML, s|<rIpt kIddi3,
     and Grepable format, respectively, to the given filename.
  -oA <basename>: Output in the three major formats at once
  -v: Increase verbosity level (use -vv or more for greater effect)
  -d: Increase debugging level (use -dd or more for greater effect)
  --reason: Display the reason a port is in a particular state
  --open: Only show open (or possibly open) ports
  --packet-trace: Show all packets sent and received
  --iflist: Print host interfaces and routes (for debugging)
  --log-errors: Log errors/warnings to the normal-format output file
  --append-output: Append to rather than clobber specified output files
  --resume <filename>: Resume an aborted scan
  --stylesheet <path/URL>: XSL stylesheet to transform XML output to HTML
  --webxml: Reference stylesheet from Nmap.Org for more portable XML
  --no-stylesheet: Prevent associating of XSL stylesheet w/XML output
MISC:
  -6: Enable IPv6 scanning
  -A: Enable OS detection, version detection, script scanning, and traceroute
  --datadir <dirname>: Specify custom Nmap data file location
  --send-eth/--send-ip: Send using raw ethernet frames or IP packets
  --privileged: Assume that the user is fully privileged
  --unprivileged: Assume the user lacks raw socket privileges
  -V: Print version number
  -h: Print this help summary page.
EXAMPLES:
  nmap -v -A scanme.nmap.org
  nmap -v -sn 192.168.0.0/16 10.0.0.0/8
  nmap -v -iR 10000 -Pn -p 80
SEE THE MAN PAGE (http://nmap.org/book/man.html) FOR MORE OPTIONS AND EXAMPLES
```

Here are several examples of nmap output:

First a ping sweep:

Here are the options we send to nmap:

```
     +--------------------------- nmap options ---------------------------+
     |             What options do you want to pass to nmap                |
     | +------------------------------------------------------------------+ |
     +-|-n -sn                                                            |-+
     | +---------------------<  OK  >------<Cancel>---------------------+ |
     +-------------------------------------------------------------------+
```

And here is the address range to sweep:

```
     +---------------------- IP Address for nmap ----------------------+
     |           What IP address do you want to pass to nmap            |
     | +--------------------------------------------------------------+ |
     +-|N.O.P.0/24                                                    |-+
     | +---------------------<  OK  >------<Cancel>---------------------+ |
     +-------------------------------------------------------------------+
```

And here is the result (a truncated version to spare space)

```
cat: /usr/share/options/nmap: No such file or directory
Wed Jul 12 19:28:35 GMT 2017

Starting Nmap 6.47 ( http://nmap.org ) at 2017-07-12 19:28 GMT
Nmap scan report for N.O.P.1
Host is up (0.00053s latency).
MAC Address: 00:03:2D:2C:4A:6A (Ibase Technology)
Nmap scan report for N.O.P.10
Host is up (-0.18s latency).
MAC Address: 88:DC:96:3B:14:74 (Senao Networks)
Nmap scan report for N.O.P.100
Host is up (0.00043s latency).
MAC Address: 88:DC:96:36:C8:3D (Senao Networks)
Nmap scan report for N.O.P.110
Host is up (-0.12s latency).
MAC Address: 88:DC:96:36:C8:37 (Senao Networks)
Nmap scan report for N.O.P.111
Host is up (0.00055s latency).
MAC Address: 88:DC:96:36:C8:61 (Senao Networks)
Nmap scan report for N.O.P.112
Host is up (0.00065s latency).
MAC Address: 88:DC:96:36:75:D2 (Senao Networks)
Nmap scan report for N.O.P.120
Host is up (0.00054s latency).
MAC Address: 88:DC:96:36:C8:34 (Senao Networks)
Nmap scan report for N.O.P.122
Host is up (-0.11s latency).
MAC Address: 88:DC:96:36:C7:17 (Senao Networks)
Nmap scan report for N.O.P.125
Host is up (0.00017s latency).
MAC Address: 88:DC:96:36:C7:0E (Senao Networks)
Nmap scan report for N.O.P.126
Host is up (0.00060s latency).
```

```
MAC Address: 88:DC:96:36:C8:31 (Senao Networks)
Nmap scan report for N.O.P.127
Host is up (0.00060s latency).
MAC Address: 88:DC:96:36:C8:2E (Senao Networks)
Nmap scan report for N.O.P.130
Host is up (0.00048s latency).
…
Nmap done: 256 IP addresses (29 hosts up) scanned in 5.55 seconds
Wed Jul 12 19:28:41 GMT 2017
```

Now this is a port scan of one specific host.

Here are the options set:

```
    +--------------------------- nmap options --------------------------+
    |             What options do you want to pass to nmap               |
    | +-----------------------------------------------------------------+ |
    +-|-n                                                              |-+
    | +---------------------<  OK  >------<Cancel>---------------------+ |
    +-------------------------------------------------------------------+
```

And here is the host we want scanned:

```
    +----------------------- IP Address for nmap -----------------------+
    |           What IP address do you want to pass to nmap              |
    | +-----------------------------------------------------------------+ |
    +-|N.O.P.100                                                      |-+
    | +---------------------<  OK  >------<Cancel>---------------------+ |
    +-------------------------------------------------------------------+
```

And here's the result:

```
cat: /usr/share/options/nmap: No such file or directory
Wed Jul 12 19:05:00 GMT 2017

Starting Nmap 6.47 ( http://nmap.org ) at 2017-07-12 19:05 GMT
Nmap scan report for N.O.P.100
Host is up (0.00046s latency).
Not shown: 994 closed ports
PORT     STATE SERVICE
23/tcp   open  telnet
53/tcp   open  domain
80/tcp   open  http
443/tcp  open  https
5001/tcp open  commplex-link
5002/tcp open  rfe
MAC Address: 88:DC:96:36:C8:3D (Senao Networks)

Nmap done: 1 IP address (1 host up) scanned in 160.49 seconds
Wed Jul 12 19:07:40 GMT 2017
```

Rdesktop

This is the same Rdesktop as referenced earlier in this document. See – VNC Menu – Rdesktop (Windows Remote Desktop), for all the details of launching and running rdesktop.

Rinetd

Rinetd is a program that allows you to open a port on the management interface and connect it to an IP and port on another IP address (on any interface). We discussed this under Rdesktop (above), but here's a copy:

If you have a machine running a newer version of the RDP protocol then our Rdesktop client supports, create a file named rinetd.conf as follows:

0.0.0.0 3389 <Remote Desktop Device IP> 3389

This says listen on port 3389, then when a connection is made, connect to port 3389 on the Remote Desktop Device IP and pass the data between the two connections. It's as if the outside user is connecting directly to the Remote Desktop Device.

Upload this file it to /config if using SFTP (/ (root file system) if using FTP. Start rinetd using Start Services, then Remote Desktop to the NUK IP address.

For VNC, create rinetd.conf as follows:

0.0.0.0 5900 <VNC IP> 5901

(Port 5900 is in use on the NUK)

Upload this file it to /config if using SFTP (/ (root file system) if using FTP. Start rinetd using Start Services, then VNC to the NUK IP address:1 (Desktop 1). The NUK display runs on Desktop 0

For Citrix:

0.0.0.0 1494 <Citrix IP> 1494

And connect with your Citrix client to the NUK IP address.

After creating and uploading this file, launch rinetd from Start Services or from this menu. If you kill the window rinetd is running in, then the app will shut down.

Tinyproxy

Tinyproxy is a lightweight proxy server that by default runs on port 8888. It's also available via Stunnel on port 48228. Tinyproxy is configured via tinyproxy.conf. The most important option is the listen command. The line needs to read:

Listen 0.0.0.0

Otherwise, Tinyproxy will only listen on ipv6 addresses and since ipv6 isn't configured, tinyproxy will do nothing.

In 8 years of experience, tinyproxy has just worked. Most of the options are for things like upstream proxies and options for higher load situations. The only problems we've ever experienced involved the Listen line. Logging is also turned off, but that can be turned on with the Syslog option – add “Syslog On” to the tinyproxy.conf in /config (via SFTP) or / (via FTP). Or to log tinyproxy messages to a separate log file, choose “Logging /config/storage/log/<filename>” to log to a specific file. You can place the file wherever you want, but /config/storage/log is where all the other log files are. Wherever you put it, use /config/storage as that is the biggest partition by far – you won't run out of space there. Somewhere in /config you risk running out of space, and anywhere not in /config is on the embedded ramdisk which gets refreshed everytime you reboot (meaning if you put the logfile in the ramdisk it gets erased every reboot).

Note: If you use stunnel, then logging is pointless as all the connections will appear to come from a local address.

Floodping

In windows you can only send a ping once a second. Mac and Unix machines have the option to do something called a flood ping. On a unix machine this sends packets at a minimum of 100 packets/second and send 500 packets a time. You don't set any options, you just enter an IP address. For example:

```
Fri Jul 21 17:10:00 GMT 2017
PING <East Coast - GW>.IPYX-118413-ZYO.zip.zayo.com (<East Coast - GW>): 56 data
bytes
.

----<East Coast - GW>.IPYX-118413-ZYO.zip.zayo.com PING Statistics----
500 packets transmitted, 500 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.257/0.493/2.213/0.139 ms
  2023.5 packets/sec sent,  2027.2 packets/sec received
Fri Jul 21 17:10:01 GMT 2017
Press Ctrl-C to close this window.
```

You can see that the <East Coast – GW> responded to all 500 packets and that it sent them at over 2000 packets a second. The NUK received them at a slightly higher rate, which is a rounding error. But this tells you that the device is working fine and that there is nothing wrong in the network infrastructure between the NUK at the <East Coast – GW>.

Telnet

Telnet is the remote screen application. It's included for connecting to network devices (routers and switches). The NUK doesn't support any form of authentication except plain text.

Here's the list of options:

```
telnet: unknown option -- h
usage: telnet [-4] [-6] [-8] [-E] [-K] [-L] [-N] [-S tos] [-X atype] [-a] [-c]
        [-d] [-e char] [-k realm] [-l user] [-f/-F] [-n tracefile] [-r] [-x]
        [-P policy] [host-name [port]]
```

If you want to look them up, they're all listed in the man page in the Appendix. Unfortunately, there are many, many telnet options, so the man page will be quite long. Here's an example telnet session:

```
cat: /usr/share/options/telnet: No such file or directory
Fri Jul 21 17:20:43 GMT 2017
Trying <RFC 1918 IP>...
Connected to <RFC 1918 IP>.
Escape character is '^]'.


User Access Verification

Password:
Router>en
Password:
Router#show config
Using 2370 out of 524288 bytes
```

```
!
version 12.2
no service pad
service timestamps debug datetime msec localtime
service timestamps log datetime msec localtime
service password-encryption
!
hostname Router
!
enable secret 5 $1$WJ7G$XMjPO/GBGwRuZP426Dyl.0
!
no aaa new-model
switch 1 provision ws-c3750g-24t
switch 2 provision ws-c3750g-24t
ip subnet-zero
!
!
!
!
no file verify auto
spanning-tree mode pvst
spanning-tree extend system-id
!
vlan internal allocation policy ascending
!
interface GigabitEthernet1/0/1
!
interface GigabitEthernet1/0/2
!
interface GigabitEthernet1/0/3
!
interface GigabitEthernet1/0/4
!
interface GigabitEthernet1/0/5
!
interface GigabitEthernet1/0/6
!
interface GigabitEthernet1/0/7
```

As you can see, the NUK connected to a Cisco router and viewed it's config. The text sent by the NUK is in italics, for reference purposes.

nslookup

Nslookup is an interactive way to query DNS servers. It can be used to do a single lookup or it can be used interactively. To do a single lookup, just enter a name in the “What IP address do you want to send to nslookup?” prompt. Here's an example:

```
    +--------------------- IP Address for nslookup ----------------------+
    |         What IP address do you want to pass to nslookup            |
    | +-------------------------------------------------------------+ |
```

```
    +-|www.bigjar.com                                                    |-+
    | +--------------------<  OK  >------<Cancel>---------------------+ |
    +-------------------------------------------------------------------+
```

And here's the result:

```
cat: /usr/share/options/nslookup: No such file or directory
Fri Jul 21 18:47:43 GMT 2017
Server:  google-public-dns-a.google.com
Address:  8.8.8.8

Non-authoritative answer:
Name:    bigjar.com
Address:  37.60.234.247
Aliases:  www.bigjar.com

Fri Jul 21 18:47:43 GMT 2017
```

You can see that this NUK is using Google's name server (8.8.8.8) and that www.bigjar.com is an alias for bigjar.com and the IP address of this host is 37.60.234.247.

If you don't enter a DNS name to look up, then you can go into interactive mode:

```
cat: /usr/share/options/nslookup: No such file or directory
Fri Jul 21 18:53:11 GMT 2017
Default Server:  google-public-dns-a.google.com
Address:  8.8.8.8

>
```

From here I enter a series of commands. Here's what I do:

Ask the server to query for mx (Mail eXchanger) records (#1) – these servers determine what servers receive mail for a domain. Query bigjar.com revealing the name servers (#2). Switch back to A (Address as in IP address) records (the default record type to search) (#3). Looked up the IP address of mail.bigjar.com (#4). Switched again to querying NS (Name Server) records (#5). Looked up bigjar.com again (#6). Switched back to A records (#7). Switched to the first name server listed (#8). Queried bigjar.com again (#9). Then I exited nslookup (#10). You'll notice that when I switched to the NS for bigjar.com the “Non-authoritative answer:” went away. That's because I was talking to one of the four authoritative servers. The google server (8.8.8.8) was giving cached responses

```
> set qtype=mx                      #1
> bigjar.com                        #2
Server:  google-public-dns-a.google.com
Address:  8.8.8.8
```

```
Non-authoritative answer:
bigjar.com      preference = 20, mail exchanger = spamtitan.bigjar.com
bigjar.com      preference = 40, mail exchanger = mail.bigjar.com
> set qtype=a                       #3
> mail.bigjar.com                   #4
Server:  google-public-dns-a.google.com
Address:  8.8.8.8

Non-authoritative answer:
Name:    mail.bigjar.com
Address:  96.95.20.252

> set qtype=ns                      #5
> bigjar.com                        #6
Server:  google-public-dns-a.google.com
Address:  8.8.8.8

Non-authoritative answer:
bigjar.com      nameserver = ns3176.dns.dyn.com
bigjar.com      nameserver = ns1167.dns.dyn.com
bigjar.com      nameserver = ns4151.dns.dyn.com
bigjar.com      nameserver = ns2141.dns.dyn.com
> set qtype=a                       #7

> server ns3176.dns.dyn.com         #8
Default Server:  ns3176.dns.dyn.com
Address:  208.76.60.176

> bigjar.com                        #9
Server:  ns3176.dns.dyn.com
Address:  208.76.60.176

Name:    bigjar.com
Address:  37.60.234.247

> exit                              #10
Fri Jul 21 19:22:09 GMT 2017
Press Ctrl-C to close this window.
```

One last feature that's helpful is turning on debug:

```
cat: /usr/share/options/nslookup: No such file or directory
Fri Jul 21 19:33:51 GMT 2017
Default Server:  google-public-dns-a.google.com
Address:  8.8.8.8

> bigjar.com
Server:  google-public-dns-a.google.com
Address:  8.8.8.8

Non-authoritative answer:
Name:    bigjar.com
Address:  37.60.234.247

> set debug
> bigjar.com
```

```
Server:  google-public-dns-a.google.com
Address:  8.8.8.8

;; res_nmkquery(QUERY, bigjar.com, IN, A)
------------
Got answer:
    HEADER:
        opcode = QUERY, id = 40803, rcode = NOERROR
        header flags:  response, want recursion, recursion avail.
        questions = 1,  answers = 1,  authority records = 0,  additional = 0

    QUESTIONS:
        bigjar.com, type = A, class = IN
    ANSWERS:
    ->  bigjar.com
        internet address = 37.60.234.247
        ttl = 415 (6m55s)

------------
Non-authoritative answer:
Name:    bigjar.com
Address:  37.60.234.247

> bigjar.com
Server:  google-public-dns-a.google.com
Address:  8.8.8.8

;; res_nmkquery(QUERY, bigjar.com, IN, A)
------------
Got answer:
    HEADER:
        opcode = QUERY, id = 40804, rcode = NOERROR
        header flags:  response, want recursion, recursion avail.
        questions = 1,  answers = 1,  authority records = 0,  additional = 0

    QUESTIONS:
        bigjar.com, type = A, class = IN
    ANSWERS:
    ->  bigjar.com
        internet address = 37.60.234.247
        ttl = 405 (6m45s)

------------
Non-authoritative answer:
Name:    bigjar.com
Address:  37.60.234.247
```

The most useful part of debug is the fact that it shows you the ttl (or time to live) for the DNS record. If you're waiting for the Internet to catch up to a recent DNS change, checking the TTL on public name servers (like Google's) is a good way to do it. You can see that the ttl dropped 10 seconds between queries.

dbclient (SSH Client)

The Dropbear client is an SSH v2 only client. Like telnet, it's included for connecting to routers and switches. Here are the options:

```
cat: /usr/share/options/dbclient: No such file or directory
Fri Jul 21 17:37:28 GMT 2017
Dropbear SSH client v2013.62 https://matt.ucc.asn.au/dropbear/dropbear.html
Usage: /usr/pkg/bin/dbclient [options] [user@]host[/port][,[user@]host/port],...]
[command]
-p <remoteport>
-l <username>
-t    Allocate a pty
-T    Don't allocate a pty
-N    Don't run a remote command
-f    Run in background after auth
-y    Always accept remote host key if unknown
-y -y Don't perform any remote host key checking (caution)
-s    Request a subsystem (use by external sftp)
-i <identityfile>   (multiple allowed)
-A    Enable agent auth forwarding
-L <[listenaddress:]listenport:remotehost:remoteport> Local port forwarding
-g    Allow remote hosts to connect to forwarded ports
-R <[listenaddress:]listenport:remotehost:remoteport> Remote port forwarding
-W <receive_window_buffer> (default 24576, larger may be faster, max 1MB)
-K <keepalive>  (0 is never, default 0)
-I <idle_timeout>  (0 is never, default 0)
-B <endhost:endport> Netcat-alike forwarding
-J <proxy_program> Use program pipe rather than TCP connection
-c <cipher list> Specify preferred ciphers ('-c help' to list options)
-m <MAC list> Specify preferred MACs for packet verification (or '-m help')
```

Here the NUK connects to another NUK. First the options:

```
    +------------------------- dbclient options -------------------------+
    |           What options do you want to pass to dbclient             |
    | +----------------------------------------------------------------+ |
    +-|-p 48226                                                         |-+
    | +---------------------<  OK  >------<Cancel>---------------------+ |
    +-----------------------------------------------------------------+
```

Then the IP address:

```
    +--------------------- IP Address for dbclient ----------------------+
    |         What IP address do you want to pass to dbclient            |
    | +----------------------------------------------------------------+ |
    +-+root@WestCoast2                                                 |-+
    | +---------------------<  OK  >------<Cancel>---------------------+ |
    +-----------------------------------------------------------------+
```

```
cat: /usr/share/options/dbclient: No such file or directory
Fri Jul 21 18:20:15 GMT 2017
root@40.135.239.67's password:
```

And then we get the Remote NUK's main menu:

```
    +---------- Network Utility Knife R1.2k43 Hostname: West Coast2 -----+
    |            Please choose from the following options:               |
    | +-----------------------------------------------------------------+ |
    | |              a)  Configure IP Address Information               | |
    | |              b)  Change Shell Password                          | |
    | |              c)  Change Hostname                                | |
    | |              d)  Advanced IP configuration                      | |
    | |              e)  Restart Network Utility Knife                  | |
    | |              f)  Load new boot image                            | |
    | |              g)  Set Date and Time                              | |
    | |              h)  Advanced Functions                             | |
    | |              i)  Exit                                           | |
    | +-----------------------------------------------------------------+ |
    +---------------------------------------------------------------------+
    |                       <  OK  >      <Cancel>                        |
    +---------------------------------------------------------------------+
```

Start Services

This menu choice starts services manually. It has the same menu (except for the title) as Startup-opts, which starts services on bootup. This is what you get when you select Start Services:

```
    +------------------------ Startup Options --------------------------+
    |        Please select which programs you want to run now:           |
    | +-----------------------------------------------------------------+ |
    | |  [ ] ntop1       Text based network monitor on Mon1 interface  | |
    | |  [ ] ntop2       Text based network monitor on Mon2 interface  | |
    | |  [ ] nprobe1     Netflow export of flows on Mon1    interface  | |
    | |  [ ] nprobe2     Netflow export of flows on Mon2    interface  | |
    | |  [ ] iperfs      iperf (throughput testing) in server mode     | |
    | |  [ ] rinetds     Tcp port redirector - like plug-gw            | |
    | |  [ ] netmonitor  Netmonitor - pings hosts - emails if down     | |
    | |  [ ] tinyproxys  Tinyproxy - proxy server w/a small footprint  | |
    | |                                                                | |
    | |                                                                | |
    | |                                                                | |
    | |                                                                | |
    | +-----------------------------------------------------------------+ |
    |                                                                     |
    |                                                                     |
    |                                                                     |
    +---------------------------------------------------------------------+
    |                       <  OK  >      <Cancel>                        |
    +---------------------------------------------------------------------+
```

To start a service or services, move the cursor to the line you want and press the spacebar. You can select multiple services. When you are done, press enter and the services will start in separate windows. Here's a brief summary of the services listed:

ntop1

This is the early text based version of ntop running on the Text based network monitor on the Monitor1 interface.

Ntop2

Same as above, but running on the Monitor2 interface.

Nprobe1

Would run nprobe (Netflow collector) on Monitor 1 interface. **Currently Disabled**

nprobe2

Same as above but on Monitor 2 interface. **Currently Disabled**

iperfs

Iperf version 3 running in server mode. Iperf 3 listens in UDP mode and TCP mode in one instance. Discussed above in Net-Tools2 section.

rinetds

rinetd redirects connections from a port on the NUK to another port on another IP address. Discussed above in Net-Tools2 section.

netmonitor

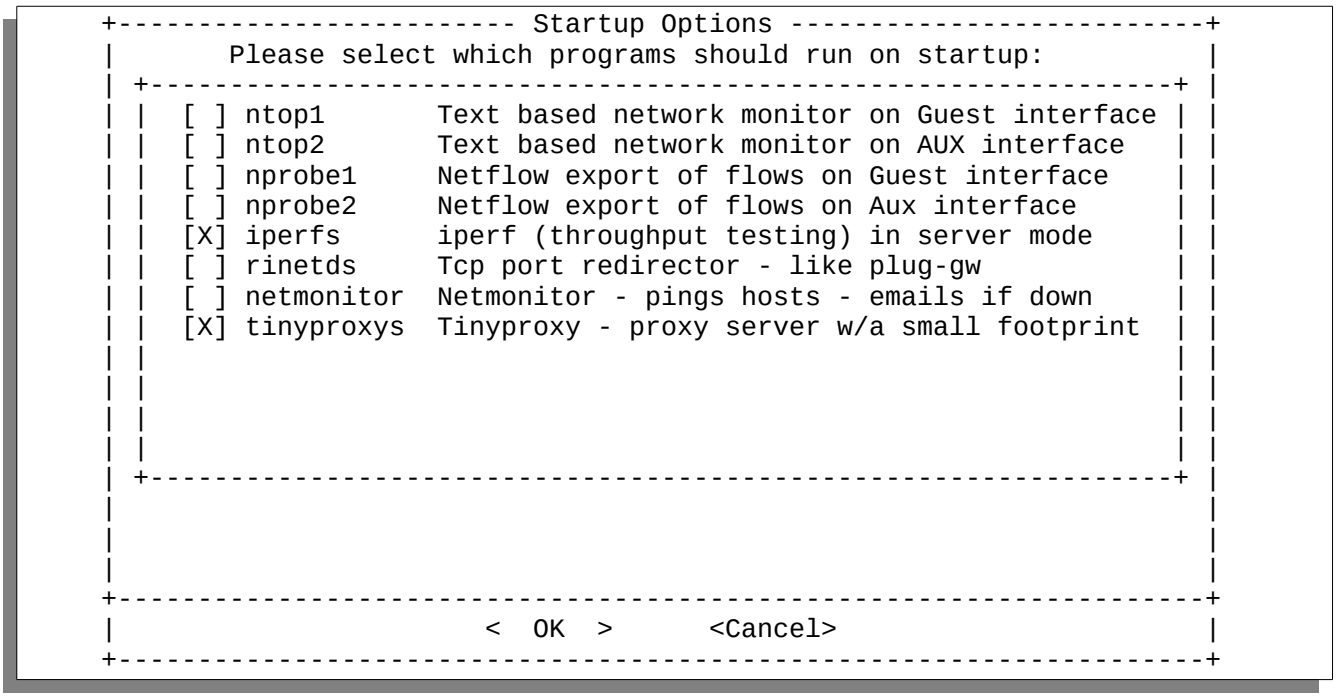
Discussed in a separate document.

tinyproxys

Tinyproxy is a small footprint proxy server. Covered above in the Net-Tools2 section

Startup-Opts

Startup-Opts allows you to select services to run every time the NUK is started. Here is what the menu looks like after you select Startup-Opts from the VNC menu:



It's exactly identical to Start Services, except for one very important difference. If you have services selected and you hit <Esc> or <Ctrl-C>, then those services will be unchecked and won't start when the NUK is rebooted. Always, always, exit this menu by hitting <Enter>. That way everything that was checked remains checked.

The services listed are the same as those mentioned under Start Services.

Appendix A

This appendix contains man (manual) pages for the utilities that are part of the NUK

Rdesktop

```
rdesktop(1)                 General Commands Manual                rdesktop(1)

NAME
       rdesktop - Remote Desktop Protocol client

SYNOPSIS
       rdesktop [options] server[:port]

DESCRIPTION
       rdesktop is a client for Remote Desktop Protocol (RDP), used in a
       number of Microsoft products including Windows NT Terminal Server,
       Windows 2000 Server, Windows XP and Windows 2003 Server.

OPTIONS
       -u <username>
              Username for authentication on the server.

       -d <domain>
              Domain for authentication.

       -s <shell>
              Startup shell for the user - starts a specific application
              instead of Explore.  If SeamlessRDP is enabled this is the
              application which i started in seamless mode.

       -c <directory>
              The initial working directory for the user.  Often used in
              combination with -s to set up a fixed login environment.

       -p <password>
              The password to authenticate with.  Note that this may have no
              effect if "Always prompt for password" is enabled on the server.
              WARNING: if you specify a password on the command line it may be
              visible to other users when they use tools like ps.  Use -p - to
              make rdesktop request a password at startup (from standard
              input).

       -n <hostname>
              Client hostname.  Normally rdesktop automatically obtains the
              hostname of the client.

       -k <keyboard-map>
              Keyboard layout to emulate.  This requires a corresponding
              keymap file to be installed.  The standard keymaps provided with
              rdesktop follow the RFC1766 naming scheme: a language code
              followed by a country code if necessary - e.g.  en-us, en-gb,
              de, fr, sv, etc.
```

The default keyboard map depends on the current locale (LC_* and LANG environment variables). If the current locale is unknown, the default keyboard map is en-us (a US English keyboard).

The keyboard maps are file names, which means that they are case sensitive. The standard keymaps are all in lowercase.

The keyboard maps are searched relative to the directories $HOME/.rdesktop/keymaps, KEYMAP_PATH (specified at build time), and $CWD/keymaps, in this order. The keyboard-map argument can also be an absolute filename.

The special value `none' can be used instead of a keyboard map. In this case, rdesktop will guess the scancodes from the X11 event key codes using an internal mapping method. This method only supports the basic alphanumeric keys and may not work properly on all platforms so its use is discouraged.

-g <geometry>
Desktop geometry (WxH). If geometry is the special word "workarea", the geometry will be fetched from the extended window manager hints property _NET_WORKAREA, from the root window. The geometry can also be specified as a percentage of the whole screen, e.g. "-g 80%".

If the specified geometry depends on the screen size, and the screen size is changed, rdesktop will automatically reconnect using the new screen size. This requires that rdesktop has been compiled with RandR support.

-i Use password as smartcard pin. If a valid user certificate is matched in smart card reader the password passed with p argument is used as pin for the smart card. This feature also requires that smart card redirection is used using r scard argument.

-f Enable fullscreen mode. This overrides the window manager and causes the rdesktop window to fully cover the current screen. Fullscreen mode can be toggled at any time using Ctrl-Alt-Enter.

-b Force the server to send screen updates as bitmaps rather than using higher-level drawing operations.

-t Disable use of remote control. This will disable features like seamless connection sharing.

-A <seamlessrdpshell>
Enable SeamlessRDP by specifying the path to seamless rdp shell. In this mode, rdesktop creates a X11 window for each window on the server side. This mode requires the SeamlessRDP server side component, which is available from http://www.cendio.com/seamlessrdp/.

When using this option, you should normally specify a startup shell which launches the desired application through SeamlessRDP.

Example: rdesktop -A 'c:\seamlessrdp\seamlessrdpshell.exe' -s 'notepad' mywts.domain.com

Any subsequential call to the above commandline example will make use of the seamless connection sharing feature which spawns another notepad in the current connection to the specified server and then exit.

-B Use the BackingStore of the Xserver instead of the integrated one in rdesktop.

-e Disable encryption. This option is only needed (and will only work) if you have a French version of NT TSE.

-E Disable encryption from client to server. This sends an encrypted login packet, but everything after this is unencrypted (including interactive logins).

-m Do not send mouse motion events. This saves bandwidth, although some Windows applications may rely on receiving mouse motion.

-C Use private colourmap. This will improve colour accuracy on an 8-bit display, but rdesktop will appear in false colour when not focused.

-D Hide window manager decorations, by using MWM hints.

-K Do not override window manager key bindings. By default rdesktop attempts to grab all keyboard input when it is in focus.

-S <button size>
Enable single application mode. This option can be used when running a single, maximized application (via -s). When the minimize button of the windows application is pressed, the rdesktop window is minimized instead of the remote application. The maximize/restore button is disabled. For this to work, you must specify the correct button size, in pixels. The special word "standard" means 18 pixels.

-T <title>
Sets the window title. The title must be specified using an UTF-8 string.

-N Enable numlock syncronization between the Xserver and the remote RDP session. This is useful with applications that looks at the numlock state, but might cause problems with some Xservers like Xvnc.

-X <windowid>
Embed rdesktop-window in another window. The windowid is expected to be decimal or hexadecimal (prefixed by 0x).

-a <bpp>
Sets the colour depth for the connection (8, 15, 16, 24 or 32). More than 8 bpp are only supported when connecting to Windows XP (up to 16 bpp) or newer. Note that the colour depth may also be limited by the server configuration. The default value is the depth of the root window.

-z Enable compression of the RDP datastream.

```
-x <experience>
       Changes default bandwidth performance behaviour for RDP5. By
       default only theming is enabled, and all other options are
       disabled (corresponding to modem (56 Kbps)). Setting experience
       to b[roadband] enables menu animations and full window dragging.
       Setting experience to l[an] will also enable the desktop
       wallpaper. Setting experience to m[odem] disables all (including
       themes). Experience can also be a hexidecimal number containing
       the flags.

-P     Enable caching of bitmaps to disk (persistent bitmap caching).
       This generally improves performance (especially on low bandwidth
       connections) and reduces network traffic at the cost of slightly
       longer startup and some disk space.  (10MB for 8-bit colour,
       20MB for 15/16-bit colour, 30MB for 24-bit colour and 40MB for
       32-bit colour sessions)

-r <device>
       Enable redirection of the specified device on the client, such
       that it appears on the server. Note that the allowed
       redirections may be restricted by the server configuration.

       Following devices are currently supported:

-r comport:<comport>=<device>,...
       Redirects serial devices on your client to the server. Note that
       if you need to change any settings on the serial device(s), do
       so with an appropriate tool before starting rdesktop. In most
       OSes you would use stty. Bidirectional/Read support requires
       Windows XP or newer.  In Windows 2000 it will create a port, but
       it's not seamless, most shell programs will not work with it.

-r disk:<sharename>=<path>,...
       Redirects a path to the share \\tsclient\<sharename> on the
       server (requires Windows XP or newer). The share name is limited
       to 8 characters.

-r lptport:<lptport>=<device>,...
       Redirects parallel devices on your client to the server.
       Bidirectional/Read support requires Windows XP or newer. In
       Windows 2000 it will create a port, but it's not seamless, most
       shell programs will not work with it.

-r printer:<printername>[=<driver>],...
       Redirects a printer queue on the client to the server. The
       <printername> is the name of the queue in your local system.
       <driver> defaults to a simple PS-driver unless you specify one.
       Keep in mind that you need a 100% match in the server
       environment, or the driver will fail. The first printer on the
       command line will be set as your default printer.

-r sound:[local|off|remote]
       Redirects sound generated on the server to the client. "remote"
       only has any effect when you connect to the console with the -0
       option. (Requires Windows XP or newer).

-r lspci
       Activates the lspci channel, which allows the server to
       enumerate the clients PCI devices. See the file lspci-
```

```
              channel.txt in the documentation for more information.

       -r scard[:<Scard Name>=<Alias Name>[;<Vendor Name>][,...]]
              Enables redirection of one or more smart-cards. You can provide
              static name binding between linux and windows. To do this you
              can use optional parameters as described: <Scard Name> - device
              name in Linux/Unix enviroment, <Alias Name> - device name shown
              in Windows enviroment <Vendor Name> - optional device vendor
              name. For list of examples run rdesktop without parameters.

       -0     Attach to the console of the server (requires Windows Server
              2003 or newer).

       -4     Use RDP version 4.

       -5     Use RDP version 5 (default).

CredSSP Smartcard options
       --sc-csp-name <name>
              Specify the CSP (Crypto Service Provider) to use on the windows
              side for the smartcard authentication. CSP is the driver for
              your smartcard and it seems like this is required to be
              specified for CredSSP authentication. For swedish NetID the
              following CSP name is used; "Net iD - CSP".

       --sc-container-name <name>
              Specify the container name, usally this is the username for
              default container and it seems like this is required to be
              specified for CredSSP authentication.

       --sc-reader-name <name>
              Specify the reader name to be used to prevent the pin code being
              sent to wrong card if there are several readers.

       --sc-card-name <name>
              Specify the card name for example; "Telia EID IP5a".

EXIT VALUES
       0      RDP session terminated normally

       1      Server initiated disconnect (also returned for logoff by XP
              joined to a domain)

       2      Server initiated logoff

       3      Server idle timeout reached

       4      Server logon timeout reached

       5      The session was replaced

       6      The server is out of memory

       7      The server denied the connection

       8      The server denied the connection for security reason

       9      The user cannot connect to the server due to insufficient access
              privileges
```

10 The server does not accept saved user credentials and requires that the user enter their credentials for each connection

11 Disconnect initiated by administration tool

12 Disconnect initiated by user

16 Internal licensing error

17 No license server available

18 No valid license available

19 Invalid licensing message

20 Hardware id doesn't match software license

21 Client license error

22 Network error during licensing protocol

23 Licensing protocol was not completed

24 Incorrect client license enryption

25 Can't upgrade license

26 The server is not licensed to accept remote connections

62 The local client window was closed

63 Some other, unknown error occured

64 Command line usage error

69 A service or resource (such as memory) is unavailable

70 An internal software error has been detected

71 Operating system error

76 Protocol error or unable to connect to remote host.

LINKS

Main website of rdesktop
http://www.rdesktop.org/

TCPDUMP

```
              TCPDUMP(1)                  General Commands Manual
TCPDUMP(1)

NAME
       tcpdump - dump traffic on a network

SYNOPSIS
       tcpdump [ -AbdDefhHIJKlLnNOpqRStuUvxX# ] [ -B buffer_size ]
       [ -c count ]
       [ -C file_size ] [ -G rotate_seconds ] [ -F file ]
       [ -i interface ] [ -j tstamp_type ] [ -m module ] [ -M secret ]
       [ --number ] [ -Q in|out|inout ] [ -r file ] [ -V file ] [ -s snaplen ]
       [ -T type ] [ -w file ]
       [ -W filecount ]
       [ -E spi@ipaddr algo:secret,... ]
       [ -y datalinktype ] [ -z postrotate-command ] [ -Z user ] [
       --time-stamp-precision=tstamp_precision ] [ --immediate-mode ] [
       --version ] [ expression ]

DESCRIPTION
       Tcpdump prints out a description of the contents of packets on a
       network interface that match the boolean expression; the description is
       preceded by a time stamp, printed, by default, as hours, minutes,
       seconds, and fractions of a second since midnight.  It can also be run
       with the -w flag, which causes it to save the packet data to a file for
       later analysis, and/or with the -r flag, which causes it to read from a
       saved packet file rather than to read packets from a network interface.
       It can also be run with the -V flag, which causes it to read a list of
       saved packet files. In all cases, only packets that match expression
       will be processed by tcpdump.

       Tcpdump will, if not run with the -c flag, continue capturing packets
       until it is interrupted by a SIGINT signal (generated, for example, by
       typing your interrupt character, typically control-C) or a SIGTERM
       signal (typically generated with the kill(1) command); if run with the
       -c flag, it will capture packets until it is interrupted by a SIGINT or
       SIGTERM signal or the specified number of packets have been processed.

       When tcpdump finishes capturing packets, it will report counts of:

              packets ``captured'' (this is the number of packets that tcpdump
              has received and processed);

              packets ``received by filter'' (the meaning of this depends on
              the OS on which you're running tcpdump, and possibly on the way
              the OS was configured - if a filter was specified on the command
              line, on some OSes it counts packets regardless of whether they
              were matched by the filter expression and, even if they were
              matched by the filter expression, regardless of whether tcpdump
              has read and processed them yet, on other OSes it counts only
              packets that were matched by the filter expression regardless of
              whether tcpdump has read and processed them yet, and on other
              OSes it counts only packets that were matched by the filter
```

expression and were processed by tcpdump);

packets ``dropped by kernel'' (this is the number of packets that were dropped, due to a lack of buffer space, by the packet capture mechanism in the OS on which tcpdump is running, if the OS reports that information to applications; if not, it will be reported as 0).

On platforms that support the SIGINFO signal, such as most BSDs (including Mac OS X) and Digital/Tru64 UNIX, it will report those counts when it receives a SIGINFO signal (generated, for example, by typing your ``status'' character, typically control-T, although on some platforms, such as Mac OS X, the ``status'' character is not set by default, so you must set it with stty(1) in order to use it) and will continue capturing packets. On platforms that do not support the SIGINFO signal, the same can be achieved by using the SIGUSR1 signal.

Reading packets from a network interface may require that you have special privileges; see the pcap (3PCAP) man page for details. Reading a saved packet file doesn't require special privileges.

OPTIONS

-A Print each packet (minus its link level header) in ASCII. Handy for capturing web pages.

-b Print the AS number in BGP packets in ASDOT notation rather than ASPLAIN notation.

-B buffer_size

--buffer-size=buffer_size
Set the operating system capture buffer size to buffer_size, in units of KiB (1024 bytes).

-c count
Exit after receiving count packets.

-C file_size
Before writing a raw packet to a savefile, check whether the file is currently larger than file_size and, if so, close the current savefile and open a new one. Savefiles after the first savefile will have the name specified with the -w flag, with a number after it, starting at 1 and continuing upward. The units of file_size are millions of bytes (1,000,000 bytes, not 1,048,576 bytes).

-d Dump the compiled packet-matching code in a human readable form to standard output and stop.

-dd Dump packet-matching code as a C program fragment.

-ddd Dump packet-matching code as decimal numbers (preceded with a count).

-D

--list-interfaces
Print the list of the network interfaces available on the system and on which tcpdump can capture packets. For each network

interface, a number and an interface name, possibly followed by a text description of the interface, is printed. The interface name or the number can be supplied to the -i flag to specify an interface on which to capture.

This can be useful on systems that don't have a command to list them (e.g., Windows systems, or UNIX systems lacking ifconfig -a); the number can be useful on Windows 2000 and later systems, where the interface name is a somewhat complex string.

The -D flag will not be supported if tcpdump was built with an older version of libpcap that lacks the pcap_findalldevs() function.

-e Print the link-level header on each dump line. This can be used, for example, to print MAC layer addresses for protocols such as Ethernet and IEEE 802.11.

-E Use spi@ipaddr algo:secret for decrypting IPsec ESP packets that are addressed to addr and contain Security Parameter Index value spi. This combination may be repeated with comma or newline separation.

Note that setting the secret for IPv4 ESP packets is supported at this time.

Algorithms may be des-cbc, 3des-cbc, blowfish-cbc, rc3-cbc, cast128-cbc, or none. The default is des-cbc. The ability to decrypt packets is only present if tcpdump was compiled with cryptography enabled.

secret is the ASCII text for ESP secret key. If preceded by 0x, then a hex value will be read.

The option assumes RFC2406 ESP, not RFC1827 ESP. The option is only for debugging purposes, and the use of this option with a true `secret' key is discouraged. By presenting IPsec secret key onto command line you make it visible to others, via ps(1) and other occasions.

In addition to the above syntax, the syntax file name may be used to have tcpdump read the provided file in. The file is opened upon receiving the first ESP packet, so any special permissions that tcpdump may have been given should already have been given up.

-f Print `foreign' IPv4 addresses numerically rather than symbolically (this option is intended to get around serious brain damage in Sun's NIS server -- usually it hangs forever translating non-local internet numbers).

The test for `foreign' IPv4 addresses is done using the IPv4 address and netmask of the interface on which capture is being done. If that address or netmask are not available, available, either because the interface on which capture is being done has no address or netmask or because the capture is being done on the Linux "any" interface, which can capture on more than one interface, this option will not work correctly.

-F file
       Use file as input for the filter expression. An additional
       expression given on the command line is ignored.

-G rotate_seconds
       If specified, rotates the dump file specified with the -w option
       every rotate_seconds seconds. Savefiles will have the name
       specified by -w which should include a time format as defined by
       strftime(3). If no time format is specified, each new file will
       overwrite the previous.

       If used in conjunction with the -C option, filenames will take
       the form of `file<count>'.

-h

--help Print the tcpdump and libpcap version strings, print a usage
       message, and exit.

--version
       Print the tcpdump and libpcap version strings and exit.

-H   Attempt to detect 802.11s draft mesh headers.

-i interface

--interface=interface
       Listen on interface. If unspecified, tcpdump searches the
       system interface list for the lowest numbered, configured up
       interface (excluding loopback), which may turn out to be, for
       example, ``eth0''.

       On Linux systems with 2.2 or later kernels, an interface
       argument of ``any'' can be used to capture packets from all
       interfaces. Note that captures on the ``any'' device will not
       be done in promiscuous mode.

       If the -D flag is supported, an interface number as printed by
       that flag can be used as the interface argument.

-I

--monitor-mode
       Put the interface in "monitor mode"; this is supported only on
       IEEE 802.11 Wi-Fi interfaces, and supported only on some
       operating systems.

       Note that in monitor mode the adapter might disassociate from
       the network with which it's associated, so that you will not be
       able to use any wireless networks with that adapter. This could
       prevent accessing files on a network server, or resolving host
       names or network addresses, if you are capturing in monitor mode
       and are not connected to another network with another adapter.

       This flag will affect the output of the -L flag. If -I isn't
       specified, only those link-layer types available when not in
       monitor mode will be shown; if -I is specified, only those link-
       layer types available when in monitor mode will be shown.

--immediate-mode
       Capture in "immediate mode". In this mode, packets are
       delivered to tcpdump as soon as they arrive, rather than being
       buffered for efficiency. This is the default when printing
       packets rather than saving packets to a ``savefile'' if the
       packets are being printed to a terminal rather than to a file or
       pipe.

-j tstamp_type

--time-stamp-type=tstamp_type
       Set the time stamp type for the capture to tstamp_type. The
       names to use for the time stamp types are given in
       pcap-tstamp(7); not all the types listed there will necessarily
       be valid for any given interface.

-J

--list-time-stamp-types
       List the supported time stamp types for the interface and exit.
       If the time stamp type cannot be set for the interface, no time
       stamp types are listed.

--time-stamp-precision=tstamp_precision
       When capturing, set the time stamp precision for the capture to
       tstamp_precision. Note that availability of high precision time
       stamps (nanoseconds) and their actual accuracy is platform and
       hardware dependent. Also note that when writing captures made
       with nanosecond accuracy to a savefile, the time stamps are
       written with nanosecond resolution, and the file is written with
       a different magic number, to indicate that the time stamps are
       in seconds and nanoseconds; not all programs that read pcap
       savefiles will be able to read those captures.

When reading a savefile, convert time stamps to the precision specified
by timestamp_precision, and display them with that resolution. If the
precision specified is less than the precision of time stamps in the
file, the conversion will lose precision.

The supported values for timestamp_precision are micro for microsecond
resolution and nano for nanosecond resolution. The default is
microsecond resolution.

-K

--dont-verify-checksums
       Don't attempt to verify IP, TCP, or UDP checksums. This is
       useful for interfaces that perform some or all of those checksum
       calculation in hardware; otherwise, all outgoing TCP checksums
       will be flagged as bad.

-l   Make stdout line buffered. Useful if you want to see the data
       while capturing it. E.g.,

              tcpdump -l | tee dat

       or

              tcpdump -l > dat & tail -f dat

Note that on Windows,``line buffered'' means ``unbuffered'', so that WinDump will write each character individually if -l is specified.

-U is similar to -l in its behavior, but it will cause output to be ``packet-buffered'', so that the output is written to stdout at the end of each packet rather than at the end of each line; this is buffered on all platforms, including Windows.

-L

--list-data-link-types
List the known data link types for the interface, in the specified mode, and exit. The list of known data link types may be dependent on the specified mode; for example, on some platforms, a Wi-Fi interface might support one set of data link types when not in monitor mode (for example, it might support only fake Ethernet headers, or might support 802.11 headers but not support 802.11 headers with radio information) and another set of data link types when in monitor mode (for example, it might support 802.11 headers, or 802.11 headers with radio information, only in monitor mode).

-m module
Load SMI MIB module definitions from file module. This option can be used several times to load several MIB modules into tcpdump.

-M secret
Use secret as a shared secret for validating the digests found in TCP segments with the TCP-MD5 option (RFC 2385), if present.

-n Don't convert addresses (i.e., host addresses, port numbers, etc.) to names.

-N Don't print domain name qualification of host names. E.g., if you give this flag then tcpdump will print ``nic'' instead of ``nic.ddn.mil''.

-#

--number
Print an optional packet number at the beginning of the line.

-O

--no-optimize
Do not run the packet-matching code optimizer. This is useful only if you suspect a bug in the optimizer.

-p

--no-promiscuous-mode
Don't put the interface into promiscuous mode. Note that the interface might be in promiscuous mode for some other reason; hence, `-p' cannot be used as an abbreviation for `ether host {local-hw-addr} or ether broadcast'.

-Q direction

--direction=direction
Choose send/receive direction direction for which packets should be captured. Possible values are `in', `out' and `inout'. Not available on all platforms.

-q Quick (quiet?) output. Print less protocol information so output lines are shorter.

-R Assume ESP/AH packets to be based on old specification (RFC1825 to RFC1829). If specified, tcpdump will not print replay prevention field. Since there is no protocol version field in ESP/AH specification, tcpdump cannot deduce the version of ESP/AH protocol.

-r file
Read packets from file (which was created with the -w option or by other tools that write pcap or pcap-ng files). Standard input is used if file is ``-''.

-S

--absolute-tcp-sequence-numbers
Print absolute, rather than relative, TCP sequence numbers.

-s snaplen

--snapshot-length=snaplen
Snarf snaplen bytes of data from each packet rather than the default of 65535 bytes. Packets truncated because of a limited snapshot are indicated in the output with ``[|proto]'', where proto is the name of the protocol level at which the truncation has occurred. Note that taking larger snapshots both increases the amount of time it takes to process packets and, effectively, decreases the amount of packet buffering. This may cause packets to be lost. You should limit snaplen to the smallest number that will capture the protocol information you're interested in. Setting snaplen to 0 sets it to the default of 65535, for backwards compatibility with recent older versions of tcpdump.

-T type
Force packets selected by "expression" to be interpreted the specified type. Currently known types are aodv (Ad-hoc On-demand Distance Vector protocol), carp (Common Address Redundancy Protocol), cnfp (Cisco NetFlow protocol), lmp (Link Management Protocol), pgm (Pragmatic General Multicast), pgm_zmtp1 (ZMTP/1.0 inside PGM/EPGM), radius (RADIUS), rpc (Remote Procedure Call), rtp (Real-Time Applications protocol), rtcp (Real-Time Applications control protocol), snmp (Simple Network Management Protocol), tftp (Trivial File Transfer Protocol), vat (Visual Audio Tool), wb (distributed White Board), zmtp1 (ZeroMQ Message Transport Protocol 1.0) and vxlan (Virtual eXtensible Local Area Network).

Note that the pgm type above affects UDP interpretation only, the native PGM is always recognised as IP protocol 113 regardless. UDP-encapsulated PGM is often called "EPGM" or

"PGM/UDP".

Note that the pgm_zmtp1 type above affects interpretation of both native PGM and UDP at once. During the native PGM decoding the application data of an ODATA/RDATA packet would be decoded as a ZeroMQ datagram with ZMTP/1.0 frames. During the UDP decoding in addition to that any UDP packet would be treated as an encapsulated PGM packet.

-t Don't print a timestamp on each dump line.

-tt Print the timestamp, as seconds since January 1, 1970, 00:00:00, UTC, and fractions of a second since that time, on each dump line.

-ttt Print a delta (micro-second resolution) between current and previous line on each dump line.

-tttt Print a timestamp, as hours, minutes, seconds, and fractions of a second since midnight, preceded by the date, on each dump line.

-ttttt Print a delta (micro-second resolution) between current and first line on each dump line.

-u Print undecoded NFS handles.

-U

--packet-buffered
If the -w option is not specified, make the printed packet output ``packet-buffered''; i.e., as the description of the contents of each packet is printed, it will be written to the standard output, rather than, when not writing to a terminal, being written only when the output buffer fills.

If the -w option is specified, make the saved raw packet output ``packet-buffered''; i.e., as each packet is saved, it will be written to the output file, rather than being written only when the output buffer fills.

The -U flag will not be supported if tcpdump was built with an older version of libpcap that lacks the pcap_dump_flush() function.

-v When parsing and printing, produce (slightly more) verbose output. For example, the time to live, identification, total length and options in an IP packet are printed. Also enables additional packet integrity checks such as verifying the IP and ICMP header checksum.

When writing to a file with the -w option, report, every 10 seconds, the number of packets captured.

-vv Even more verbose output. For example, additional fields are printed from NFS reply packets, and SMB packets are fully decoded.

-vvv Even more verbose output. For example, telnet SB ... SE options

are printed in full. With -X Telnet options are printed in hex as well.

-V file
Read a list of filenames from file. Standard input is used if file is ``-''.

-w file
Write the raw packets to file rather than parsing and printing them out. They can later be printed with the -r option. Standard output is used if file is ``-''.

This output will be buffered if written to a file or pipe, so a program reading from the file or pipe may not see packets for an arbitrary amount of time after they are received. Use the -U flag to cause packets to be written as soon as they are received.

The MIME type application/vnd.tcpdump.pcap has been registered with IANA for pcap files. The filename extension .pcap appears to be the most commonly used along with .cap and .dmp. Tcpdump itself doesn't check the extension when reading capture files and doesn't add an extension when writing them (it uses magic numbers in the file header instead). However, many operating systems and applications will use the extension if it is present and adding one (e.g. .pcap) is recommended.

See pcap-savefile(5) for a description of the file format.

-W Used in conjunction with the -C option, this will limit the number of files created to the specified number, and begin overwriting files from the beginning, thus creating a 'rotating' buffer. In addition, it will name the files with enough leading 0s to support the maximum number of files, allowing them to sort correctly.

Used in conjunction with the -G option, this will limit the number of rotated dump files that get created, exiting with status 0 when reaching the limit. If used with -C as well, the behavior will result in cyclical files per timeslice.

-x When parsing and printing, in addition to printing the headers of each packet, print the data of each packet (minus its link level header) in hex. The smaller of the entire packet or snaplen bytes will be printed. Note that this is the entire link-layer packet, so for link layers that pad (e.g. Ethernet), the padding bytes will also be printed when the higher layer packet is shorter than the required padding.

-xx When parsing and printing, in addition to printing the headers of each packet, print the data of each packet, including its link level header, in hex.

-X When parsing and printing, in addition to printing the headers of each packet, print the data of each packet (minus its link level header) in hex and ASCII. This is very handy for analysing new protocols.

-XX When parsing and printing, in addition to printing the headers

```
              of each packet, print the data of each packet, including its
              link level header, in hex and ASCII.

       -y datalinktype

       --linktype=datalinktype
              Set the data link type to use while capturing packets to
              datalinktype.

       -z postrotate-command
              Used in conjunction with the -C or -G options, this will make
              tcpdump run " postrotate-command file " where file is the
              savefile being closed after each rotation. For example,
              specifying -z gzip or -z bzip2 will compress each savefile using
              gzip or bzip2.

              Note that tcpdump will run the command in parallel to the
              capture, using the lowest priority so that this doesn't disturb
              the capture process.

              And in case you would like to use a command that itself takes
              flags or different arguments, you can always write a shell
              script that will take the savefile name as the only argument,
              make the flags & arguments arrangements and execute the command
              that you want.

       -Z user

       --relinquish-privileges=user
              If tcpdump is running as root, after opening the capture device
              or input savefile, but before opening any savefiles for output,
              change the user ID to user and the group ID to the primary group
              of user.

              This behavior can also be enabled by default at compile time.

        expression
              selects which packets will be dumped.  If no expression is
              given, all packets on the net will be dumped.  Otherwise, only
              packets for which expression is `true' will be dumped.

              For the expression syntax, see pcap-filter(7).

              The expression argument can be passed to tcpdump as either a
              single Shell argument, or as multiple Shell arguments, whichever
              is more convenient.  Generally, if the expression contains Shell
              metacharacters, such as backslashes used to escape protocol
              names, it is easier to pass it as a single, quoted argument
              rather than to escape the Shell metacharacters.  Multiple
              arguments are concatenated with spaces before being parsed.

EXAMPLES
       To print all packets arriving at or departing from sundown:
              tcpdump host sundown

       To print traffic between helios and either hot or ace:
              tcpdump host helios and \( hot or ace \)

       To print all IP packets between ace and any host except helios:
```

```
              tcpdump ip host ace and not helios

       To print all traffic between local hosts and hosts at Berkeley:
              tcpdump net ucb-ether

       To print all ftp traffic through internet gateway snup: (note that the
       expression is quoted to prevent the shell from (mis-)interpreting the
       parentheses):
              tcpdump 'gateway snup and (port ftp or ftp-data)'

       To print traffic neither sourced from nor destined for local hosts (if
       you gateway to one other net, this stuff should never make it onto your
       local net).
              tcpdump ip and not net localnet

       To print the start and end packets (the SYN and FIN packets) of each
       TCP conversation that involves a non-local host.
              tcpdump 'tcp[tcpflags] & (tcp-syn|tcp-fin) != 0 and not src and dst
net localnet'

       To print all IPv4 HTTP packets to and from port 80, i.e. print only
       packets that contain data, not, for example, SYN and FIN packets and
       ACK-only packets.  (IPv6 is left as an exercise for the reader.)
              tcpdump 'tcp port 80 and (((ip[2:2] - ((ip[0]&0xf)<<2)) -
((tcp[12]&0xf0)>>2)) != 0)'

       To print IP packets longer than 576 bytes sent through gateway snup:
              tcpdump 'gateway snup and ip[2:2] > 576'

       To print IP broadcast or multicast packets that were not sent via
       Ethernet broadcast or multicast:
              tcpdump 'ether[0] & 1 = 0 and ip[16] >= 224'

       To print all ICMP packets that are not echo requests/replies (i.e., not
       ping packets):
              tcpdump 'icmp[icmptype] != icmp-echo and icmp[icmptype] != icmp-
echoreply'

OUTPUT FORMAT
       The output of tcpdump is protocol dependent.  The following gives a
       brief description and examples of most of the formats.
       Link Level Headers

       If the '-e' option is given, the link level header is printed out.  On
       Ethernets, the source and destination addresses, protocol, and packet
       length are printed.

       On FDDI networks, the  '-e' option causes tcpdump to print the `frame
       control' field,  the source and destination addresses, and the packet
       length.  (The `frame control' field governs the interpretation of the
       rest of the packet.  Normal packets (such as those containing IP
       datagrams) are `async' packets, with a priority value between 0 and 7;
       for example, `async4'.  Such packets are assumed to contain an 802.2
       Logical Link Control (LLC) packet; the LLC header is printed if it is
       not an ISO datagram or a so-called SNAP packet.

       On Token Ring networks, the '-e' option causes tcpdump to print the
       `access control' and `frame control' fields, the source and destination
       addresses, and the packet length.  As on FDDI networks, packets are
```

assumed to contain an LLC packet. Regardless of whether the '-e' option is specified or not, the source routing information is printed for source-routed packets.

On 802.11 networks, the '-e' option causes tcpdump to print the `frame control' fields, all of the addresses in the 802.11 header, and the packet length. As on FDDI networks, packets are assumed to contain an LLC packet.

(N.B.: The following description assumes familiarity with the SLIP compression algorithm described in RFC-1144.)

On SLIP links, a direction indicator (``I'' for inbound, ``O'' for outbound), packet type, and compression information are printed out. The packet type is printed first. The three types are ip, utcp, and ctcp. No further link information is printed for ip packets. For TCP packets, the connection identifier is printed following the type. If the packet is compressed, its encoded header is printed out. The special cases are printed out as *S+n and *SA+n, where n is the amount by which the sequence number (or sequence number and ack) has changed. If it is not a special case, zero or more changes are printed. A change is indicated by U (urgent pointer), W (window), A (ack), S (sequence number), and I (packet ID), followed by a delta (+n or -n), or a new value (=n). Finally, the amount of data in the packet and compressed header length are printed.

For example, the following line shows an outbound compressed TCP packet, with an implicit connection identifier; the ack has changed by 6, the sequence number by 49, and the packet ID by 6; there are 3 bytes of data and 6 bytes of compressed header:

```
O ctcp * A+6 S+49 I+6 3 (6)
```

ARP/RARP Packets

Arp/rarp output shows the type of request and its arguments. The format is intended to be self explanatory. Here is a short sample taken from the start of an `rlogin' from host rtsg to host csam:

```
arp who-has csam tell rtsg
arp reply csam is-at CSAM
```

The first line says that rtsg sent an arp packet asking for the Ethernet address of internet host csam. Csam replies with its Ethernet address (in this example, Ethernet addresses are in caps and internet addresses in lower case).

This would look less redundant if we had done tcpdump -n:

```
arp who-has 128.3.254.6 tell 128.3.254.68
arp reply 128.3.254.6 is-at 02:07:01:00:01:c4
```

If we had done tcpdump -e, the fact that the first packet is broadcast and the second is point-to-point would be visible:

```
RTSG Broadcast 0806  64: arp who-has csam tell rtsg
CSAM RTSG 0806  64: arp reply csam is-at CSAM
```

For the first packet this says the Ethernet source address is RTSG, the destination is the Ethernet broadcast address, the type field contained hex 0806 (type ETHER_ARP) and the total length was 64 bytes.

TCP Packets

(N.B.:The following description assumes familiarity with the TCP protocol described in RFC-793. If you are not familiar with the protocol, neither this description nor tcpdump will be of much use to

you.)

The general format of a tcp protocol line is:

```
src > dst: flags data-seqno ack window urgent options
```

Src and dst are the source and destination IP addresses and ports. Flags are some combination of S (SYN), F (FIN), P (PUSH), R (RST), U (URG), W (ECN CWR), E (ECN-Echo) or `.' (ACK), or `none' if no flags are set. Data-seqno describes the portion of sequence space covered by the data in this packet (see example below). Ack is sequence number of the next data expected the other direction on this connection. Window is the number of bytes of receive buffer space available the other direction on this connection. Urg indicates there is `urgent' data in the packet. Options are tcp options enclosed in angle brackets (e.g., <mss 1024>).

Src, dst and flags are always present. The other fields depend on the contents of the packet's tcp protocol header and are output only if appropriate.

Here is the opening portion of an rlogin from host rtsg to host csam.

```
rtsg.1023 > csam.login: S 768512:768512(0) win 4096 <mss 1024>
csam.login > rtsg.1023: S 947648:947648(0) ack 768513 win 4096 <mss
1024>
rtsg.1023 > csam.login: . ack 1 win 4096
rtsg.1023 > csam.login: P 1:2(1) ack 1 win 4096
csam.login > rtsg.1023: . ack 2 win 4096
rtsg.1023 > csam.login: P 2:21(19) ack 1 win 4096
csam.login > rtsg.1023: P 1:2(1) ack 21 win 4077
csam.login > rtsg.1023: P 2:3(1) ack 21 win 4077 urg 1
csam.login > rtsg.1023: P 3:4(1) ack 21 win 4077 urg 1
```

The first line says that tcp port 1023 on rtsg sent a packet to port login on csam. The S indicates that the SYN flag was set. The packet sequence number was 768512 and it contained no data. (The notation is `first:last(nbytes)' which means `sequence numbers first up to but not including last which is nbytes bytes of user data'.) There was no piggy-backed ack, the available receive window was 4096 bytes and there was a max-segment-size option requesting an mss of 1024 bytes.

Csam replies with a similar packet except it includes a piggy-backed ack for rtsg's SYN. Rtsg then acks csam's SYN. The `.' means the ACK flag was set. The packet contained no data so there is no data sequence number. Note that the ack sequence number is a small integer (1). The first time tcpdump sees a tcp `conversation', it prints the sequence number from the packet. On subsequent packets of the conversation, the difference between the current packet's sequence number and this initial sequence number is printed. This means that sequence numbers after the first can be interpreted as relative byte positions in the conversation's data stream (with the first data byte each direction being `1'). `-S' will override this feature, causing the original sequence numbers to be output.

On the 6th line, rtsg sends csam 19 bytes of data (bytes 2 through 20 in the rtsg -> csam side of the conversation). The PUSH flag is set in the packet. On the 7th line, csam says it's received data sent by rtsg up to but not including byte 21. Most of this data is apparently sitting in the socket buffer since csam's receive window has gotten 19 bytes smaller. Csam also sends one byte of data to rtsg in this packet. On the 8th and 9th lines, csam sends two bytes of urgent, pushed data to rtsg.

If the snapshot was small enough that tcpdump didn't capture the full TCP header, it interprets as much of the header as it can and then reports ``[|tcp]'' to indicate the remainder could not be interpreted. If the header contains a bogus option (one with a length that's either too small or beyond the end of the header), tcpdump reports it as ``[bad opt]'' and does not interpret any further options (since it's impossible to tell where they start). If the header length indicates options are present but the IP datagram length is not long enough for the options to actually be there, tcpdump reports it as ``[bad hdr length]''.
Capturing TCP packets with particular flag combinations (SYN-ACK, URG-ACK, etc.)

There are 8 bits in the control bits section of the TCP header:

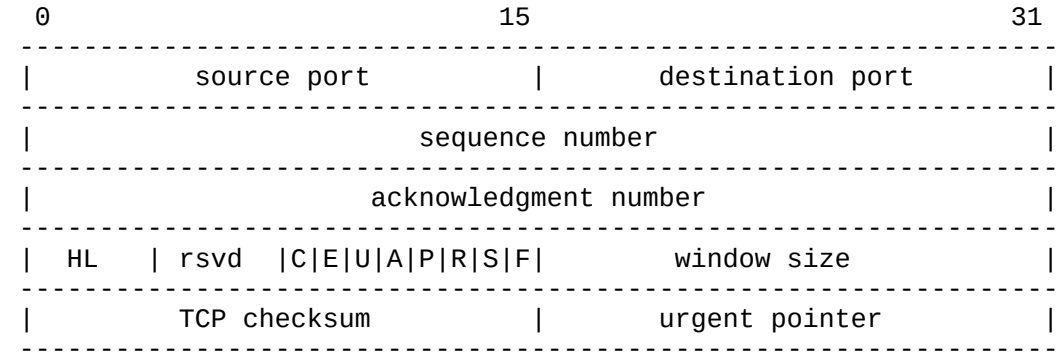
CWR | ECE | URG | ACK | PSH | RST | SYN | FIN

Let's assume that we want to watch packets used in establishing a TCP connection. Recall that TCP uses a 3-way handshake protocol when it initializes a new connection; the connection sequence with regard to the TCP control bits is

1) Caller sends SYN
2) Recipient responds with SYN, ACK
3) Caller sends ACK

Now we're interested in capturing packets that have only the SYN bit set (Step 1). Note that we don't want packets from step 2 (SYN-ACK), just a plain initial SYN. What we need is a correct filter expression for tcpdump.

Recall the structure of a TCP header without options:



A TCP header usually holds 20 octets of data, unless options are present. The first line of the graph contains octets 0 - 3, the second line shows octets 4 - 7 etc.

Starting to count with 0, the relevant TCP control bits are contained in octet 13:

Let's have a closer look at octet no. 13:

```
                |               |
                |---------------|
                |C|E|U|A|P|R|S|F|
                |---------------|
                |7   5   3     0|
```

These are the TCP control bits we are interested in. We have numbered the bits in this octet from 0 to 7, right to left, so the PSH bit is bit number 3, while the URG bit is number 5.

Recall that we want to capture packets with only SYN set. Let's see what happens to octet 13 if a TCP datagram arrives with the SYN bit set in its header:

```
                |C|E|U|A|P|R|S|F|
                |---------------|
                |0 0 0 0 0 0 1 0|
                |---------------|
                |7 6 5 4 3 2 1 0|
```

Looking at the control bits section we see that only bit number 1 (SYN) is set.

Assuming that octet number 13 is an 8-bit unsigned integer in network byte order, the binary value of this octet is

```
       00000010
```

and its decimal representation is

```
   7     6     5     4     3     2     1     0
0*2 + 0*2 + 0*2 + 0*2 + 0*2 + 0*2 + 1*2 + 0*2  =  2
```

We're almost done, because now we know that if only SYN is set, the value of the 13th octet in the TCP header, when interpreted as a 8-bit unsigned integer in network byte order, must be exactly 2.

This relationship can be expressed as

```
       tcp[13] == 2
```

We can use this expression as the filter for tcpdump in order to watch packets which have only SYN set:

```
       tcpdump -i xl0 tcp[13] == 2
```

The expression says "let the 13th octet of a TCP datagram have the decimal value 2", which is exactly what we want.

Now, let's assume that we need to capture SYN packets, but we don't care if ACK or any other TCP control bit is set at the same time. Let's see what happens to octet 13 when a TCP datagram with SYN-ACK set arrives:

```
     |C|E|U|A|P|R|S|F|
     |---------------|
     |0 0 0 1 0 0 1 0|
     |---------------|
```

```
     |7 6 5 4 3 2 1 0|
```

Now bits 1 and 4 are set in the 13th octet. The binary value of octet 13 is

```
            00010010
```

which translates to decimal

```
   7     6     5     4     3     2     1     0
0*2 + 0*2 + 0*2 + 1*2 + 0*2 + 0*2 + 1*2 + 0*2   = 18
```

Now we can't just use 'tcp[13] == 18' in the tcpdump filter expression, because that would select only those packets that have SYN-ACK set, but not those with only SYN set. Remember that we don't care if ACK or any other control bit is set as long as SYN is set.

In order to achieve our goal, we need to logically AND the binary value of octet 13 with some other value to preserve the SYN bit. We know that we want SYN to be set in any case, so we'll logically AND the value in the 13th octet with the binary value of a SYN:

```
          00010010 SYN-ACK              00000010 SYN
     AND  00000010 (we want SYN)   AND  00000010 (we want SYN)
          --------                      --------
     =    00000010                 =    00000010
```

We see that this AND operation delivers the same result regardless whether ACK or another TCP control bit is set. The decimal representation of the AND value as well as the result of this operation is 2 (binary 00000010), so we know that for packets with SYN set the following relation must hold true:

```
       ( ( value of octet 13 ) AND ( 2 ) ) == ( 2 )
```

This points us to the tcpdump filter expression

```
            tcpdump -i xl0 'tcp[13] & 2 == 2'
```

Some offsets and field values may be expressed as names rather than as numeric values. For example tcp[13] may be replaced with tcp[tcpflags]. The following TCP flag field values are also available: tcp-fin, tcp-syn, tcp-rst, tcp-push, tcp-act, tcp-urg.

This can be demonstrated as:

```
            tcpdump -i xl0 'tcp[tcpflags] & tcp-push != 0'
```

Note that you should use single quotes or a backslash in the expression to hide the AND ('&') special character from the shell.

UDP Packets

UDP format is illustrated by this rwho packet:

```
       actinide.who > broadcast.who: udp 84
```

This says that port who on host actinide sent a udp datagram to port who on host broadcast, the Internet broadcast address. The packet contained 84 bytes of user data.

Some UDP services are recognized (from the source or destination port number) and the higher level protocol information printed. In particular, Domain Name service requests (RFC-1034/1035) and Sun RPC

calls (RFC-1050) to NFS.

UDP Name Server Requests

(N.B.:The following description assumes familiarity with the Domain Service protocol described in RFC-1035. If you are not familiar with the protocol, the following description will appear to be written in greek.)

Name server requests are formatted as

```
src > dst: id op? flags qtype qclass name (len)
h2opolo.1538 > helios.domain: 3+ A? ucbvax.berkeley.edu. (37)
```

Host h2opolo asked the domain server on helios for an address record (qtype=A) associated with the name ucbvax.berkeley.edu. The query id was `3'. The `+' indicates the recursion desired flag was set. The query length was 37 bytes, not including the UDP and IP protocol headers. The query operation was the normal one, Query, so the op field was omitted. If the op had been anything else, it would have been printed between the `3' and the `+'. Similarly, the qclass was the normal one, C_IN, and omitted. Any other qclass would have been printed immediately after the `A'.

A few anomalies are checked and may result in extra fields enclosed in square brackets: If a query contains an answer, authority records or additional records section, ancount, nscount, or arcount are printed as `[na]', `[nn]' or `[nau]' where n is the appropriate count. If any of the response bits are set (AA, RA or rcode) or any of the `must be zero' bits are set in bytes two and three, `[b2&3=x]' is printed, where x is the hex value of header bytes two and three.

UDP Name Server Responses

Name server responses are formatted as

```
src > dst: id op rcode flags a/n/au type class data (len)
helios.domain > h2opolo.1538: 3 3/3/7 A 128.32.137.3 (273)
helios.domain > h2opolo.1537: 2 NXDomain* 0/1/0 (97)
```

In the first example, helios responds to query id 3 from h2opolo with 3 answer records, 3 name server records and 7 additional records. The first answer record is type A (address) and its data is internet address 128.32.137.3. The total size of the response was 273 bytes, excluding UDP and IP headers. The op (Query) and response code (NoError) were omitted, as was the class (C_IN) of the A record.

In the second example, helios responds to query 2 with a response code of non-existent domain (NXDomain) with no answers, one name server and no authority records. The `*' indicates that the authoritative answer bit was set. Since there were no answers, no type, class or data were printed.

Other flag characters that might appear are `-' (recursion available, RA, not set) and `|' (truncated message, TC, set). If the `question' section doesn't contain exactly one entry, `[nq]' is printed.

SMB/CIFS decoding

tcpdump now includes fairly extensive SMB/CIFS/NBT decoding for data on UDP/137, UDP/138 and TCP/139. Some primitive decoding of IPX and NetBEUI SMB data is also done.

By default a fairly minimal decode is done, with a much more detailed decode done if -v is used. Be warned that with -v a single SMB packet may take up a page or more, so only use -v if you really want all the

gory details.

For information on SMB packet formats and what all the fields mean see www.cifs.org or the pub/samba/specs/ directory on your favorite samba.org mirror site. The SMB patches were written by Andrew Tridgell (tridge@samba.org).

NFS Requests and Replies

Sun NFS (Network File System) requests and replies are printed as:

```
src.sport > dst.nfs: NFS request xid xid len op args
src.nfs > dst.dport: NFS reply xid xid reply stat len op results

sushi.1023 > wrl.nfs: NFS request xid 26377
     112 readlink fh 21,24/10.73165
wrl.nfs > sushi.1023: NFS reply xid 26377
     reply ok 40 readlink "../var"
sushi.1022 > wrl.nfs: NFS request xid 8219
     144 lookup fh 9,74/4096.6878 "xcolors"
wrl.nfs > sushi.1022: NFS reply xid 8219
     reply ok 128 lookup fh 9,74/4134.3150
```

In the first line, host sushi sends a transaction with id 26377 to wrl. The request was 112 bytes, excluding the UDP and IP headers. The operation was a readlink (read symbolic link) on file handle (fh) 21,24/10.731657119. (If one is lucky, as in this case, the file handle can be interpreted as a major,minor device number pair, followed by the inode number and generation number.) In the second line, wrl replies `ok' with the same transaction id and the contents of the link.

In the third line, sushi asks (using a new transaction id) wrl to lookup the name `xcolors' in directory file 9,74/4096.6878. In the fourth line, wrl sends a reply with the respective transaction id.

Note that the data printed depends on the operation type. The format is intended to be self explanatory if read in conjunction with an NFS protocol spec. Also note that older versions of tcpdump printed NFS packets in a slightly different format: the transaction id (xid) would be printed instead of the non-NFS port number of the packet.

If the -v (verbose) flag is given, additional information is printed. For example:

```
sushi.1023 > wrl.nfs: NFS request xid 79658
     148 read fh 21,11/12.195 8192 bytes @ 24576
wrl.nfs > sushi.1023: NFS reply xid 79658
     reply ok 1472 read REG 100664 ids 417/0 sz 29388
```

(-v also prints the IP header TTL, ID, length, and fragmentation fields, which have been omitted from this example.) In the first line, sushi asks wrl to read 8192 bytes from file 21,11/12.195, at byte offset 24576. Wrl replies `ok'; the packet shown on the second line is the first fragment of the reply, and hence is only 1472 bytes long (the other bytes will follow in subsequent fragments, but these fragments do not have NFS or even UDP headers and so might not be printed, depending on the filter expression used). Because the -v flag is given, some of the file attributes (which are returned in addition to the file data) are printed: the file type (``REG'', for regular file), the file mode (in octal), the uid and gid, and the file size.

If the -v flag is given more than once, even more details are printed.

Note that NFS requests are very large and much of the detail won't be printed unless snaplen is increased. Try using `-s 192' to watch NFS traffic.

NFS reply packets do not explicitly identify the RPC operation. Instead, tcpdump keeps track of ``recent'' requests, and matches them to the replies using the transaction ID. If a reply does not closely follow the corresponding request, it might not be parsable.

AFS Requests and Replies

Transarc AFS (Andrew File System) requests and replies are printed as:

```
src.sport > dst.dport: rx packet-type
src.sport > dst.dport: rx packet-type service call call-name args
src.sport > dst.dport: rx packet-type service reply call-name args

elvis.7001 > pike.afsfs:
     rx data fs call rename old fid 536876964/1/1 ".newsrc.new"
     new fid 536876964/1/1 ".newsrc"
pike.afsfs > elvis.7001: rx data fs reply rename
```

In the first line, host elvis sends a RX packet to pike. This was a RX data packet to the fs (fileserver) service, and is the start of an RPC call. The RPC call was a rename, with the old directory file id of 536876964/1/1 and an old filename of `.newsrc.new', and a new directory file id of 536876964/1/1 and a new filename of `.newsrc'. The host pike responds with a RPC reply to the rename call (which was successful, because it was a data packet and not an abort packet).

In general, all AFS RPCs are decoded at least by RPC call name. Most AFS RPCs have at least some of the arguments decoded (generally only the `interesting' arguments, for some definition of interesting).

The format is intended to be self-describing, but it will probably not be useful to people who are not familiar with the workings of AFS and RX.

If the -v (verbose) flag is given twice, acknowledgement packets and additional header information is printed, such as the RX call ID, call number, sequence number, serial number, and the RX packet flags.

If the -v flag is given twice, additional information is printed, such as the RX call ID, serial number, and the RX packet flags. The MTU negotiation information is also printed from RX ack packets.

If the -v flag is given three times, the security index and service id are printed.

Error codes are printed for abort packets, with the exception of Ubik beacon packets (because abort packets are used to signify a yes vote for the Ubik protocol).

Note that AFS requests are very large and many of the arguments won't be printed unless snaplen is increased. Try using `-s 256' to watch AFS traffic.

AFS reply packets do not explicitly identify the RPC operation. Instead, tcpdump keeps track of ``recent'' requests, and matches them

to the replies using the call number and service ID. If a reply does not closely follow the corresponding request, it might not be parsable.

KIP AppleTalk (DDP in UDP)

AppleTalk DDP packets encapsulated in UDP datagrams are de-encapsulated and dumped as DDP packets (i.e., all the UDP header information is discarded). The file /etc/atalk.names is used to translate AppleTalk net and node numbers to names. Lines in this file have the form

```
number    name

1.254          ether
16.1      icsd-net
1.254.110 ace
```

The first two lines give the names of AppleTalk networks. The third line gives the name of a particular host (a host is distinguished from a net by the 3rd octet in the number - a net number must have two octets and a host number must have three octets.) The number and name should be separated by whitespace (blanks or tabs). The /etc/atalk.names file may contain blank lines or comment lines (lines starting with a `#').

AppleTalk addresses are printed in the form

```
net.host.port

144.1.209.2 > icsd-net.112.220
office.2 > icsd-net.112.220
jssmag.149.235 > icsd-net.2
```

(If the /etc/atalk.names doesn't exist or doesn't contain an entry for some AppleTalk host/net number, addresses are printed in numeric form.) In the first example, NBP (DDP port 2) on net 144.1 node 209 is sending to whatever is listening on port 220 of net icsd node 112. The second line is the same except the full name of the source node is known (`office'). The third line is a send from port 235 on net jssmag node 149 to broadcast on the icsd-net NBP port (note that the broadcast address (255) is indicated by a net name with no host number - for this reason it's a good idea to keep node names and net names distinct in /etc/atalk.names).

NBP (name binding protocol) and ATP (AppleTalk transaction protocol) packets have their contents interpreted. Other protocols just dump the protocol name (or number if no name is registered for the protocol) and packet size.

NBP packets are formatted like the following examples:

```
icsd-net.112.220 > jssmag.2: nbp-lkup 190: "=:LaserWriter@*"
jssmag.209.2 > icsd-net.112.220: nbp-reply 190:
"RM1140:LaserWriter@*" 250
techpit.2 > icsd-net.112.220: nbp-reply 190:
"techpit:LaserWriter@*" 186
```

The first line is a name lookup request for laserwriters sent by net icsd host 112 and broadcast on net jssmag. The nbp id for the lookup is 190. The second line shows a reply for this request (note that it has the same id) from host jssmag.209 saying that it has a laserwriter resource named "RM1140" registered on port 250. The third line is another reply to the same request saying host techpit has laserwriter "techpit" registered on port 186.

ATP packet formatting is demonstrated by the following example:

```
            jssmag.209.165 > helios.132: atp-req  12266<0-7> 0xae030001
            helios.132 > jssmag.209.165: atp-resp 12266:0 (512) 0xae040000
            helios.132 > jssmag.209.165: atp-resp 12266:1 (512) 0xae040000
            helios.132 > jssmag.209.165: atp-resp 12266:2 (512) 0xae040000
            helios.132 > jssmag.209.165: atp-resp 12266:3 (512) 0xae040000
            helios.132 > jssmag.209.165: atp-resp 12266:4 (512) 0xae040000
            helios.132 > jssmag.209.165: atp-resp 12266:5 (512) 0xae040000
            helios.132 > jssmag.209.165: atp-resp 12266:6 (512) 0xae040000
            helios.132 > jssmag.209.165: atp-resp*12266:7 (512) 0xae040000
            jssmag.209.165 > helios.132: atp-req  12266<3,5> 0xae030001
            helios.132 > jssmag.209.165: atp-resp 12266:3 (512) 0xae040000
            helios.132 > jssmag.209.165: atp-resp 12266:5 (512) 0xae040000
            jssmag.209.165 > helios.132: atp-rel  12266<0-7> 0xae030001
            jssmag.209.133 > helios.132: atp-req* 12267<0-7> 0xae030002
```

Jssmag.209 initiates transaction id 12266 with host helios by requesting up to 8 packets (the `<0-7>'). The hex number at the end of the line is the value of the `userdata' field in the request.

Helios responds with 8 512-byte packets. The `:digit' following the transaction id gives the packet sequence number in the transaction and the number in parens is the amount of data in the packet, excluding the atp header. The `*' on packet 7 indicates that the EOM bit was set.

Jssmag.209 then requests that packets 3 & 5 be retransmitted. Helios resends them then jssmag.209 releases the transaction. Finally, jssmag.209 initiates the next request. The `*' on the request indicates that XO (`exactly once') was not set.

IP Fragmentation

Fragmented Internet datagrams are printed as

```
            (frag id:size@offset+)
            (frag id:size@offset)
```

(The first form indicates there are more fragments. The second indicates this is the last fragment.)

Id is the fragment id. Size is the fragment size (in bytes) excluding the IP header. Offset is this fragment's offset (in bytes) in the original datagram.

The fragment information is output for each fragment. The first fragment contains the higher level protocol header and the frag info is printed after the protocol info. Fragments after the first contain no higher level protocol header and the frag info is printed after the source and destination addresses. For example, here is part of an ftp from arizona.edu to lbl-rtsg.arpa over a CSNET connection that doesn't appear to handle 576 byte datagrams:

```
            arizona.ftp-data > rtsg.1170: . 1024:1332(308) ack 1 win 4096 (frag
595a:328@0+)
            arizona > rtsg: (frag 595a:204@328)
            rtsg.1170 > arizona.ftp-data: . ack 1536 win 2560
```

There are a couple of things to note here: First, addresses in the 2nd line don't include port numbers. This is because the TCP protocol information is all in the first fragment and we have no idea what the port or sequence numbers are when we print the later fragments. Second, the tcp sequence information in the first line is printed as if there were 308 bytes of user data when, in fact, there are 512 bytes (308 in the first frag and 204 in the second). If you are looking for holes in the sequence space or trying to match up acks with packets,

this can fool you.

A packet with the IP don't fragment flag is marked with a trailing (DF).

Timestamps

By default, all output lines are preceded by a timestamp. The timestamp is the current clock time in the form

hh:mm:ss.frac

and is as accurate as the kernel's clock. The timestamp reflects the time the kernel first saw the packet. No attempt is made to account for the time lag between when the Ethernet interface removed the packet from the wire and when the kernel serviced the `new packet' interrupt.

SEE ALSO

stty(1), pcap(3PCAP), bpf(4), nit(4P), pcap-savefile(5), pcap-filter(7), pcap-tstamp(7)

http://www.iana.org/assignments/media-types/application/vnd.tcpdump.pcap

AUTHORS

The original authors are:

Van Jacobson, Craig Leres and Steven McCanne, all of the Lawrence Berkeley National Laboratory, University of California, Berkeley, CA.

It is currently being maintained by tcpdump.org.

The current version is available via http:

http://www.tcpdump.org/

The original distribution is available via anonymous ftp:

ftp://ftp.ee.lbl.gov/old/tcpdump.tar.Z

IPv6/IPsec support is added by WIDE/KAME project. This program uses Eric Young's SSLeay library, under specific configurations.

BUGS

Please send problems, bugs, questions, desirable enhancements, patches etc. to:

tcpdump-workers@lists.tcpdump.org

NIT doesn't let you watch your own outbound traffic, BPF will. We recommend that you use the latter.

On Linux systems with 2.0[.x] kernels:

packets on the loopback device will be seen twice;

packet filtering cannot be done in the kernel, so that all packets must be copied from the kernel in order to be filtered in user mode;

all of a packet, not just the part that's within the snapshot length, will be copied from the kernel (the 2.0[.x] packet

capture mechanism, if asked to copy only part of a packet to userland, will not report the true length of the packet; this would cause most IP packets to get an error from tcpdump);

capturing on some PPP devices won't work correctly.

We recommend that you upgrade to a 2.2 or later kernel.

Some attempt should be made to reassemble IP fragments or, at least to compute the right length for the higher level protocol.

Name server inverse queries are not dumped correctly: the (empty) question section is printed rather than real query in the answer section. Some believe that inverse queries are themselves a bug and prefer to fix the program generating them rather than tcpdump.

A packet trace that crosses a daylight savings time change will give skewed time stamps (the time change is ignored).

Filter expressions on fields other than those in Token Ring headers will not correctly handle source-routed Token Ring packets.

Filter expressions on fields other than those in 802.11 headers will not correctly handle 802.11 data packets with both To DS and From DS set.

ip6 proto should chase header chain, but at this moment it does not. ip6 protochain is supplied for this behavior.

Arithmetic expression against transport layer headers, like tcp[0], does not work against IPv6 packets. It only looks at IPv4 packets.

MTR

```
MTR(8)                                  mtr                                  MTR(8)

NAME
       mtr - a network diagnostic tool

SYNOPSIS
       mtr [-4|-6] [-F FILENAME] [--report] [--report-wide] [--xml] [--gtk]
       [--curses] [--displaymode MODE] [--raw] [--csv] [--json] [--split]
       [--no-dns] [--show-ips] [-o FIELDS] [-y IPINFO] [--aslookup]
       [-i INTERVAL] [-c COUNT] [-s PACKETSIZE] [-B BITPATTERN]
       [-G GRACEPERIOD] [-Q TOS] [--mpls] [-a ADDRESS] [-f FIRST-TTL]
       [-m MAX-TTL] [-U MAX-UNKNOWN] [--udp] [--tcp] [-P PORT] [-L LOCALPORT]
       [-Z TIMEOUT] [-M MARK] HOSTNAME

DESCRIPTION
       mtr combines the functionality of the traceroute and ping programs in a
       single network diagnostic tool.

       As mtr starts, it investigates the network connection between the host
       mtr runs on and HOSTNAME by sending packets with purposely low TTLs.
       It continues to send packets with low TTL, noting the response time of
       the intervening routers.  This allows mtr to print the response
       percentage and response times of the internet route to HOSTNAME. A
       sudden increase in packet loss or response time is often an indication
       of a bad (or simply overloaded) link.

       The results are usually reported as round-trip-response times in
       milliseconds and the percentage of packetloss.

OPTIONS
       -h, --help
              Print the summary of command line argument options.

       -v, --version
              Print the installed version of mtr.

       -4     Use IPv4 only.

       -6     Use IPv6 only.  (IPV4 may be used for DNS lookups.)

       -F FILENAME, --filename FILENAME
              Reads the list of hostnames from the specified file.

       -r, --report
              This option puts mtr into report mode.  When in this mode, mtr
              will run for the number of cycles specified by the -c option,
              and then print statistics and exit.

              This mode is useful for generating statistics about network
              quality.  Note that each running instance of mtr generates a
              significant amount of network traffic.  Using mtr to measure the
              quality of your network may result in decreased network
              performance.
```

```
       -w, --report-wide
              This option puts mtr into wide report mode.  When in this mode,
              mtr will not cut hostnames in the report.

       -x, --xml
              Use this option to tell mtr to use the xml output format.  This
              format is better suited for automated processing of the
              measurement results.

       -t, --curses
              Use this option to force mtr to use the curses based terminal
              interface (if available).

       --displaymode MODE
              Use this option to select the initial display mode: 0 (default)
              selects statistics, 1 selects the stripchart without latency
              information, and 2 selects the stripchart with latency
              information.

       -g, --gtk
              Use this option to force mtr to use the GTK+ based X11 window
              interface (if available).  GTK+ must have been available on the
              system when mtr was built for this to work.  See the GTK+ web
              page at http://www.gtk.org/ for more information about GTK+.

       -l, --raw
              Use the raw output format.  This format is better suited for
              archival of the measurement results.  It could be parsed to be
              presented into any of the other display methods.

              Example of the raw output format:
              h 0 10.1.1.1
              p 0 339
              h 1 46.149.16.4
              p 1 530
              h 2 172.31.1.16
              p 2 531
              h 3 82.221.168.236
              p 3 1523
              h 5 195.130.211.8
              p 5 1603
              h 6 193.4.58.17
              p 6 1127
              h 7 193.4.58.17
              d 7 www.isnic.is

       -C, --csv
              Use the Comma-Separated-Value (CSV) output format.  (Note: The
              separator is actually a semi-colon ';'.)

              Example of the CSV output format:
              MTR.0.86+git:16e39fc0;1435562787;OK;nic.is;1;r-76520-
PROD.greenqloud.internal;288
              MTR.0.86+git:16e39fc0;1435562787;OK;nic.is;2;46.149.16.4;2086
              MTR.0.86+git:16e39fc0;1435562787;OK;nic.is;3;172.31.1.16;600
              MTR.0.86+git:16e39fc0;1435562787;OK;nic.is;4;82.221.168.236;1163
              MTR.0.86+git:16e39fc0;1435562787;OK;nic.is;5;???;0
              MTR.0.86+git:16e39fc0;1435562787;OK;nic.is;6;rix-k2-
```

```
gw.isnic.is;1654
              MTR.0.86+git:16e39fc0;1435562787;OK;nic.is;7;www.isnic.is;1036

       -j, --json
              Use this option to tell mtr to use the JSON output format.  This
              format is better suited for automated processing of the
              measurement results.

       -p, --split
              Use this option to set mtr to spit out a format that is suitable
              for a split-user interface.

       -n, --no-dns
              Use this option to force mtr to display numeric IP numbers and
              not try to resolve the host names.

       -b, --show-ips
              Use this option to tell mtr to display both the host names and
              numeric IP numbers.  In split mode this adds an extra field to
              the output.  In report mode, there is usually too little space
              to add the IPs, and they will be truncated.  Use the wide report
              (-w) mode to see the IPs in report mode.

       -o FIELDS, --order FIELDS
              Use this option to specify which fields to display and in which
              order.  You may use one or more space characters to separate
              fields.
              Available fields:

              Example: -o "LSD NBAW  X"

       -y n, --ipinfo n
              Displays information about each IP hop.  Valid values for n are:

              It is possible to cycle between these fields at runtime (using
              the y key).

       -z, --aslookup
              Displays the Autonomous System (AS) number alongside each hop.
              Equivalent to --ipinfo 0.

              Example (columns to the right not shown for clarity):
              1. AS???   r-76520-PROD.greenqloud.internal
              2. AS51969 46.149.16.4
              3. AS???   172.31.1.16
              4. AS30818 82.221.168.236
              5. ???
              6. AS???   rix-k2-gw.isnic.is
              7. AS1850  www.isnic.is

       -i SECONDS, --interval SECONDS
              Use this option to specify the positive number of seconds
              between ICMP ECHO requests.  The default value for this
              parameter is one second.  The root user may choose values
              between zero and one.

       -c COUNT, --report-cycles COUNT
              Use this option to set the number of pings sent to determine
              both the machines on the network and the reliability of those
```

machines. Each cycle lasts one second.

-s PACKETSIZE, --psize PACKETSIZE
This option sets the packet size used for probing. It is in bytes, inclusive IP and ICMP headers.

If set to a negative number, every iteration will use a different, random packet size up to that number.

-B NUM, --bitpattern NUM
Specifies bit pattern to use in payload. Should be within range 0 - 255. If NUM is greater than 255, a random pattern is used.

-G SECONDS, --graceperiod SECONDS
Use this option to specify the positive number of seconds to wait for responses after the final request. The default value is five seconds.

-Q NUM, --tos NUM
Specifies value for type of service field in IP header. Should be within range 0 - 255.

-e, --mpls
Use this option to tell mtr to display information from ICMP extensions for MPLS (RFC 4950) that are encoded in the response packets.

-a ADDRESS, --address ADDRESS
Use this option to bind the outgoing socket to ADDRESS, so that all packets will be sent with ADDRESS as source address. NOTE that this option doesn't apply to DNS requests (which could be and could not be what you want).

-f NUM, --first-ttl NUM
Specifies with what TTL to start. Defaults to 1.

-m NUM, --max-ttl NUM
Specifies the maximum number of hops (max time-to-live value) traceroute will probe. Default is 30.

-U NUM, --max-unknown NUM
Specifies the maximum unknown host. Default is 5.

-u, --udp
Use UDP datagrams instead of ICMP ECHO.

-T, --tcp
Use TCP SYN packets instead of ICMP ECHO. PACKETSIZE is ignored, since SYN packets can not contain data.

-P PORT, --port PORT
The target port number for TCP/SCTP/UDP traces.

-L LOCALPORT, --localport LOCALPORT
The source port number for UDP traces.

-Z SECONDS, --timeout SECONDS
The number of seconds to keep the TCP socket open before giving up on the connection. This will only affect the final hop.

```
              Using large values for this, especially combined with a short
              interval, will use up a lot of file descriptors.

       -M MARK, --mark MARK
              MISSING

ENVIRONMENT
       mtr recognizes a few environment variables.

       MTR_OPTIONS
              This environment variable allows to specify options, as if they
              were passed on the command line.  It is parsed before reading
              the actual command line options, so that options specified in
              MTR_OPTIONS are overridden by command-line options.

              Example:

              MTR_OPTIONS="-4 -c 1" mtr -6 localhost

              would send one probe (because of -c 1) towards ::1 (because of
              -6, which overrides the -4 passed in MTR_OPTIONS).

       DISPLAY
              Used for the GTK+ frontend.

BUGS
       Some modern routers give a lower priority to ICMP ECHO packets than to
       other network traffic.  Consequently, the reliability of these routers
       reported by mtr will be significantly lower than the actual reliability
       of these routers.

CONTACT INFORMATION
       For the latest version, see the mtr web page at
       http://www.bitwizard.nl/mtr/.

       The mtr mailinglist was little used and is no longer active.

       For patches, bug reports, or feature requests, please open an issue on
       GitHub at: https://github.com/traviscross/mtr.

SEE ALSO
       traceroute(8), ping(8), TCP/IP Illustrated (Stevens, ISBN 0201633469).

mtr                               July 12, 2014                          MTR(8)
```

iPerf

```
IPERF(1)                          User Manuals                          IPERF(1)

NAME
       iperf3 - perform network throughput tests

SYNOPSIS
       iperf3 -s [ options ]
```

```
       iperf3 -c server [ options ]

DESCRIPTION
       iperf3 is a tool for performing network throughput measurements.  It
       can test either TCP or UDP throughput.  To perform an iperf3 test the
       user must establish both a server and a client.

GENERAL OPTIONS
       -p, --port n
              set server port to listen on/connect to to n (default 5201)

       -f, --format
              [kmKM]   format to report: Kbits, Mbits, KBytes, MBytes

       -i, --interval n
              pause n seconds between periodic bandwidth reports; default is
              1, use 0 to disable

       -F, --file name
              client-side: read from the file and write to the network,
              instead of using random data; server-side: read from the network
              and write to the file, instead of throwing the data away

       -A, --affinity n/n,m
              Set the CPU affinity, if possible (linux only).  On both the
              client and server you can set the local affinity; in addition,
              on the client side you can override the server's affinity for
              just that one test, using the n,m form.

       -V, --verbose
              give more detailed output

       -J, --json
              output in JSON format

       -d, --debug
              emit debugging output.  Primarily (perhaps exclusively) of use
              to developers.

       -v, --version
              show version information and quit

       -h, --help
              show a help synopsis

SERVER SPECIFIC OPTIONS
       -s, --server
              run in server mode

       -D, --daemon
              run the server in background as a daemon

       -1, --one-off
              handle one client connection, then exit.

CLIENT SPECIFIC OPTIONS
       -c, --client host
              run in client mode, connecting to the specified server
```

```
-u, --udp
       use UDP rather than TCP

-b, --bandwidth n[KM]
       set target bandwidth to n bits/sec (default 1 Mbit/sec for UDP,
       unlimited for TCP).  If there are multiple streams (-P flag),
       the bandwidth limit is applied separately to each stream.  You
       can also add a '/' and a number to the bandwidth specifier.
       This is called "burst mode".  It will send the given number of
       packets without pausing, even if that temporarily exceeds the
       specified bandwidth limit.  Setting the target bandwidth to 0
       will disable bandwidth limits (particularly useful for UDP
       tests).

-t, --time n
       time in seconds to transmit for (default 10 secs)

-n, --bytes n[KM]
       number of bytes to transmit (instead of -t)

-k, --blockcount n[KM]
       number of blocks (packets) to transmit (instead of -t or -n)

-l, --length n[KM]
       length of buffer to read or write (default 128 KB for TCP, 8KB
       for UDP)

-P, --parallel n
       number of parallel client streams to run

-R, --reverse
       run in reverse mode (server sends, client receives)

-w, --window n[KM]
       window size / socket buffer size (this gets sent to the server
       and used on that side too)

-B, --bind n[KM]
       bind to a specific interface or multicast address

-M, --set-mss n
       set TCP maximum segment size (MTU - 40 bytes)

-N, --no-delay
       set TCP no delay, disabling Nagle's Algorithm

-4, --version4
       only use IPv4

-6, --version6
       only use IPv6

-S, --tos n
       set the IP 'type of service'

-L, --flowlabel n
       set the IPv6 flow label (currently only supported on Linux)

-Z, --zerocopy
```

```
              Use a "zero copy" method of sending data, such as sendfile(2),
              instead of the usual write(2).

       -O, --omit n
              Omit the first n seconds of the test, to skip past the TCP slow-
              start period.

       -T, --title str
              Prefix every output line with this string.

       -C, --linux-congestion algo
              Set the congestion control algorithm (linux only).

       --get-server-output
              Get the output from the server.  The output format is determined
              by the server (in particular, if the server was invoked with the
              --json flag, the output will be in JSON format, otherwise it
              will be in human-readable format).  If the client is run with
              --json, the server output is included in a JSON object;
              otherwise it is appended at the bottom of the human-readable
              output.

AUTHORS
       Iperf was originally written by Mark Gates and Alex Warshavsky.  Man
       page and maintence by Jon Dugan <jdugan at x1024 dot net>.  Other
       contributions from Ajay Tirumala, Jim Ferguson, Feng Qin, Kevin Gibbs,
       John Estabrook <jestabro at ncsa.uiuc.edu>, Andrew Gallatin <gallatin
       at gmail.com>, Stephen Hemminger <shemminger at linux-foundation.org>

SEE ALSO
       libiperf(3), http://software.es.net/iperf

ESnet                             January 2015                        IPERF(1)
```

NMAP

Please see: https://nmap.org/book/man.html for a complete walk through of nmap.

Rinetd

```
RINETD(8)                  System Manager's Manual                  RINETD(8)

NAME
     rinetd -- internet ``redirection server''

SYNOPSIS
     /usr/pkg/sbin/rinetd

VERSION
     Version 0.62, 04/14/2003.

DESCRIPTION
```

rinetd redirects TCP connections from one IP address and port to another. rinetd is a single-process server which handles any number of connections to the address/port pairs specified in the file /usr/pkg/etc/rinetd.conf. Since rinetd runs as a single process using nonblocking I/O, it is able to redirect a large number of connections without a severe impact on the machine. This makes it practical to run TCP services on machines inside an IP masquerading firewall. rinetd does not redirect FTP, because FTP requires more than one socket.

rinetd is typically launched at boot time, using the following syntax:

/usr/pkg/sbin/rinetd

The configuration file is found in the file /usr/pkg/etc/rinetd.conf, unless another file is specified using the -c command line option.

FORWARDING RULES

Most entries in the configuration file are forwarding rules. The format of a forwarding rule is as follows:

bindaddress bindport connectaddress connectport

For example:

206.125.69.81 80 10.1.1.2 80

Would redirect all connections to port 80 of the "real" IP address 206.125.69.81, which could be a virtual interface, through rinetd to port 80 of the address 10.1.1.2, which would typically be a machine on the inside of a firewall which has no direct routing to the outside world.

Although responding on individual interfaces rather than on all interfaces is one of rinetd's primary features, sometimes it is preferable to respond on all IP addresses that belong to the server. In this situation, the special IP address 0.0.0.0 can be used. For example:

0.0.0.0 23 10.1.1.2 23

Would redirect all connections to port 23, for all IP addresses assigned to the server. This is the default behavior for most other programs.

Service names can be specified instead of port numbers. On most systems, service names are defined in the file /etc/services.

Both IP addresses and hostnames are accepted for bindaddress and connectaddress.

ALLOW AND DENY RULES

Configuration files can also contain allow and deny rules.

Allow rules which appear before the first forwarding rule are applied globally: if at least one global allow rule exists, and the address of a new connection does not satisfy at least one of the global allow rules, that connection is immediately rejected, regardless of any other rules.

Allow rules which appear after a specific forwarding rule apply to that forwarding rule only. If at least one allow rule exists for a particular forwarding rule, and the address of a new connection does not satisfy at least one of the allow rules for that forwarding rule, that connection is

immediately rejected, regardless of any other rules.

Deny rules which appear before the first forwarding rule are applied globally: if the address of a new connection satisfies any of the global allow rules, that connection is immediately rejected, regardless of any other rules.

Deny rules which appear after a specific forwarding rule apply to that forwarding rule only. If the address of a new connection satisfies any of the deny rules for that forwarding rule, that connection is immediately rejected, regardless of any other rules.

The format of an allow rule is as follows:

allow pattern

Patterns can contain the following characters: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, . (period), ?, and *. The ? wildcard matches any one character. The * wildcard matches any number of characters, including zero.

For example:

allow 206.125.69.*

This allow rule matches all IP addresses in the 206.125.69 class C domain.

Host names are NOT permitted in allow and deny rules. The performance cost of looking up IP addresses to find their corresponding names is prohibitive. Since rinetd is a single process server, all other connections would be forced to pause during the address lookup.

LOGGING

rinetd is able to produce a log file in either of two formats: tab-delimited and web server-style "common log format."

By default, rinetd does not produce a log file. To activate logging, add the following line to the configuration file:

logfile log-file-location

Example: logfile /var/log/rinetd.log

By default, rinetd logs in a simple tab-delimited format containing the following information:

Date and time

Client address

Listening host

Listening port

Forwarded-to host

Forwarded-to port

Bytes received from client

Bytes sent to client

Result message

To activate web server-style "common log format" logging, add the following line to the configuration file:

logcommon

COMMAND LINE OPTIONS

The -c command line option is used to specify an alternate configuration file.

The -h command line option produces a short help message.

The -v command line option displays the version number.

REINITIALIZING RINETD

The kill -1 signal (SIGHUP) can be used to cause rinetd to reload its configuration file without interrupting existing connections. The process id is saved in the file /var/run/rinetd.pid to facilitate the kill -HUP. An alternate filename can be provided by using the <code>pidlogfile</code> configuration file option.

LIMITATIONS

rinetd redirects TCP connections only. There is no support for UDP. rinetd only redirects protocols which use a single TCP socket. This rules out FTP.

BUGS

The server redirected to is not able to identify the host the client really came from. This cannot be corrected; however, the log produced by rinetd provides a way to obtain this information. Under Unix, Sockets would theoretically lose data when closed with SO_LINGER turned off, but in Linux this is not the case (kernel source comments support this belief on my part). On non-Linux Unix platforms, alternate code which uses a different trick to work around blocking close() is provided, but this code is untested. The logging is inadequate. The duration of each connection should be logged.

LICENSE

CONTACT INFORMATION

See http://www.boutell.com/rinetd/ for the latest release. Thomas Boutell can be reached by email: boutell@boutell.com

THANKS

Thanks are due to Bill Davidsen, Libor Pechachek, Sascha Ziemann, the Apache Group, and many others who have contributed advice and/or source code to this and other free software projects.

Tinyproxy

```
TINYPROXY(8)                    Tinyproxy manual                    TINYPROXY(8)

NAME
       tinyproxy - A light-weight HTTP proxy daemon

SYNOPSIS
       tinyproxy [-vldch]

DESCRIPTION
       tinyproxy is a light-weight HTTP proxy daemon designed to consume a
       minimum amount of system resources. It listens on a given TCP port and
       handles HTTP proxy requests. Designed from the ground up to be fast and
       yet small, it is an ideal solution for use cases such as embedded
       deployments where a full featured HTTP proxy is required, but the
       system resources for a larger proxy are unavailable.

OPTIONS
       tinyproxy accepts the following options:

       -c <config-file>
           Use an alternate configuration file.

       -d
           Don't daemonize and stay in the foreground. Useful for debugging
           purposes.

       -h
           Display a short help screen of command line arguments and exit.

       -l
           Display the licensing agreement.

       -v
           Display version information and exit.

SIGNALS
       In addition to command-line options, there are also several signals
       that can be sent to tinyproxy while it is running to generate debugging
       information and to force certain events.

       SIGHUP
           Force Tinyproxy to do a garbage collection on the current
           connections linked list. This is usually done automatically after a
           certain number of connections have been handled.

TEMPLATE FILES
       There are two occasions when Tinyproxy delivers HTML pages to the
       client on it's own right:

            1.When an error occurred, a corresponding error page is
           returned.

            2.When a request for the stathost is made, a page
           summarizing the connection statistics is returned. (See STATHOST
           below.)
```

The layout of both error pages and the statistics page can be controlled via configurable HTML template files that are plain HTML files that additionally understand a few template variables.

TEMPLATE VARIABLES

There are several standard HTML variables that are available in every template file:

request
: The full HTTP request line.

cause
: The abbreviated cause of the error condition.

clientip
: The IP address of the client making the request.

clienthost
: The hostname of the client making the request.

version
: The version of Tinyproxy.

package
: The package name. Presently, resolves to tinyproxy.

date
: The current date/time in HTTP format.

In addition, almost all templates support:

detail
: A detailed, plain English explanation of the error and possible causes.

When Tinyproxy finds a variable name enclosed in braces, e.g. "{request}", then this is replaced by the value of the corresponding variable before delivery of the page.

STATHOST

Tinyproxy returns a HTML page with connection statistics when it receives a HTTP request for a certain host -- the stathost. The stathost name defaults to tinyproxy.stats and can be changed at runtime to any name or IP address with the configuration variable StatHost.

The stat file template can be changed at runtime through the configuration variable StatFile.

FILES

/usr/pkg/etc/tinyproxy/tinyproxy.conf, /var/run/tinyproxy/tinyproxy.pid, /var/log/tinyproxy/tinyproxy.log

BUGS

To report bugs in Tinyproxy, please visit <https://www.banu.com/tinyproxy/>.

SEE ALSO

tinyproxy.conf(5)

AUTHOR
       Written by the Tinyproxy project team.

COPYRIGHT

NAME
       tinyproxy.conf - Tinyproxy HTTP proxy daemon configuration file

SYNOPSIS
       tinyproxy.conf

DESCRIPTION
       tinyproxy(8) reads its configuration file, typically stored in
       /usr/pkg/etc/tinyproxy/tinyproxy.conf (or passed to Tinyproxy with -c
       on the command line). This manpage describes the syntax and contents of
       the configuration file.

       The Tinyproxy configuration file contains key-value pairs, one per
       line. Lines starting with # and empty lines are comments and are
       ignored. Keywords are case-insensitive, whereas values are
       case-sensitive. Values may be enclosed in double-quotes (") if they
       contain spaces.

       The possible keywords and their descriptions are as follows:

       User
           The user which the Tinyproxy process should run as, after the
           initial port-binding has been done as the root user. Either the
           user name or the UID may be specified.

       Group
           The group which the Tinyproxy process should run as, after the
           initial port-binding has been done as the root user. Either the
           group name or the GID may be specified.

       Port
           The port which the Tinyproxy service will listen on. If the port is
           less than 1024, you will need to start the Tinyproxy process as the
           root user.

       Listen
           By default, Tinyproxy listens for connections on all available

interfaces (i.e. it listens on the wildcard address 0.0.0.0). With this configuration parameter, Tinyproxy can be told to listen only on one specific address.

Bind
This allows you to specify which address Tinyproxy will bind to for outgoing connections to web servers or upstream proxies.

BindSame
If this boolean parameter is set to yes, then Tinyproxy will bind the outgoing connection to the IP address of the incoming connection that triggered the outgoing request.

Timeout
The maximum number of seconds of inactivity a connection is allowed to have before it is closed by Tinyproxy.

ErrorFile
This parameter controls which HTML file Tinyproxy returns when a given HTTP error occurs. It takes two arguments, the error number and the location of the HTML error file.

DefaultErrorFile
This parameter controls the HTML template file returned when an error occurs for which no specific error file has been set.

StatHost
This configures the host name or IP address that is treated as the stat host: Whenever a request for this host is received, Tinyproxy will return an internal statistics page instead of forwarding the request to that host. The template for this page can be configured with the StatFile configuration option. The default value of StatHost is tinyproxy.stats.

StatFile
This configures the HTML file that Tinyproxy sends when a request for the stathost is received. If this parameter is not set, Tinyproxy returns a hard-coded basic statistics page. See the STATHOST section in the tinyproxy(8) manual page for details.

Note that the StatFile and the error files configured with ErrorFile and DefaultErrorFile are template files that can contain a few template variables that Tinyproxy expands prior to delivery. Examples are "{cause}" for an abbreviated error description and "{detail}" for a detailed error message. The tinyproxy(8) manual page contains a description of all template variables.

LogFile
This controls the location of the file to which Tinyproxy writes its debug output. Alternatively, Tinyproxy can log to syslog -- see the Syslog option.

Syslog
When set to On, this option tells Tinyproxy to write its debug messages to syslog instead of to a log file configured with LogFile. These two options are mutually exclusive.

LogLevel
Sets the log level. Messages from the set level and above are

```
            logged. For example, if the LogLevel was set to Warning, then all
            log messages from Warning to Critical would be output, but Notice
            and below would be suppressed. Allowed values are:

                oCritical (least verbose)

                oError

                oWarning

                oNotice

                oConnect (log connections without Info's noise)

                oInfo (most verbose)

        PidFile
            This option controls the location of the file where the main
            Tinyproxy process stores its process ID for signaling purposes.

        XTinyproxy
            Setting this option to Yes tells Tinyproxy to add a header
            X-Tinyproxy containing the client's IP address to the request.

        Upstream, No Upstream
            This option allows you to set up a set of rules for deciding
            whether an upstream proxy server is to be used, based on the host
            or domain of the site being accessed. The rules are stored in the
            order encountered in the configuration file and the LAST matching
            rule wins. There are three possible forms for specifying upstream
            rules:

                oupstream host:port turns proxy upstream support on generally.

                oupstream host:port "site_spec" turns on the upstream proxy for
                the sites matching site_spec.

                ono upstream "site_spec" turns off upstream support for sites
                matching site_spec.

                    The site can be specified in various forms as a hostname,
domain
                    name or as an IP range:

                oname matches host exactly

                o.name matches any host in domain "name"

                o. matches any host with no domain (in empty domain)

                oIP/bits matches network/mask

                oIP/mask matches network/mask

        MaxClients
            Tinyproxy creates one child process for each connected client. This
            options specifies the absolute highest number processes that will
            be created. With other words, only MaxClients clients can be
            connected to Tinyproxy simultaneously.
```

MinSpareServers, MaxSpareServers
    Tinyproxy always keeps a certain number of idle child processes so that it can handle new incoming client requests quickly. MinSpareServer and MaxSpareServers control the lower and upper limits for the number of spare processes. I.e. when the number of spare servers drops below MinSpareServers then Tinyproxy will start forking new spare processes in the background and when the number of spare processes exceeds MaxSpareServers then Tinyproxy will kill off extra processes.

StartServers
    The number of servers to start initially. This should usually be set to a value between MinSpareServers and MaxSpareServers.

MaxRequestsPerChild
    This limits the number of connections that a child process will handle before it is killed. The default value is 0 which disables this feature. This option is meant as an emergency measure in the case of problems with memory leakage. In that case, setting MaxRequestsPerChild to a value of e.g. 1000, or 10000 can be useful.

Allow, Deny
    The Allow and Deny options provide a means to customize which clients are allowed to access Tinyproxy. Allow and Deny lines can be specified multiple times to build the access control list for Tinyproxy. The order in the config file is important. If there are no Allow or Deny lines, then all clients are allowed. Otherwise, the default action is to deny access. The argument to Allow or Deny can be a single IP address of a client host, like 127.0.0.1, an IP address range, like 192.168.0.1/24 or a string that will be matched against the end of the client host name, i.e, this can be a full host name like host.example.com or a domain name like .example.com or even a top level domain name like .com.

AddHeader
    Configure one or more HTTP request headers to be added to outgoing HTTP requests that Tinyproxy makes. Note that this option will not work for HTTPS traffic, as Tinyproxy has no control over what headers are exchanged.

    AddHeader "X-My-Header" "Powered by Tinyproxy"

ViaProxyName
    RFC 2616 requires proxies to add a Via header to the HTTP requests, but using the real host name can be a security concern. If the ViaProxyname option is present, then its string value will be used as the host name in the Via header. Otherwise, the server's host name will be used.

DisableViaHeader
    When this is set to yes, Tinyproxy does NOT add the Via header to the requests. This virtually puts Tinyproxy into stealth mode. Note that RFC 2616 requires proxies to set the Via header, so by enabling this option, you break compliance. Don't disable the Via header unless you know what you are doing...

Filter

Tinyproxy supports filtering of web sites based on URLs or domains. This option specifies the location of the file containing the filter rules, one rule per line.

FilterURLs
If this boolean option is set to Yes or On, filtering is performed for URLs rather than for domains. The default is to filter based on domains.

FilterExtended
If this boolean option is set to Yes, then extended POSIX regular expressions are used for matching the filter rules. The default is to use basic POSIX regular expressions.

FilterCaseSensitive
If this boolean option is set to Yes, then the filter rules are matched in a case sensitive manner. The default is to match case-insensitively.

FilterDefaultDeny
The default filtering policy is to allow everything that is not matched by a filtering rule. Setting FilterDefaultDeny to Yes changes the policy do deny everything but the domains or URLs matched by the filtering rules.

Anonymous
If an Anonymous keyword is present, then anonymous proxying is enabled. The headers listed with Anonymous are allowed through, while all others are denied. If no Anonymous keyword is present, then all headers are allowed through. You must include quotes around the headers.

Most sites require cookies to be enabled for them to work correctly, so you will need to allow cookies through if you access those sites.

Example:

```
Anonymous "Host"
Anonymous "Authorization"
Anonymous "Cookie"
```

ConnectPort
This option can be used to specify the ports allowed for the CONNECT method. If no ConnectPort line is found, then all ports are allowed. To disable CONNECT altogether, include a single ConnectPort line with a value of 0.

ReversePath
Configure one or more ReversePath directives to enable reverse proxy support. With reverse proxying it's possible to make a number of sites appear as if they were part of a single site.

If you uncomment the following two directives and run Tinyproxy on your own computer at port 8888, you can access example.com, using http://localhost:8888/example/.

```
ReversePath "/example/" "http://www.example.com/"
```

```
       ReverseOnly
           When using Tinyproxy as a reverse proxy, it is STRONGLY recommended
           that the normal proxy is turned off by setting this boolean option
           to Yes.

       ReverseMagic
           Setting this option to Yes, makes Tinyproxy use a cookie to track
           reverse proxy mappings. If you need to reverse proxy sites which
           have absolute links you must use this option.

       ReverseBaseURL
           The URL that is used to access this reverse proxy. The URL is used
           to rewrite HTTP redirects so that they won't escape the proxy. If
           you have a chain of reverse proxies, you'll need to put the
           outermost URL here (the address which the end user types into
           his/her browser). If this option is not set then no rewriting of
           redirects occurs.

BUGS
       To report bugs in Tinyproxy, please visit
       <https://www.banu.com/tinyproxy/>.

SEE ALSO
       tinyproxy(8)

AUTHOR
       Written by the Tinyproxy project team.

COPYRIGHT
       Copyright (c) 1998-2000 Steven Young; Copyright (c) 2000-2001 Robert
       James Kaes; Copyright (c) 2009-2010 Mukund Sivaraman; Copyright (c)
       2009-2010 Michael Adam.

       This program is distributed under the terms of the GNU General Public
       License version 2 or above. See the COPYING file for additional
       information.

Version 1.8.3                      03/25/2017                 TINYPROXY.CONF(5)
```

Telnet

```
TELNET(1)                  General Commands Manual                  TELNET(1)

NAME
     telnet -- user interface to the TELNET protocol

SYNOPSIS
     telnet [-4] [-6] [-8] [-E] [-F] [-K] [-L] [-N] [-S tos] [-X authtype]
            [-a] [-c] [-d] [-e escapechar] [-f] [-k realm] [-l user]
            [-n tracefile] [-P policy] [-r] [-x] [host [port]]

DESCRIPTION
     The telnet command is used to communicate with another host using the
     TELNET protocol.  If telnet is invoked without the host argument, it
     enters command mode, indicated by its prompt (telnet>).  In this mode, it
```

accepts and executes the commands listed below. If it is invoked with arguments, it performs an open command with those arguments.

Options:

-4 Forces telnet to use IPv4 addresses only.

-6 Forces telnet to use IPv6 addresses only.

-8 Specifies an 8-bit data path. This causes an attempt to negotiate the TELNET BINARY option on both input and output.

-E Stops any character from being recognized as an escape character.

-F If Kerberos V5 authentication is being used, the -F option allows the local credentials to be forwarded to the remote system, including any credentials that have already been forwarded into the local environment.

-K Specifies no automatic login to the remote system.

-L Specifies an 8-bit data path on output. This causes the BINARY option to be negotiated on output.

-N Numeric host address. No attempt will be made to look up symbolic names for host addresses.

-S tos Sets the IP type-of-service (TOS) option for the telnet connection to the value tos, which can be a numeric TOS value or, on systems that support it, a symbolic TOS name found in the /etc/iptos file.

-X atype
Disables the atype type of authentication.

-a Attempt automatic login. Currently, this sends the user name via the USER variable of the ENVIRON option if supported by the remote system. The name used is that of the current user as returned by getlogin(2) if it agrees with the current user ID, otherwise it is the name associated with the user ID.

-c Disables the reading of the user's .telnetrc file. (See the toggle skiprc command on this man page.)

-d Sets the initial value of the debug toggle to TRUE.

-e escape char
Sets the initial telnet escape character to escape char. If escape char is omitted, then there will be no escape character.

-f If Kerberos V5 authentication is being used, the -f option allows the local credentials to be forwarded to the remote system.

-k realm
If Kerberos authentication is being used, the -k option requests that telnet obtain tickets for the remote host in realm realm instead of the remote host's realm, as determined by krb_realmofhost(3).

-l user
When connecting to the remote system, if the remote system understands the ENVIRON option, then user will be sent to the remote system as the value for the variable USER. This option implies the -a option. This option may also be used with the open command.

-n tracefile
Opens tracefile for recording trace information. See the set tracefile command below.

-P policy
Use IPsec policy specification string policy, for the connections. See ipsec_set_policy(3) for details.

-r Specifies a user interface similar to rlogin(1). In this mode, the escape character is set to the tilde (~) character, unless modified by the -e option.

-x Turns on encryption of the data stream if possible. This option is not available outside of the United States and Canada.

host Indicates the official name, an alias, or the Internet address of a remote host.

port Indicates a port number (address of an application). If a number is not specified, the default telnet port is used.

When in rlogin mode, a line of the form ~. disconnects from the remote host; ~ is the telnet escape character. Similarly, the line ~^Z suspends the telnet session. The line ~^] escapes to the normal telnet escape prompt.

Once a connection has been opened, telnet will attempt to enable the TELNET LINEMODE option. If this fails, then telnet will revert to one of two input modes: either ``character at a time'' or ``old line by line'' depending on what the remote system supports.

When LINEMODE is enabled, character processing is done on the local system, under the control of the remote system. When input editing or character echoing is to be disabled, the remote system will relay that information. The remote system will also relay changes to any special characters that happen on the remote system, so that they can take effect on the local system.

In ``character at a time'' mode, most text typed is immediately sent to the remote host for processing.

In ``old line by line'' mode, all text is echoed locally, and (normally) only completed lines are sent to the remote host. The ``local echo character'' (initially ``^E'') may be used to turn off and on the local echo (this would mostly be used to enter passwords without the password being echoed).

If the LINEMODE option is enabled, or if the localchars toggle is TRUE (the default for ``old line by line``; see below), the user's quit, intr, and flush characters are trapped locally, and sent as TELNET protocol sequences to the remote side. If LINEMODE has ever been enabled, then the user's susp and eof are also sent as TELNET protocol sequences, and

quit is sent as a TELNET ABORT instead of BREAK. There are options (see toggle autoflush and toggle autosynch below) which cause this action to flush subsequent output to the terminal (until the remote host acknowledges the TELNET sequence) and flush previous terminal input (in the case of quit and intr).

While connected to a remote host, telnet command mode may be entered by typing the telnet ``escape character'' (initially ``^]''). When in command mode, the normal terminal editing conventions are available.

The following telnet commands are available. Only enough of each command to uniquely identify it need be typed (this is also true for arguments to the mode, set, toggle, unset, slc, environ, and display commands).

auth argument ...
The auth command manipulates the information sent through the TELNET AUTHENTICATE option. Valid arguments for the auth command are as follows:

disable type Disables the specified type of authentication. To obtain a list of available types, use the auth disable ? command.

enable type Enables the specified type of authentication. To obtain a list of available types, use the auth enable ? command.

status Lists the current status of the various types of authentication.

close Close a TELNET session and return to command mode.

display argument ...
Displays all, or some, of the set and toggle values (see below).

encrypt argument ...
The encrypt command manipulates the information sent through the TELNET ENCRYPT option.

Note: Because of export controls, the TELNET ENCRYPT option is not supported outside of the United States and Canada.

Valid arguments for the encrypt command are:

disable type [input|output]
Disables the specified type of encryption. If you omit the input and output, both input and output are disabled. To obtain a list of available types, use the encrypt disable ? command.

enable type [input|output]
Enables the specified type of encryption. If you omit input and output, both input and output are enabled. To obtain a list of available types, use the encrypt enable ? command.

input This is the same as the encrypt start input

command.

-input This is the same as the encrypt stop input command.

output This is the same as the encrypt start output command.

-output This is the same as the encrypt stop output command.

start [input|output]
Attempts to start encryption. If you omit input and output, both input and output are enabled. To obtain a list of available types, use the encrypt enable ? command.

status Lists the current status of encryption.

stop [input|output]
Stops encryption. If you omit input and output, encryption is on both input and output.

type type Sets the default type of encryption to be used with later encrypt start or encrypt stop commands.

environ arguments ...
The environ command is used to manipulate the variables that may be sent through the TELNET ENVIRON option. The initial set of variables is taken from the users environment, with only the DISPLAY and PRINTER variables being exported by default. The USER variable is also exported if the -a or -l options are used.

Valid arguments for the environ command are:

define variable value
Define the variable variable to have a value of value. Any variables defined by this command are automatically exported. The value may be enclosed in single or double quotes so that tabs and spaces may be included.

undefine variable
Remove variable from the list of environment variables.

export variable
Mark the variable variable to be exported to the remote side.

unexport variable
Mark the variable variable to not be exported unless explicitly asked for by the remote side.

list List the current set of environment variables. Those marked with a * will be sent automatically, other variables will only be sent if explicitly

requested.

? Prints out help information for the environ command.

logout Sends the TELNET LOGOUT option to the remote side. This command is similar to a close command; however, if the remote side does not support the LOGOUT option, nothing happens. If, however, the remote side does support the LOGOUT option, this command should cause the remote side to close the TELNET connection. If the remote side also supports the concept of suspending a user's session for later reattachment, the logout argument indicates that you should terminate the session immediately.

mode type Type is one of several options, depending on the state of the TELNET session. The remote host is asked for permission to go into the requested mode. If the remote host is capable of entering that mode, the requested mode will be entered.

character Disable the TELNET LINEMODE option, or, if the remote side does not understand the LINEMODE option, then enter ``character at a time`` mode.

line Enable the TELNET LINEMODE option, or, if the remote side does not understand the LINEMODE option, then attempt to enter ``old-line-by-line`` mode.

isig (-isig) Attempt to enable (disable) the TRAPSIG mode of the LINEMODE option. This requires that the LINEMODE option be enabled.

edit (-edit) Attempt to enable (disable) the EDIT mode of the LINEMODE option. This requires that the LINEMODE option be enabled.

softtabs (-softtabs)
Attempt to enable (disable) the SOFT_TAB mode of the LINEMODE option. This requires that the LINEMODE option be enabled.

litecho (-litecho)
Attempt to enable (disable) the LIT_ECHO mode of the LINEMODE option. This requires that the LINEMODE option be enabled.

? Prints out help information for the mode command.

open host [-l user] [-a] [[-]port]
Open a connection to the named host. If no port number is specified, telnet will attempt to contact a TELNET server at the default port. The host specification may be either a host name (see hosts(5)) or an Internet address specified in the ``dot notation'' (see inet(3)). The -l option may be used to specify the user name to be passed to the remote system via the ENVIRON option. If a port is specified telnet omits any automatic initialisation of TELNET options. When the port

number is preceded by a minus sign, the initial option negotiation is done.

After establishing a connection, the file .telnetrc in the user's home directory is read. Lines beginning with a # are comment lines. Blank lines are ignored. Lines that begin without white space are the start of a machine entry. The first thing on such a line is a string identifying the machine that is being connected to. It may be the hostname or numeric address specified as the argument host, the canonical name of that string as determined by getaddrinfo(3), or the string ``DEFAULT'' indicating all hosts. The rest of the line, and successive lines that begin with white space are assumed to be telnet commands and are processed as if they had been typed in manually to the telnet command prompt.

quit Close any open TELNET session and exit telnet. An end of file (in command mode) will also close a session and exit.

send arguments

Sends one or more special character sequences to the remote host. The following are the arguments which may be specified (more than one argument may be specified at a time):

abort Sends the TELNET ABORT (Abort processes) sequence.

ao Sends the TELNET AO (Abort Output) sequence, which should cause the remote system to flush all output from the remote system to the user's terminal.

ayt Sends the TELNET AYT (Are You There) sequence, to which the remote system may or may not choose to respond.

brk Sends the TELNET BRK (Break) sequence, which may have significance to the remote system.

ec Sends the TELNET EC (Erase Character) sequence, which should cause the remote system to erase the last character entered.

el Sends the TELNET EL (Erase Line) sequence, which should cause the remote system to erase the line currently being entered.

eof Sends the TELNET EOF (End Of File) sequence.

eor Sends the TELNET EOR (End of Record) sequence.

escape Sends the current telnet escape character (initially ``^'').

ga Sends the TELNET GA (Go Ahead) sequence, which likely has no significance to the remote system.

getstatus

If the remote side supports the TELNET STATUS command, getstatus will send the subnegotiation to request that the server send its current option status.

ip Sends the TELNET IP (Interrupt Process) sequence, which should cause the remote system to abort the currently running process.

nop Sends the TELNET NOP (No OPeration) sequence.

susp Sends the TELNET SUSP (SUSPend process) sequence.

synch Sends the TELNET SYNCH sequence. This sequence causes the remote system to discard all previously typed (but not yet read) input. This sequence is sent as TCP urgent data (and may not work if the remote system is a 4.2BSD system -- if it doesn't work, a lower case ``r'' may be echoed on the terminal).

do cmd

dont cmd

will cmd

wont cmd
Sends the TELNET DO cmd sequence. Cmd can be either a decimal number between 0 and 255, or a symbolic name for a specific TELNET command. Cmd can also be either help or ? to print out help information, including a list of known symbolic names.

? Prints out help information for the send command.

set argument value

unset argument value
The set command will set any one of a number of telnet variables to a specific value or to TRUE. The special value off turns off the function associated with the variable, this is equivalent to using the unset command. The unset command will disable or set to FALSE any of the specified functions. The values of variables may be interrogated with the display command. The variables which may be set or unset, but not toggled, are listed here. In addition, any of the variables for the toggle command may be explicitly set or unset using the set and unset commands.

ayt If TELNET is in localchars mode, or LINEMODE is enabled, and the status character is typed, a TELNET AYT sequence (see send ayt above) is sent to the remote host. The initial value for the "Are You There" character is the terminal's status character.

echo This is the value (initially ``^E'') which, when in ``line by line'' mode, toggles between doing local echoing of entered characters (for normal processing), and suppressing echoing of entered characters (for entering, say, a password).

eof If telnet is operating in LINEMODE or ``old line by line'' mode, entering this character as the first

character on a line will cause this character to be sent to the remote system. The initial value of the eof character is taken to be the terminal's eof character.

erase If telnet is in localchars mode (see toggle localchars below), and if telnet is operating in ``character at a time'' mode, then when this character is typed, a TELNET EC sequence (see send ec above) is sent to the remote system. The initial value for the erase character is taken to be the terminal's erase character.

escape This is the telnet escape character (initially ``^['') which causes entry into telnet command mode (when connected to a remote system).

flushoutput
If telnet is in localchars mode (see toggle localchars below) and the flushoutput character is typed, a TELNET AO sequence (see send ao above) is sent to the remote host. The initial value for the flush character is taken to be the terminal's flush character.

forw1

forw2 If TELNET is operating in LINEMODE, these are the characters that, when typed, cause partial lines to be forwarded to the remote system. The initial value for the forwarding characters are taken from the terminal's eol and eol2 characters.

interrupt
If telnet is in localchars mode (see toggle localchars below) and the interrupt character is typed, a TELNET IP sequence (see send ip above) is sent to the remote host. The initial value for the interrupt character is taken to be the terminal's intr character.

kill If telnet is in localchars mode (see toggle localchars below), and if telnet is operating in ``character at a time'' mode, then when this character is typed, a TELNET EL sequence (see send el above) is sent to the remote system. The initial value for the kill character is taken to be the terminal's kill character.

lnext If telnet is operating in LINEMODE or ``old line by line`` mode, then this character is taken to be the terminal's lnext character. The initial value for the lnext character is taken to be the terminal's lnext character.

quit If telnet is in localchars mode (see toggle localchars below) and the quit character is typed, a TELNET BRK sequence (see send brk above) is sent to the remote host. The initial value for the quit character is taken to be the terminal's quit character.

reprint
If telnet is operating in LINEMODE or ``old line by line`` mode, then this character is taken to be the terminal's reprint character. The initial value for the reprint character is taken to be the terminal's reprint character.

rlogin This is the rlogin escape character. If set, the normal TELNET escape character is ignored unless it is preceded by this character at the beginning of a line. This character, at the beginning of a line followed by a "." closes the connection; when followed by a ^Z it suspends the telnet command. The initial state is to disable the rlogin escape character.

start If the TELNET TOGGLE-FLOW-CONTROL option has been enabled, then this character is taken to be the terminal's start character. The initial value for the start character is taken to be the terminal's start character.

stop If the TELNET TOGGLE-FLOW-CONTROL option has been enabled, then this character is taken to be the terminal's stop character. The initial value for the stop character is taken to be the terminal's stop character.

susp If telnet is in localchars mode, or LINEMODE is enabled, and the suspend character is typed, a TELNET SUSP sequence (see send susp above) is sent to the remote host. The initial value for the suspend character is taken to be the terminal's suspend character.

tracefile
This is the file to which the output, caused by netdata or option tracing being TRUE, will be written. If it is set to ``-'', then tracing information will be written to standard output (the default).

worderase
If telnet is operating in LINEMODE or ``old line by line`` mode, then this character is taken to be the terminal's worderase character. The initial value for the worderase character is taken to be the terminal's worderase character.

? Displays the legal set (unset) commands.

slc state The slc command (Set Local Characters) is used to set or change the state of the special characters when the TELNET LINEMODE option has been enabled. Special characters are characters that get mapped to TELNET commands sequences (like ip or quit) or line editing characters (like erase and kill). By default, the local special characters are exported.

check Verify the current settings for the current special characters. The remote side is requested

to send all the current special character settings, and if there are any discrepancies with the local side, the local side will switch to the remote value.

export — Switch to the local defaults for the special characters. The local default characters are those of the local terminal at the time when telnet was started.

import — Switch to the remote defaults for the special characters. The remote default characters are those of the remote system at the time when the TELNET connection was established.

? — Prints out help information for the slc command.

status — Show the current status of telnet. This includes the peer one is connected to, as well as the current mode.

toggle arguments ...

Toggle (between TRUE and FALSE) various flags that control how telnet responds to events. These flags may be set explicitly to TRUE or FALSE using the set and unset commands listed above. More than one argument may be specified. The state of these flags may be interrogated with the display command. Valid arguments are:

authdebug — Turns on debugging information for the authentication code.

autoflush — If autoflush and localchars are both TRUE, then when the ao, or quit characters are recognized (and transformed into TELNET sequences; see set above for details), telnet refuses to display any data on the user's terminal until the remote system acknowledges (via a TELNET TIMING MARK option) that it has processed those TELNET sequences. The initial value for this toggle is TRUE if the terminal user had not done an "stty noflsh", otherwise FALSE (see stty(1)).

autodecrypt — When the TELNET ENCRYPT option is negotiated, by default the actual encryption (decryption) of the data stream does not start automatically. The autoencrypt (autodecrypt) command states that encryption of the output (input) stream should be enabled as soon as possible.

Note: Because of export controls, the TELNET ENCRYPT option is not supported outside the United States and Canada.

autologin — If the remote side supports the TELNET AUTHENTICATION option TELNET attempts to use it to perform automatic authentication. If the AUTHENTICATION option is not supported, the user's login name are propagated through the TELNET ENVIRON option. This command is the same

as specifying the -a option on the open command.

autosynch — If autosynch and localchars are both TRUE, then when either the intr or quit characters is typed (see set above for descriptions of the intr and quit characters), the resulting TELNET sequence sent is followed by the TELNET SYNCH sequence. This procedure should cause the remote system to begin throwing away all previously typed input until both of the TELNET sequences have been read and acted upon. The initial value of this toggle is FALSE.

binary — Enable or disable the TELNET BINARY option on both input and output.

inbinary — Enable or disable the TELNET BINARY option on input.

outbinary — Enable or disable the TELNET BINARY option on output.

crlf — If this is TRUE, then carriage returns will be sent as <CR><LF>. If this is FALSE, then carriage returns will be send as <CR><NUL>. The initial value for this toggle is FALSE.

crmod — Toggle carriage return mode. When this mode is enabled, most carriage return characters received from the remote host will be mapped into a carriage return followed by a line feed. This mode does not affect those characters typed by the user, only those received from the remote host. This mode is not very useful unless the remote host only sends carriage return, but never line feed. The initial value for this toggle is FALSE.

debug — Toggles socket level debugging (useful only to the super user). The initial value for this toggle is FALSE.

encdebug — Turns on debugging information for the encryption code.

localchars — If this is TRUE, then the flush, interrupt, quit, erase, and kill characters (see set above) are recognized locally, and transformed into (hopefully) appropriate TELNET control sequences (respectively ao, ip, brk, ec, and el; see send above). The initial value for this toggle is TRUE in ``old line by line'' mode, and FALSE in ``character at a time'' mode. When the LINEMODE option is enabled, the value of localchars is ignored, and assumed to always be TRUE. If LINEMODE has ever been enabled, then quit is sent as abort, and eof and suspend are sent as eof and susp (see send above).

netdata Toggles the display of all network data (in hexadecimal format). The initial value for this toggle is FALSE.

options Toggles the display of some internal telnet protocol processing (having to do with TELNET options). The initial value for this toggle is FALSE.

prettydump When the netdata toggle is enabled, if prettydump is enabled the output from the netdata command will be formatted in a more user readable format. Spaces are put between each character in the output, and the beginning of any TELNET escape sequence is preceded by a '*' to aid in locating them.

skiprc When the skiprc toggle is TRUE, TELNET skips the reading of the .telnetrc file in the users home directory when connections are opened. The initial value for this toggle is FALSE.

termdata Toggles the display of all terminal data (in hexadecimal format). The initial value for this toggle is FALSE.

verbose_encrypt
When the verbose_encrypt toggle is TRUE, telnet prints out a message each time encryption is enabled or disabled. The initial value for this toggle is FALSE. Note: Because of export controls, data encryption is not supported outside of the United States and Canada.

? Displays the legal toggle commands.

z Suspend telnet. This command only works when the user is using the csh(1).

! [command]
Execute a single command in a subshell on the local system. If command is omitted, then an interactive subshell is invoked.

? [command]
Get help. With no arguments, telnet prints a help summary. If a command is specified, telnet will print the help information for just that command.

ENVIRONMENT

telnet uses at least the HOME, SHELL, DISPLAY, and TERM environment variables. Other environment variables may be propagated to the other side via the TELNET ENVIRON option.

FILES

~/.telnetrc user customized telnet startup values

HISTORY

The telnet command appeared in 4.2BSD. IPsec support was added by

```
     WIDE/KAME project, in 1999.

NOTES
     On some remote systems, echo has to be turned off manually when in ``old
     line by line'' mode.

     In ``old line by line'' mode or LINEMODE the terminal's eof character is
     only recognized (and sent to the remote system) when it is the first
     character on a line.

NetBSD 6.1.5                  October 28, 2003                  NetBSD 6.1.5
```

nslookup

```
NSLOOKUP(8)             NetBSD System Manager's Manual             NSLOOKUP(8)

NAME
     nslookup - query Internet name servers interactively

SYNOPSIS
     nslookup [-option ...] [host-to-find | -[server]]

DESCRIPTION
     Nslookup is a program to query Internet domain name servers.  Nslookup
     has two modes: interactive and non-interactive.  Interactive mode allows
     the user to query name servers for information about various hosts and
     domains or to print a list of hosts in a domain.  Non-interactive mode is
     used to print just the name and requested information for a host or do-
     main.

ARGUMENTS
     Interactive mode is entered in the following cases:

     a)    when no arguments are given (the default name server will be used),

     b)    when the first argument is a hyphen (-) and the second argument is
           the host name or Internet address of a name server.

     Non-interactive mode is used when the name or Internet address of the
     host to be looked up is given as the first argument. The optional second
     argument specifies the host name or address of a name server.

     The options listed under the ``set'' command below can be specified in
     the .nslookuprc file in the user's home directory if they are listed one
     per line.  Options can also be specified on the command line if they pre-
     cede the arguments and are prefixed with a hyphen.  For example, to
     change the default query type to host information, and the initial time-
     out to 10 seconds, type:

                    nslookup -query=hinfo  -timeout=10

INTERACTIVE COMMANDS
     Commands may be interrupted at any time by typing a control-C.  To exit,
     type a control-D (EOF) or type exit.  The command line length must be
     less than 256 characters.  To treat a built-in command as a host name,
```

precede it with an escape character (`\'). **N.B.:** unrecognized command will be interpreted as a N.B.:0 0N.B.:1

host [server]
Look up information for host using the current default server or using server, if specified. If host is an Internet address and the query type is A or PTR, the name of the host is returned. If host is a name and does not have a trailing period, the default domain name is appended to the name. (This behavior depends on the state of the **set** options **domain**, **srchlist**, **defname**, and **search**.)

To look up a host not in the current domain, append a period to the name.

server domain

lserver domain
Change the default server to domain; **lserver** uses the initial server to look up information about domain, while **server** uses the current default server. If an authoritative answer can't be found, the names of servers that might have the answer are returned.

root Changes the default server to the server for the root of the domain name space. Currently, the host ns.internic.net is used. (This command is a synonym for ``**lserver ns.internic.net**''.) The name of the root server can be changed with the ``**set root**'' command.

finger [name] [**>** filename]

finger [name] [**>>** filename]
Connects with the finger server on the current host. The current host is defined when a previous lookup for a host was successful and returned address information (see the ``**set querytype**=A'' command). The name is optional. **>** and **>>** can be used to redirect output in the usual manner.

ls [option] domain [**>** filename]

ls [option] domain [**>>** filename]
List the information available for domain, optionally creating or appending to filename. The default output contains host names and their Internet addresses. Option can be one of the following:

-t querytype
lists all records of the specified type (see querytype below).

-a lists aliases of hosts in the domain; synonym for ``**-t** CNAME''.

-d lists all records for the domain; synonym for ``**-t** ANY''.

-h lists CPU and operating system information for the domain; synonym for ``**-t** HINFO''.

-s lists well-known services of hosts in the domain; synonym for ``**-t** WKS''.

When output is directed to a file, hash marks are printed for every 50 records received from the server.

view filename
Sorts and lists the output of previous **ls** command(s) with more(1).

help

? Prints a brief summary of commands.

exit Exits the program.

set keyword[=value]
This command is used to change state information that affects the lookups. Valid keywords are:

all Prints the current values of the frequently-used options to **set**. Information about the current default server and host is also printed.

class=value
Change the query class to one of:

IN the Internet class

CHAOS the Chaos class

HESIOD the MIT Athena Hesiod class

ANY wildcard (any of the above)

The class specifies the protocol group of the information.

(Default = IN; abbreviation = **cl**)

[**no**]**debug**
Turn debugging mode on. A lot more information is printed about the packet sent to the server and the resulting answer.

(Default = **nodebug**; abbreviation = [**no**]**deb**)

[**no**]**d2** Turn exhaustive debugging mode on. Essentially all fields of every packet are printed.

(Default = **nod2**)

domain=name
Change the default domain name to name. The default domain name is appended to a lookup request depending on the state of the **defname** and **search** options. The domain search list contains the parents of the default domain if it has at least two components in its name. For example, if the default domain is CC.Berkeley.EDU,

the search list is CC.Berkeley.EDU and Berkeley.EDU. Use the ``**set srchlist**'' command to specify a different list. Use the ``**set all**'' command to display the list.

(Default = value from hostname(1), /etc/resolv.conf, or LOCALDOMAIN; abbreviation = **do**)

srchlist=name1/name2/...
Change the default domain name to name1 and the domain search list to name1, name2, etc. A maximum of 6 names separated by slashes (/) can be specified. For example,

```
set srchlist=lcs.MIT.EDU/ai.MIT.EDU/MIT.EDU
```

sets the domain to lcs.MIT.EDU and the search list to the three names. This command overrides the default domain name and search list of the ``**set domain**'' command. Use the ``**set all**'' command to display the list.

(Default = value based on hostname(1), /etc/resolv.conf, or LOCALDOMAIN; abbreviation = **srchl**)

[**no**]**defname**
If set, append the default domain name to a single-component lookup request (i.e., one that does not contain a period).

(Default = **defname**; abbreviation = [**no**]**defname**)

[**no**]**search**
If the lookup request contains at least one period but doesn't end with a trailing period, append the domain names in the domain search list to the request until an answer is received.

(Default = **search**; abbreviation = [**no**]**sea**)

port=value
Change the default TCP/UDP name server port to value.

(Default = 53; abbreviation = **po**)

querytype=value

type=value
Change the type of information query to one of:

A	the host's Internet address.
CNAME	the canonical name for an alias.
HINFO	the host CPU and operating system type.
MINFO	the mailbox or mail list information.
MX	the mail exchanger.
NS	the name server for the named zone.

PTR the host name if the query is an Internet address; otherwise, the pointer to other information.

SOA the domain's ``start-of-authority'' information.

TXT the text information.

UINFO the user information.

WKS the supported well-known services.

Other types (ANY, AXFR, MB, MD, MF, NULL) are described in the RFC-1035 document.

(Default = A; abbreviations = **q**, **ty**)

[**no**]**recurse**
Tell the name server to query other servers if it does not have the information.

(Default = **recurse**; abbreviation = [**no**]**rec**)

retry=number
Set the number of retries to number. When a reply to a request is not received within a certain amount of time (changed with ``**set timeout**''), the timeout period is doubled and the request is resent. The retry value controls how many times a request is resent before giving up.

(Default = 4, abbreviation = **ret**)

root=host
Change the name of the root server to host. This affects the ``**root**'' command.

(Default = **ns.internic.net.**; abbreviation = **ro**)

timeout=number
Change the initial timeout interval for waiting for a reply to number seconds. Each retry doubles the timeout period.

(Default = 5 seconds; abbreviation = **ti**)

[**no**]**vc** Always use a virtual circuit when sending requests to the server.

(Default = **novc**; abbreviation = [**no**]**v**)

[**no**]**ignoretc**
Ignore packet truncation errors.

(Default = **noignoretc**; abbreviation = [**no**]**ig**)

DIAGNOSTICS

If the lookup request was not successful, an error message is printed. Possible errors are:

Timed out
: The server did not respond to a request after a certain amount of time (changed with ``**set timeout**=value'') and a certain number of retries (changed with ``**set retry**=value'').

No response from server
: No name server is running on the server machine.

No records
: The server does not have resource records of the current query type for the host, although the host name is valid. The query type is specified with the ``**set querytype**'' command.

Non-existent domain
: The host or domain name does not exist.

Connection refused

Network is unreachable
: The connection to the name or finger server could not be made at the current time. This error commonly occurs with **ls** and **finger** requests.

Server failure
: The name server found an internal inconsistency in its database and could not return a valid answer.

Refused
: The name server refused to service the request.

Format error
: The name server found that the request packet was not in the proper format. It may indicate an error in **nslookup**.

FILES

/etc/resolv.conf	initial domain name and name server addresses
$HOME/.nslookuprc	user's initial options
/usr/share/misc/nslookup.help	summary of commands

ENVIRONMENT

HOSTALIASES	file containing host aliases
LOCALDOMAIN	overrides default domain

SEE ALSO

named(8), resolver(3), resolv.conf(5); RFC-1034, ``Domain Names - Concepts and Facilities''; RFC-1035, ``Domain Names - Implementation and Specification''.

AUTHOR

Andrew Cherenson

dbclient

```
dbclient(1)                 General Commands Manual                dbclient(1)

NAME
       dbclient - lightweight SSH client

SYNOPSIS
       dbclient [-Tt] [-p port] [-i id] [-L l:h:r] [-R l:h:r] [-l user] host
       [command]

       dbclient [ args ] [user1]@host1[%port1],[user2]@host2[%port2],...

DESCRIPTION
       dbclient is a SSH client designed to be small enough to be used in
       small memory environments, while still being functional and secure
       enough for general use.

OPTIONS
       -p port
              Connect to port on the remote host. Alternatively a port can be
              specified as hostname%port.  Default is 22.

       -i idfile
              Identity file.  Read the identity key from file idfile (multiple
              allowed). This file is created with dropbearkey(1) or converted
              from OpenSSH with dropbearconvert(1).

       -L [listenaddress]:listenport:host:port
              Local port forwarding.  Forward the port listenport on the local
              host through the SSH connection to port port on the host host.

       -R [listenaddress]:listenport:host:port
              Remote port forwarding.  Forward the port listenport on the
              remote host through the SSH connection to port port on the host
              host.

       -l user
              Username.  Login as user on the remote host.

       -t     Allocate a PTY.

       -T     Don't allocate a PTY.

       -N     Don't request a remote shell or run any commands. Any command
              arguments are ignored.

       -f     Fork into the background after authentication. A command
              argument (or -N) is required.  This is useful when using
              password authentication.

       -g     Allow non-local hosts to connect to forwarded ports. Applies to
              -L and -R forwarded ports, though remote connections to -R
              forwarded ports may be limited by the ssh server.

       -y     Always accept hostkeys if they are unknown. If a hostkey
              mismatch occurs the connection will abort as normal. If
```

```
              specified a second time no host key checking is performed at
              all, this is usually undesirable.

       -A     Forward agent connections to the remote host. dbclient will use
              any OpenSSH-style agent program if available ($SSH_AUTH_SOCK
              will be set) for public key authentication.  Forwarding is only
              enabled if -A is specified.

       -W windowsize
              Specify the per-channel receive window buffer size. Increasing
              this may improve network performance at the expense of memory
              use. Use -h to see the default buffer size.

       -K timeout_seconds
              Ensure that traffic is transmitted at a certain interval in
              seconds. This is useful for working around firewalls or routers
              that drop connections after a certain period of inactivity. The
              trade-off is that a session may be closed if there is a
              temporary lapse of network connectivity. A setting if 0 disables
              keepalives.

       -I idle_timeout
              Disconnect the session if no traffic is transmitted or received
              for idle_timeout seconds.

       -J proxy_command
              Use the standard input/output of the program proxy_command
              rather than using a normal TCP connection. A hostname should be
              still be provided, as this is used for comparing saved hostkeys.

       -B endhost:endport
              "Netcat-alike" mode, where Dropbear will connect to the given
              host, then create a forwarded connection to endhost. This will
              then be presented as dbclient's standard input/output.

       -c cipherlist
              Specify a comma separated list of ciphers to enable. Use -c help
              to list possibilities.

       -m MAClist
              Specify a comma separated list of authentication MACs to enable.
              Use -m help to list possibilities.

       -s     The specified command will be requested as a subsystem, used for
              sftp. Dropbear doesn't implement sftp itself but the OpenSSH
              sftp client can be used eg sftp -S dbclient user@host

MULTI-HOP
       Dropbear will also allow multiple "hops" to be specified, separated by
       commas. In this case a connection will be made to the first host, then
       a TCP forwarded connection will be made through that to the second
       host, and so on. Hosts other than the final destination will not see
       anything other than the encrypted SSH stream.  A port for a host can be
       specified with a hash (eg matt@martello%44 ).  This syntax can also be
       used with scp or rsync (specifying dbclient as the ssh/rsh command). A
       file can be "bounced" through multiple SSH hops, eg

       scp -S dbclient matt@martello,root@wrt,canyons:/tmp/dump .
```

Note that hostnames are resolved by the prior hop (so "canyons" would be resolved by the host "wrt") in the example above, the same way as other -L TCP forwarded hosts are. Host keys are checked locally based on the given hostname.

ESCAPE CHARACTERS

Typing a newline followed by the key sequence ~. (tilde, dot) will terminate a connection. The sequence ~^Z (tilde, ctrl-z) will background the connection. This behaviour only applies when a PTY is used.

ENVIRONMENT

DROPBEAR_PASSWORD

A password to use for remote authentication can be specified in the environment variable DROPBEAR_PASSWORD. Care should be taken that the password is not exposed to other users on a multi-user system, or stored in accessible files.

SSH_ASKPASS

dbclient can use an external program to request a password from a user. SSH_ASKPASS should be set to the path of a program that will return a password on standard output. This program will only be used if either DISPLAY is set and standard input is not a TTY, or the environment variable SSH_ASKPASS_ALWAYS is set.

AUTHOR

Matt Johnston (matt@ucc.asn.au).
Mihnea Stoenescu wrote initial Dropbear client support
Gerrit Pape (pape@smarden.org) wrote this manual page.

SEE ALSO

dropbear(8), dropbearkey(1)

https://matt.ucc.asn.au/dropbear/dropbear.html