

# Ngrep Quick Tutorial

Mark E. Donaldson

## ngrep - network grep

Basic Packet Sniffing | Debugging HTTP Interactions | Processing PCAP dump files | Observing Binary Protocols

### Example: Basic Packet Sniffing

Basic packet sniffing is easy with ngrep. It supports BPF filter logic, which means to say constraining what ngrep sees and displays is as easy as saying something like ``ngrep host foo.bar.com and port 25''. Following are a few examples of common invocations of ngrep to do basic packet sniffing. Please note the usage of ``any'' as the specified ethernet adaptor to attach to; in most recent UNIX libpcap implementations this will instruct ngrep to attach to all interfaces at once, local (lo) and all external interfaces that may be active.

Monitor all activity crossing source or destination port 25 (SMTP).

```
ngrep -d any port 25
```

Monitor any network-based syslog traffic for the occurrence of the word ``error''. ngrep knows how to convert service port names (on UNIX, located in ``/etc/services'') to port numbers.

```
ngrep -d any 'error' port syslog
```

Monitor any traffic crossing source or destination port 21 (FTP), looking case-insensitively for the words ``user'' or ``pass'', matched as word-expressions (the match term(s) must have non-alphanumeric, delimiting characters surrounding them).

```
ngrep -wi -d any 'user|pass' port 21
```

### Example: Debugging HTTP interactions

In certain scenarios it is desirable to see how web browsers communicate with web servers, and to inspect the HTTP headers and possibly cookie values that they are exchanging.

In this example, we run an ngrep on a webserver. Since it only has one interface, eth0, we omit specifying the interface manually on the command line and allow ngrep to choose the default interface for us, for convenience.

```
# ngrep port 80
interface: eth0 (64.90.164.72/255.255.255.252)
filter: ip and ( port 80 )
####
T 67.169.59.38:42167 -> 64.90.164.74:80 [AP]
GET / HTTP/1.1..User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; X11; Linux i
686) Opera 7.21 [en]..Host: www.darkridge.com..Accept: text/html, applicat
ion/xml;q=0.9, application/xhtml+xml;q=0.9, image/png, image/jpeg, image/gi
f, image/x-xbitmap, */*;q=0.1..Accept-Charset: iso-8859-1, utf-8, utf-16, *
;q=0.1..Accept-Encoding: deflate, gzip, x-gzip, identity, *;q=0..Cookie: SQ
MSESSID=5272f9ae21c07eca4dfd75f9a3cda22e..Cookie2: $Version=1..Connection:
Keep-Alive, TE..TE: deflate, gzip, chunked, identity, trailers....
##
T 64.90.164.74:80 -> 67.169.59.38:42167 [AP]
HTTP/1.1 200 OK..Date: Mon, 29 Mar 2004 00:44:40 GMT..Server: Apache/2.0.49
(Unix)..Last-Modified: Tue, 04 Nov 2003 12:09:41 GMT..ETag: "210e23-326-f8
200b40"..Accept-Ranges: bytes..Vary: Accept-Encoding,User-Agent..Content-En
```

# Ngrep Quick Tutorial

Mark E. Donaldson

```
coding: gzip..Content-Length: 476..Keep-Alive: timeout=15, max=100..Connect
ion: Keep-Alive..Content-Type: text/html; charset=ISO-8859-1..Content-Langu
age: en.....}S]..0.|.....H...8.....@..\.....(.....Dw.%,...
;k.....Y>q<.....d .....3.i..kdm.u@d{.Q..\....@..B1.0.2YI^..R.....
...X.....X..y..\.....,.(.....1...g.....*...j..a.`_@.W....0.....?.
.R.K.j..Y.....>...;kw*U.j.<... \0Tn.l.:.....>Fs.....'.....h.'...u.H4...'..6.vIDI.....N.r .....}...I.w. ...mX...L.s...{.L.R.-...e....~nu..t.3...H..#..J...
.u?...].....^..2.....e8v/gP.....].48...qD!.....#y...m}..>/?...#.....I
..I..4.P.....2:....n8l.....!.Yr&...
```

##

As you can see, all headers and aspects of the HTTP transmission are exposed in their gory detail. It's a little hard to parse though, so let's see what happens when ``-W byline" mode is used:

```
# ngrep -W byline port 80
interface: eth0 (64.90.164.72/255.255.255.252)
filter: ip and ( port 80 )
####
T 67.169.59.38:42177 -> 64.90.164.74:80 [AP]
GET / HTTP/1.1.
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; X11; Linux i686) Opera ...
Host: www.darkridge.com.
Accept: text/html, application/xml;q=0.9, application/xhtml+xml;q=0.9 ...
Accept-Charset: iso-8859-1, utf-8, utf-16, *;q=0.1.
Accept-Encoding: deflate, gzip, x-gzip, identity, *;q=0.
Cookie: SQMSESSID=5272f9ae21c07eca4dfd75f9a3cda22e.
Cookie2: $Version=1.
Cache-Control: no-cache.
Connection: Keep-Alive, TE.
TE: deflate, gzip, chunked, identity, trailers.
.
```

```
##
T 64.90.164.74:80 -> 67.169.59.38:42177 [AP]
HTTP/1.1 200 OK.
Date: Mon, 29 Mar 2004 00:47:25 GMT.
Server: Apache/2.0.49 (Unix).
Last-Modified: Tue, 04 Nov 2003 12:09:41 GMT.
ETag: "210e23-326-f8200b40".
Accept-Ranges: bytes.
Vary: Accept-Encoding,User-Agent.
Content-Encoding: gzip.
Content-Length: 476.
Keep-Alive: timeout=15, max=100.
Connection: Keep-Alive.
Content-Type: text/html; charset=ISO-8859-1.
Content-Language: en.
```

```
.....}S]..0.|.....H...8.....@..\.....(.....Dw.%,...;k.. ...
.;kw*U.j.<... \0Tn.l.:.....>Fs.....'.....h.'...u.H4...'..6.vIDI.....N.r ...
..H..#..J....u?...].....^..2.....e8v/gP.....].48...qD!.....#y...m ...
####
```

# Ngrep Quick Tutorial

Mark E. Donaldson

"-W byline" mode tells ngrep to respect embedded line feeds when they occur. You'll note from the output above that there is still a trailing dot (`. `) on each line, which is the carriage-return portion of the CRLF pair. Using this mode, now the output has become much easier to visually parse.

## Example: Processing PCAP dump files, looking for patterns

I had a friend who worked at Network Solutions and among the things he did was analyze huge 500M+ PCAP dump files of DNS traffic, looking for patterns and anomalies. ngrep was an invaluable tool for this purpose; it allowed him to take one instance of a network dump and search it quickly and repeatedly for patterns in the data packets.

To save a PCAP dump file from ngrep is very easy; simply run ngrep as you normally would but add one more command line option: ``-O some.file.dump" (the name of the file is largely irrelevant). To illustrate another feature of ngrep, we will use the ``-T" option (print time differential information).

```
# ngrep -O /tmp/dns.dump -d any -T port domain
interface: any
filter: ip and ( port domain )
output: /tmp/dns.dump
#
U +0.000000 203.115.225.24:53 -> 64.90.164.74:53
.....m.razor2.cloudmark.com.....).....
#
U +0.000281 64.90.164.74:53 -> 203.115.225.24:53
.....m.razor2.cloudmark.com.....'.ns1...hostmaster..ws..
..P.... ..:.....).....
#
U +0.078184 195.113.155.7:2949 -> 64.90.164.74:53
.....a.razor2.cloudmark.com.....
#
U +0.000351 64.90.164.74:53 -> 195.113.155.7:2949
.....a.razor2.cloudmark.com.....agony...4.....B..
.....ns1.....ns2.....ns3...X.....@Z.J.j..
.....@Z...|.....B.;
^Cexit
6 received, 0 dropped
```

Note the ``output:" indicator and timestamp information. Now we have a PCAP dump file, and so let's search it for some patterns:

```
# ngrep -w 'm' -I /tmp/dns.dump
input: /tmp/dns.dump
match: ((^m\W)|(\Wm$)|(\Wm\W))
#
U 203.115.225.24:53 -> 64.90.164.74:53
.....m.razor2.cloudmark.com.....).....
#
U 64.90.164.74:53 -> 203.115.225.24:53
.....m.razor2.cloudmark.com.....'.ns1...hostmaster..ws..
..P.... ..:.....).....
##exit
```

Above we searched for the letter `m', matched as a word (``-w'). This yields two packets.

# Ngrep Quick Tutorial

Mark E. Donaldson

```
# ngrep -tD ns3 -I /tmp/dns.dump
input: /tmp/dns.dump
match: ns3
####
U 2004/03/28 20:32:37.088525 64.90.164.74:53 -> 195.113.155.7:2949
.....a.razor2.cloudmark.com.....agony...4.....B..
.....nsl.....ns2.....ns3...X.....@Z.J.j..
.....@Z...|.....B..;
exit
```

Here we've added ``-t" which means print the absolute timestamp on the packet, and ``-D" which means replay the packets by the time interval at which they were recorded. The latter is a neat little feature for observing the traffic at the rates/times they originally seen, though in this example it's not terribly effective as there is only one packet being matched.

```
# ngrep -I /tmp/dns.dump port 80
input: /tmp/dns.dump
filter: ip and ( port 80 )
exit
```

There's no port 80 traffic in the dump, so of course the BPF filter yields us no results.

## Example: Observing binary being transferred across the wire

One interesting feature of ngrep is its ability to take a hexademical (binary) expression and search for that in lieu of a regular expression. ngrep can also display the packets it observes in a hexadecimal format, which is more effective for inspecting binary content patterns.

In this example, we will simply look for a binary pattern in a web stream, but the more obvious usage is to look for a DDoS Zombie's unique binary signature (say, from a command packet), or even a Worm/Virus being transferred across the wire as it propagates itself.

For this test, let's assume we have a GIF on a web server that has the data pattern ``0xc5d5e5f55666768696a6b6c6d6e6" (hexademical) in it. Once ``-X" is specified, the expression will be interpreted as a hexademical pattern instead of a regular expression, and the ``0x" prefix is optional.

To see a packet like this cross the wire:

```
# ngrep -xX '0xc5d5e5f55666768696a6b6c6d6e6' port 80
interface: eth0 (64.90.164.72/255.255.255.252)
filter: ip and ( port 80 )
match: 0xc5d5e5f55666768696a6b6c6d6e6
###
T 64.90.164.74:80 -> 67.169.59.38:42306 [A]
ff d8 ff e0 00 10 4a 46      49 46 00 01 02 01 00 48      .....JFIF.....H
00 48 00 00 ff ed 13 ba      50 68 6f 74 6f 73 68 6f      .H.....Photosho
70 20 33 2e 30 00 38 42      49 4d 03 ed 00 00 00 00      p 3.0.8BIM.....
00 10 00 48 00 00 00 01      00 01 00 48 00 00 00 01      ...H.....H....
00 01 38 42 49 4d 04 0d      00 00 00 00 00 04 00 00      ..8BIM.....
00 78 38 42 49 4d 03 f3      00 00 00 00 00 08 00 00      .x8BIM.....
00 00 00 00 00 00 38 42      49 4d 04 0a 00 00 00 00      .....8BIM.....
00 01 00 00 38 42 49 4d      27 10 00 00 00 00 00 0a      ....8BIM'.....
00 01 00 00 00 00 00 00      00 02 38 42 49 4d 03 f5      .....8BIM..
```

# Ngrep Quick Tutorial

Mark E. Donaldson

```
00 00 00 00 00 48 00 2f      66 66 00 01 00 6c 66 66      .....H./ff...lff
00 06 00 00 00 00 00 01      00 2f 66 66 00 01 00 a1      ...../ff....
99 9a 00 06 00 00 00 00      00 01 00 32 00 00 00 01      .....2....
00 5a 00 00 00 06 00 00      00 00 00 01 00 35 00 00      .Z.....5..
00 01 00 2d 00 00 00 06      00 00 00 00 00 01 38 42      ...-.....8B
49 4d 03 f8 00 00 00 00      00 70 00 00 ff ff ff ff      IM.....p.....
ff ff ff ff ff ff ff ff      ff ff ff ff ff ff ff ff      .....
ff ff 03 e8 00 00 00 00      ff ff ff ff ff ff ff ff      .....
ff ff ff ff ff ff ff ff      ff ff ff ff ff ff 03 e8      .....
00 00 00 00 ff ff ff ff      ff ff ff ff ff ff ff ff      .....
ff ff ff ff ff ff ff ff      ff ff 03 e8 00 00 00 00      .....
ff ff ff ff ff ff ff ff      ff ff ff ff ff ff ff ff      .....
ff ff ff ff ff ff 03 e8      00 00 38 42 49 4d 04 08      .....8BIM..
00 00 00 00 00 10 00 00      00 01 00 00 02 40 00 00      .....@..
02 40 00 00 00 00 38 42      49 4d 04 14 00 00 00 00      .@....8BIM.....
00 04 00 00 00 06 38 42      49 4d 04 0c 00 00 00 00      .....8BIM.....
12 2a 00 00 00 01 00 00      00 70 00 00 00 57 00 00      .*.....p...W..
01 50 00 00 72 30 00 00      12 0e 00 18 00 01 ff d8      .P..r0.....
ff e0 00 10 4a 46 49 46      00 01 02 01 00 48 00 48      ....JFIF.....H.H
00 00 ff fe 00 26 46 69      6c 65 20 77 72 69 74 74      .....&File writt
65 6e 20 62 79 20 41 64      6f 62 65 20 50 68 6f 74      en by Adobe Phot
6f 73 68 6f 70 a8 20 35      2e 30 ff ee 00 0e 41 64      oshop. 5.0....Ad
6f 62 65 00 64 80 00 00      00 01 ff db 00 84 00 0c      obe.d.....
08 08 08 09 08 0c 09 09      0c 11 0b 0a 0b 11 15 0f      .....
0c 0c 0f 15 18 13 13 15      13 13 18 11 0c 0c 0c 0c      .....
0c 0c 11 0c 0c 0c 0c 0c      0c 0c 0c 0c 0c 0c 0c 0c      .....
0c 0c 0c 0c 0c 0c 0c 0c      0c 0c 0c 0c 0c 0c 0c 0c      .....
0d 0b 0b 0d 0e 0d 10 0e      0e 10 14 0e 0e 0e 14 14      .....
0e 0e 0e 0e 14 11 0c 0c      0c 0c 0c 11 11 0c 0c 0c      .....
0c 0c 0c 11 0c 0c 0c 0c      0c 0c 0c 0c 0c 0c 0c 0c      .....
0c 0c 0c 0c 0c 0c 0c 0c      0c 0c 0c 0c 0c 0c 0c 0c      .....
ff c0 00 11 08 00 57 00      70 03 01 22 00 02 11 01      .....W.p.."....
03 11 01 ff dd 00 04 00      07 ff c4 01 3f 00 00 01      .....?....
05 01 01 01 01 01 01 00      00 00 00 00 00 00 03 00      .....
01 02 04 05 06 07 08 09      0a 0b 01 00 01 05 01 01      .....
01 01 01 01 00 00 00 00      00 00 00 01 00 02 03 04      .....
05 06 07 08 09 0a 0b 10      00 01 04 01 03 02 04 02      .....
05 07 06 08 05 03 0c 33      01 00 02 11 03 04 21 12      .....3.....!.
31 05 41 51 61 13 22 71      81 32 06 14 91 a1 b1 42      1.AQa."q.2.....B
23 24 15 52 c1 62 33 34      72 82 d1 43 07 25 92 53      #$.R.b34r..C.%S
f0 e1 f1 63 73 35 16 a2      b2 83 26 44 93 54 64 45      ...cs5....&D.TdE
c2 a3 74 36 17 d2 55 e2      65 f2 b3 84 c3 d3 75 e3      ..t6..U.e.....u.
f3 46 27 94 a4 85 b4 95      c4 d4 e4 f4 a5 b5 c5 d5      .F'.....
e5 f5 56 66 76 86 96 a6      b6 c6 d6 e6 f6 37 47 57      ..Vfv.....7GW
67 77 87 97 a7 b7 c7 d7      e7 f7 11 00 02 02 01 02      gw.....
04 04 03 04 05 06 07 07      06 05 35 01 00 02 11 03      .....5.....
21 31 12 04 41 51 61 71      22 13 05 32 81 91 14 a1      !1..AQaq"..2....
b1 42 23 c1 52 d1 f0 33      24 62 e1 72 82 92 43 53      .B#.R..3$b.r..CS
15 63 73 34 f1 25 06 16      a2 b2 83 07 26 35 c2 d2      .cs4.%.....&5..
44 93 54 a3 17 64 45 55      36 74 65 e2 f2 b3 84 c3      D.T..dEU6te.....
d3 75 e3 f3 46 94 a4 85      b4 95 c4 d4 e4 f4 a5 b5      .u..F.....
c5 d5 e5 f5 56 66 76 86      96 a6 b6 c6 d6 e6 f6 27      ...Vfv.....'
37 47 57 67 77 87 97 a7      b7 c7 ff da 00 0c 03 01      7GWgw.....
00 02 11 03 11 00 3f 00      f2 a5 3a ad 35 ba 40 0e      .....?.....:5.@.
04 16 90 78 20 a8 25 07      94 aa d3 19 18 90 41 a2      ...x .%.....A.
13 9a 4b 9b b9 a0 91 c8      3d c8 ef a7 f2 14 46 35      ..K.....=.....F5
af fe 6c 6f f8 73 e3 3b      7e 92 6a ad 2c 30 75 64      ..lo.s.;~.j.,0ud
```

# Ngrep Quick Tutorial

Mark E. Donaldson

```
82 47 fd f9 a7 f3 5c 8a      ec d7 b5 e4 d2 4b 79 0d      .G....\.....Ky.
73 a0 ba 3f f2 49 87 8b      61 4d 88 fd de 40 4a 66      s..?.I..aM...@Jf
51 fd e8 c7 e6 ff 00 03      f4 5a ee 63 d8 76 bd a5      Q.....Z.c.v..
a4 76 22 13 29 d9 75 b6      99 b1 ee 7c 71 b8 ca 82      .v".).u....|q...
78 be ad 79 70 f1 1e 1b      e1 e9 c5 f3 29 24 92 49      x..yp.....)$.I
0a 49 24 92 52 92 45 c7      c4 bf 25 c5 b4 b7 76 d1      .I$.R.E...%...v.
2e 3c 00 3f 94 ef a2 d5      6f 33 a3 64 e1 63 7a f9      .<.?....o3.d.cz.
0f a9 85 c5 bb 29 f5 18      eb 1c 1c 1d b9 e2 ba 9c      .....).....
ff 00 63 36 fe 7a 69 c9      00 44 4c 87 11 da 3d 57      ..c6.zi..DL...=W
8c 59 0c 4c c4 4f 08 fd      2e 8d 3a da e7 1d 8d 11      .Y.L.O.....:.....
22 75 47 ca fb 35 78 d5      d2 c2 1f 7c 87 58 f6 ea      "uG..5x....|.X..
06 91 e9 ef fc e4 1b 5f      4c 33 d1 05 a7 68 0f 27      ....._L3...h.'
b9 fc e8 42 4a ac 83 a8      ae 8c 9e e0 84 65 00 23      ...BJ.....e.#
23 21 5c 7f 37 0c 7e 6f      47 f5 9f ff d0 f2 ae ca      #!\.7.~oG.....
62 36 c1 3a 1f c0 84 cd      69 71 81 c9 47 a6 f6 e3      b6.:....iq..G...
3f 75 41 af 78 e1 ef 12      27 fe 0d bf f9 24 f1 3e      ?uA.x...'....$.>
0d 40 e2 24 55 7f 15 f0      80 91 1c 52 e0 85 eb 2a      .@.$U.....R...*
e2 ff 00 16 3f a4 c2 fc      5c 8a 1a c7 da c2 c6 d9      .....?\.....
f4 67 c9 05 5f ca ea 37      3c fa 77 1a b2 1b e2 01      .g..._...7<.w.....
81 3d 83 bd 8a 2e a8 67      39 b5 e0 63 90 e6 34 9b      .=.....g9..c..4.
00 20 ff 00 68 f1 ec 67      ef a8 63 29 50 e3 00 5f      . .h..g..c)P.._
51 f2 b3 f3 18 79 70 66      70 e5 26 30 e1 a8 65 1f      Q....ypfp.&0..e.
ad 9c bf 4e 8e 3e 2c 5f      f3 d8 62 f4 dc ac a6 ef      ...N.>/_..b.....
a8 02 c0 40 73 8b 86 93      fb df 9c b6 3a 66 36 0d      ...@s.....:f6.
6c 73 18 45 b7 6a 2c de      ls.E.j,.
```

#####

Above we specified ``-X" to tell ngrep to treat the match expression as hexadecimal, and ``-x" to tell ngrep to print out the patterns it matches in hexadecimal form.

As it turns out, several other packets also matched this pattern, but this should give the prospective user a good idea of how to use hexadecimal patterns and the hex output mode.

## NGREP

Updated: December 2001

NAME

ngrep - network grep

## SYNOPSIS

```
ngrep <-hXViqpevxIDtT> <-IO pcap_dump > <-n num > <-d dev > <-A num > <-s snaplen > <-S limitlen > < match expression > < bpf filter >
```

## 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, UDP and ICMP across Ethernet, PPP, SLIP, FDDI and null interfaces, and understands bpf filter logic in the same fashion as more common packet sniffing tools, such as tcpdump(8) and snoop(1).

# Ngrep Quick Tutorial

Mark E. Donaldson

## OPTIONS

- h Display help/usage information.
- X Treat the match expression as a hexadecimal string. See the explanation of *match expression* below.
- 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).
- p Don't put the interface into promiscuous mode.
- 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.
- v Invert the match; only display packets that don't match.
- x Dump packet contents as hexadecimal as well as ASCII.
- l Make stdout line buffered.
- D When reading pcap\_dump files, replay them at their recorded time intervals (mimic realtime).
- t Print a timestamp in the form of YYYY/MM/DD HH:MM:SS.UUUUUU everytime a packet is matched.
- T Print a timestamp in the form of +S.UUUUUU, indicating the delta between packet matches.
- s snaplen  
Set the bpf caplen to snaplen (default 65536).
- S limitlen  
Set the upper limit on the size of packets that ngrep will look at. Useful for looking at only the first N bytes of packets without changing the BPF snaplen.
- I pcap\_dump  
Input file pcap\_dump into ngrep. Works with any pcap-compatible dump file format. This option is useful for searching for a wide range of different patterns over the same packet stream.
- O pcap\_dump  
Output matched packets to a pcap-compatible dump file. This feature does not interfere with normal output to stdout.
- n num  
Match only *num* packets total, then exit.
- d dev

# Ngrep Quick Tutorial

Mark E. Donaldson

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

**-A num**

Dump *num* packets of trailing context after matching a packet.

**match expression**

A match expression is either an extended regular expression, or if the **-X** option is specified, a string signifying a hexadecimal value. An extended regular expression follows the rules as implemented by the **GNU regex library**. Hexadecimal expressions can optionally be preceded by ``0x'`. E.g., ``DEADBEEF'`, ``0xDEADBEEF'`.

**bpf filter**

Selects a filter that specifies what packets will be dumped. If no *bpf filter* is given, all IP packets seen on the selected interface will be dumped. Otherwise, only packets for which *bpf filter* is ``true'` will be dumped.

The *bpf filter* 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 blort'`, ``net 1.2.3'`, ``port 80'`. 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 1.2.3'`, ``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 **outbound** qualifiers can be used to specify a desired direction.

**proto**

qualifiers are restricted to ip-only protocols. Possible protos are: **tcp**, **udp** and **icmp**. 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)'`, and ``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 blort 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'`.

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*

# Ngrep Quick Tutorial

Mark E. Donaldson

which is equivalent to:

**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 destination 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**  $\leq$  *length*.

**greater** *length*

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

**len**  $\geq$  *length*.

**ip proto** *protocol*

# Ngrep Quick Tutorial

Mark E. Donaldson

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 *tcp*, *udp* or *icmp*. 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, icmp

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 **ip**, **tcp**, **udp** or **icmp**, 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,

# Ngrep Quick Tutorial

Mark E. Donaldson

## **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.