86 - Network DiagnosticsTools

- **ethereal** - Very good Network trafic monitoring program under X-Windows.
  - Configurable listing format. Includes also packets data and filtering features
  - `/usr/X11R6/bin/ethereal` (prgm) from the CD
  - eg. `tethereal -i eth0 port 80`
    `tethereal -i eth0 host 192.168.100.40`
  - `/usr/doc/packages/ethereal/` (docs)
  - `man ethereal` for help file
- **iptraf** - Network trafic monitor. Menu oriented
  - `/usr/sbin/iptraf` (prgm) from the CD
  - `/usr/doc/packages/iptraf/` (docs)
- **tcpdump** - Very good network trafic monitor displays packet headers
  - `/usr/sbin/tcpdump` (prgm) from the CD
  - send output to console or to file via `>` or `>>` or `tee`
  - `/usr/doc/packages/tcpdump/` (docs)
  - `man tcpdump` for help file
- **ngrep** - Very good command line Network Monitor with filtering capabilities
  - `/usr/bin/ngrep` (prgm) no more on SuSE CD
  - usage: `ngrep -qd eth0 port 53` (monitors all DNS(port 53) requests on eth0)
  - `/usr/doc/packages/ngrep/` (docs)
  - `man ngrep` for mor help (see following pages for man pages)
- **tleds** - Display network trafic through keyboard flashing leds: input and output leds
  - `/usr/bin/tleds` (daemon) from the CD
  - `Usage: startproc /usr/bin/tleds -d 100 eth0`
  - `/usr/X11R6/bin/xtleds` (daemon) in the series n
  - `/usr/doc/packages/tleds` (docs)
  - `/usr/X11R6/man/man1/xtleds.1.gz` (docs)
- **ntop** - Very good Network trafic monitor interactive or through http.
  - `/usr/sbin/ntop` (prgm) from the CD
  - usage:
    - interactive mode: `ntop -i eth0`
    - http mode (port 3000): `ntop -i eth0 -w 3000`
      - Uses `~/.ntop` file for users and non-encripted passwords for allowed users
  - `/usr/doc/packages/ntop` (docs)
  - `man ntop` for more help
- **mrtg** - Complex Network trafic visualised through browser. Needs to configure
  - `/usr/bin/mrtg` (prgm) from the CD
  - `/usr/bin/rateup` (prgm)
  - `/usr/doc/packages/mrtg/` (docs)
- **netacct** - Similar to tcpdump logs network trafic by collumns
  - `/usr/sbin/nacctd` (daemon) from the CD
  - `/usr/doc/packages/netacct/` (docs)

- **netcat** - A form of telnet for UDP and TCP. (telnet is only for TCP)
  - `/usr/bin/netcat` (prgm) from the CD
  - usage: `netcat hostname port`
  - `/usr/doc/packages/netcat` (docs)

- **netdiag** - Suite of network cards debugging tools (one per card type)
  - `/sbin/ne2k-pci-diag` and others on CD
  - No docs

- **traf_vis** - Suite of network monitoring tools (programs) as follows:
  - programs are all in `/usr/sbin/` directory
    - `trafic-collector`
    - `trafic-exclude`
    - `trafic-sort`
    - `trafic-togif`
    - `trafic-tohtml`
    - `trafic-tops`
    - `trafic-totext`
  - `/usr/doc/packagesa/traf_vis/` (docs)

- **ksniffer** - KDE (X-Windows) network monitoring tool- Quite good - From kpa series
  - `/opt/kde/bin/ksniffer` (prgm)
  - Help is provided in help menu item.

- **ksnuffle** - KDE (X-Windows) network monitoring tool- Quite good but more complex
  - `/opt/kde/bin/ksnuffle` (prgm) on CD
  - Help is provided in help menu item.

- **knetdump** - KDE (X-Windows) network monitoring tool with graphical tcp/ip packets header view- Quite good -
  - Loads ksniffer from menu item as complementary tool
  - `/opt/kde/bin/knetdump` (prgm)
  - Help is provided in help menu item.

- **knetmon** - KDE (X-Windows) network monitoring tool - Not so good!!
  - `/opt/kde/bin/knetmon` (prgm) from kpa series

- **netstat** - System network diagnostics (mostly already installed with netkita)
  - `netstat -taupe | less` Displays all tcp/ip ready sockets of the system and which program uses them.
  - `watch -n1 netstat -t` Watches dynamically the established tcp connections.

- **nmap** / **xnmap** - TCP/IP Port scanner (package: `nmap`(includes `xnmap`))

- **inetd.conf**'- enable `netstat` and `systat`
  - access the system process status (`ps -ax`) via:

```
telnet IPAddr. netstat
telnet IPAddr. systat
```

NGREP often used parameters
(Capture filter Syntax)

```
ngrep -d eth0 port 23 and \(host sun or laptop\)
```
Detects only packets of port 23 of communication beween sun and laptop

```
ngrep -d eth0 port 23 net 192.168.10
```
Detects only packets of port 23 of communications on the network 192.168.10.x
and not 192.168.11.x or anything else

```
ngrep -d eth0 port 23 net 192.168.10.0 mask 255.255.255.192
```
Detects only packets of port 23 of communications on the
network 192.168.10.0 to 192.168.10.63

```
ngrep 'icmp[0] != 8 and icmp[0] != 0'
```
Detects all other ICMP packets that are NOT Pings and Pongs

```
ngrep 'tcp[13] & 3 != 0 and not src and dst net localnet'
```
To print the start and end packets (the SYN and FIN packets) of each TCP conversation that involves a non-local host.

Ethereal / Tethereal

PROMISCUOUS MODE

`tethereal -i eth0` ----> Captures ALL the packets on the local Network (default).
`tethereal -p -i eth0` ----> Captures only the packets coming in and out of the host.
`tethereal icmp -i eth0` ----> Captures all icmp packets (ping.....etc).

Filters Syntax (Capture and Display)

The default for **Tethereal** is Capture Filter Syntax:

eg . `tethereal -i eth0 port ftp`

It can also use Display Filter Syntax by using the `-R` option as follows:

eg. `tethereal -i eth0 -R tcp.port == ftp`

Capture Filter Format:(see `man tcpdump` or `man ngrep` for all Capture Filters syntax....)

`ether host <hwAdpdr>`	Source or Destination Ethernet Hardware Address eg. `ether host 08:00:20:00:61:CA`
`ether src <hwAddr>`	Source Ethernet Hardware Address
`ether dst <hwAddr>`	Destination Ethernet Hardware Address
`port <port>`	Source or Destination Port
`dst port <port>`	Destination Port
`src port <port>`	Source Port
`host <IP Addr>`	Source or Destination IP Address
`src host <IP Addr>`	Source IP Address
`dst host <IP Addr>`	Destination IP Address

Examples of Capture Filter Syntax:

More complex filter expressions are built up by using the words **and**, **or** and **not** to combine the above primitives.
Primitives parameters names can also be used. Underlined in examples below.
E.g., `host foo and not port ftp and not port ftp-data`
To save typing, identical qualifier lists can be omitted.
E.g., `tcp dst port ftp or ftp-data or domain`
is exactly the same as
`tcp dst port ftp or tcp dst port ftp-data or tcp dst port domain`
E.g.
Catching only the ping requests and replies of any interface:
`'icmp[icmptype] = icmp-echo or icmp[icmptype] = icmp-echoreply'`

Display Filter Syntax:(see `man tethereal` for all Display Filters syntax....a lot of them!!)

`eth.dst`	Destination Hardware (MAC) Address eg . `eth.src == 47-b4-c6-12-46-2b`
`eth.src`	Source Hardware (MAC) Address
`tcp.port`	Source or Destination Port
`tcp.srcport`	Source Port
`tcp.dstport`	Destination Port
`ip.addr`	Source or Destination Address
`ip.src`	Source Address
`ip.dst`	Destination Address

Operators:

`eq`	`==`	Equal to
`ne`	`!=`	Not equal to
`gt`	`>`	Greater than

`lt`	`<`	Less than
`ge`	`>=`	Greater than or Equal to
`le`	`<=`	Less than or Equal to

Example of Display Filter string format:

```
(ip.src eq 192.168.10.155 and tcp.dstport == 80) and ( ip.dst ne 192.168.10.1)
```

Displays all packets that have:

- IP Source address 192.168.10.155 and Destination Port 80
- but not the packet that have the ip Address 192.168.10.1

NOTE: To get the whole TCP Communication shown in a window traced from begin to end. Just right-click on a packet and select **Follow TCP Stream.**

ngrep

NAME

ngrep - network grep (JULY 1999)

SYNOPSIS

`ngrep <-hviwqe> <-n num> <-d dev> <-A num> <regex> <pcap filter logic>`

DESCRIPTION

ngrep strives to provide most of GNU grep's common features, applying them to the network layer. Ngrep is a pcap-aware tool that will allow you to specify extended regular expressions to match against data payloads of packets. It currently recognizes TCP and UDP across eth ernet, ppp and slip interfaces, and understands bpf filter logic in the same fashion as more common packet sniffing tools, such as tcpdump(8) and snoop(1).

OPTIONS

-h Display help/usage information.

-v Display version information.

-i Ignore case for the regex expression.

-w Match the regex expression as a word.

-q Be quiet; don't output any information other than packet headers and their payloads (if relevant).

-e Show empty packets. Normally empty packets are discarded because they have no payload to search.If specified, empty packets will be shown, regardless of the specified regex expression.

-n Match only num packets total, then exit.

-d By default ngrep will select a default interface to listen on. Use this option to force ngrep to listen on interface dev.

-A Dump num packets of trailing context after matching a packet.

pcap filter logic

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

The pcap filter logic consists of one or more primitives. Primitives usually consist of an id (name or number) preceded by one or more qualifiers.

There are three different kinds of qualifier:

type qualifiers say what kind of thing the id name or number refers to. Possible types are host, net and port. E.g., `host foo', `net 128.3', `port 20'.
If there is no type qualifier, host is assumed.

Dir qualifiers specify a particular transfer direction to and/or from id. Possible directions are src, dst, src or dst and src and dst. E.g., `src foo', `dst net 128.3', `src or dst port ftp-data'. If there is no dir qualifier, src or dst is assumed. For `null' link layers (i.e. Point to point protocols such as slip) the inbound and out bound qualifiers can be used to specify a desired direction.

Proto qualifiers are restricted to ip-only protocols. Possible protos are: tcp and udp.

e.g., `udp src foo or tcp port 21`

If there is no proto qualifier, all protocols consistent with the type are assumed.

E.g., `src foo` means `ip and ((tcp or udp) src foo)`

`net bar` means `ip and (net bar)`

`port 53` means `ip and ((tcp or udp) port 53)`

In addition to the above, there are some special `primitive' keywords that don't follow the pattern: `gateway`, `broadcast`, `less`, `greater` and `arithmetic` expressions. All of these are described below.

More complex filter expressions are built up by using the words and, or and not to combine primitives. E.g., `host foo and not port ftp and not port ftp-data'.
To save typing, identical qualifier lists can be omitted.

E.g.,

```
tcp dst port ftp or ftp-data or domain
```

is exactly the same as:

```
tcp dst port ftp or tcp dst port ftp-data or tcp dst port domain.
```

Eg.2

Catching only the ping requests and replies of any interface:

```
'icmp[icmptype] = icmp-echo or icmp[icmptype] = icmp-echoreply'
```

Allowable primitives are:

`dst host` *host*

True if the IP destination field of the packet is host, which may be either an address or a name.

`Src host` *host*

True if the IP source field of the packet is *host*.

Host *host*

True if either the IP source or destination of the packet is host. Any of the above host expressions can be prepended with the keywords, ip, arp, or rarp as in: `ip host` *host*

ether dst *ehost*

True if the ethernet destination address is ehost. Ehost may be either a name from `/etc/ethers` or a number (see ethers(3N) for numeric format).

Ether src *ehost*

True if the ethernet source address is *ehost*.

Ether host *ehost*

True if either the ethernet source or desti nation address is *ehost*.

Gateway *host*

True if the packet used *host* as a gateway.

I.e., the ethernet source or destination address was host but neither the IP source nor the IP destination was host. Host must be a name and must be found in both `/etc/hosts` and `/etc/ethers`.

An equivalent expression is:

`ether host` *ehost* `and not host` *host*

which can be used with either names or numbers for `host/ehost`.

dst net *net*

True if the IP destination address of the packet has a network number of net. Net may be either a name from /etc/networks or a network number (see networks(4) for details).

Src net *net*

True if the IP source address of the packet has a network number of *net*.

Net *net*

True if either the IP source or destination address of the packet has a network number of *net*.

Net net mask *mask*

True if the IP address matches net with the specific netmask.
May be qualified with src or dst.

Net *net/len*

True if the IP address matches net a netmask *len* bits wide.
May be qualified with src or dst.

Dst port *port*

True if the packet is ip/tcp or ip/udp and has a destination port value of port. The port can be a number or a name used in /etc/services (see tcp(4P) and udp(4P)). If a name is used, both the port number and protocol are checked. If a number or ambiguous name is used, only the port number is checked (e.g., `dst port 513` will print both tcp/login traffic and udp/who traffic, and port domain will print both tcp/domain and udp/domain traffic).

Src port *port*

True if the packet has a source port value of port.

Port *port*

True if either the source or destination port of the packet is port. Any of the above port expressions can be prepended with the keywords, tcp or udp, as in:

tcp src port *port*

which matches only tcp packets whose source port is *port*.

Less *length*

True if the packet has a length less than or equal to length.
This is equivalent to: len <= length.

Greater *length*

True if the packet has a length greater than or equal to length.
This is equivalent to: len >= length.

Ip proto *protocol*

True if the packet is an ip packet (see ip(4P)) of protocol type protocol. Protocol can be a number or one of the names udp or tcp. Note that the identifiers tcp and udp are also keywords and must be escaped via backslash (\), which is \\ in the C-shell.

Ip *broadcast*

True if the packet is an IP broadcast packet. It checks for both the all-zeroes and all-ones broadcast conventions, and looks up the local subnet mask.

Ip *multicast*

True if the packet is an IP multicast packet.

Ip Abbreviation for: `ether proto ip`

tcp, udp Abbreviations for: `ip proto` *p*

where *p* is one of the above protocols.

Expr *relop expr*

True if the relation holds, where relop is one of >, <, >=, <=, =, !=, and expr is an arithmetic expression composed of integer constants (expressed in standard C syntax), the normal binary operators [+, -, *, /, &, |], a length operator, and special packet data accessors. To access data inside the packet, use the following syntax:

proto [*expr* **:** *size* **]**

Proto is one of ether, fddi, ip, tcp, or udp, and indicates the protocol layer for the index operation. The byte offset, relative to the indicated protocol layer, is given by expr. Size is optional and indicates the number of bytes in the field of interest; it can be either one, two, or four, and defaults to one. The length operator, indicated by the keyword len, gives the length of the packet.

For example, `ether[0] & 1 != 0' catches all multicast traffic. The expression `ip[0]

`& 0xf != 5'` catches all IP packets with options. The expression `` `ip[6:2] & 0x1fff = 0' `` catches only unfragmented datagrams and frag zero of fragmented datagrams. This check is implicitly applied to the tcp and udp index operations. For instance, `tcp[0]` always means the first byte of the TCP header, and never means the first byte of an intervening fragment.

Primitives may be combined using:
A parenthesized group of primitives and operators (parentheses are special to the Shell and must be escaped).

Negation	(`!' or `not').
Concatenation	(`&&' or `and').
Alternation	(`||' or `or').

Negation has highest precedence. Alternation and concatenation have equal precedence and associate left to right. Note that explicit and tokens, not juxtaposition, are now required for concatenation.

If an identifier is given without a keyword, the most recent keyword is assumed. For example, not host vs and ace is short for not host vs and host ace which should not be confused with not (host vs or ace)

Expression arguments can be passed to ngrep as either a single argument or as multiple arguments, whichever is more convenient. Generally, if the expression contains Shell metacharacters, it is easier to pass it as a single, quoted argument.

Multiple arguments are concatenated with spaces before being parsed.

DIAGNOSTICS

Errors from ngrep, libpcap, and the GNU regex library are all output to stderr.

AUTHOR

Jordan Ritter <jpr5@darkridge.com>

BUGS

None known at this time.