Generate a HOSTS file (like /etc/hosts) based on DNS lookups in a PCAP file:

```
tshark -r dump.pcap -q -z hosts > hosts.txt
```

Print Protocol Hierarchy Statistics (PHS) listing for all traffic in dump.pcap

```
tshark -r dump.pcap -q -z io,phs
```

== NGREP ==

```
ngrep <-iqvx> <-IO pcap_dump > < -n num > <
    match expression > < bpf filter >
```

-i   Ignore case for the regex expression.

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

-v   Invert the match; only display packets that don't match.

-x   Dump packet contents as hexadecimal as well as ASCII.

-I   pcap_dump
Input file pcap file into ngrep.

-O   pcap_dump
Output matched packets to a pcap file.

-n   num
Match only num packets total, then exit.

match expression
A match expression is an extended regular expression.

bpf filter
Selects a filter that specifies what packets will be dumped.

EXAMPLES

Search a PCAP file for packets containing the email address "user@internet.se"

```
ngrep -I dump.pcap -q user@internet.se
```

Search for DNS requests (to port 53) for "pwned.se"

```
ngrep -I snort.log.1428364808 -q -i pwned.se dst port 53
```

SWEDISH ARMED FORCES

www.forsvarsmakten.se

Unzip the VirtualBox machine from Hands-on_Network_Forensics.zip on your USB thumb drive to your local hard drive

Start VirtualBox and run the Security Onion VM

Usernames/Passwords

Security Onion VM
user / password

ELSA : https://127.0.0.1/elsa/
user / password

Squert : https://127.0.0.1/squert/
user / password

Snorby : https://127.0.0.1:444/
user@internet.se / password

Xplico : https://127.0.0.1:9876/
xplico / xplico

Paths

PCAP files:
/nsm/sensor_data/securityonion_eth1/dailylogs/
Argus files:
/nsm/sensor_data/securityonion_eth1/argus/
Bro-IDS logs:
/nsm/bro/logs/
ip whitelist.py
/usr/local/bin/ip_whitelist.py

```
                                             [INTERNET]
                                                 |
                                             Default Gateway
                                             192.168.0.1
       PASSWORD-NED-XP      www.pwned.se         |
       192.168.0.53         192.168.0.2        [TAP]--->Security-
             |                   |               |      Onion
 -----+------+---------+---------+---------------+-------
      |                |
Homer-xubuntu      Krustys-PC
192.168.0.51       192.168.0.54
```

27th ANNUAL FIRST BERLIN CONFERENCE 14-19 JUNE 2015

SWE.MIL.CERT
SWEDISH ARMED FORCES COMPUTER EMERGENCY RESPONSE TEAM

Hands-on Network Forensics Workshop Cheat Sheet

== ARGUS ==

```
ra [options] [-- filter-expression]
```

-n   Suppress port number to service conversion.

-r   [- | <file file ...>]
Read data from <files> in the order presented on the commandline. '-' denotes stdin (default).

-R   <dir dir ...>
Recursively descend the directory and process all the regular files that are encountered.

-w   <file>
Append matching data to <file>, in argus file format. An output-file of '-' directs ra to write the argus(5) records to stdout, allowing for "chaining" ra* style commands together.

```
racluster [-m aggregation-objects][options]
[-- filter-expression]
```

Supported aggregation-objects are:

saddr/[l|m]	source IP addr/[cidr len | m.a.s.k].
daddr/[l|m]	destination IP addr/[cidr len | m.a.s.k].
proto	transaction protocol.
sport	source port number. Implies use of 'proto'.
dport	destination port number. Implies use of 'proto'.

```
rasort [-m sort-fields] [options] [-- filter-expression]
```

Supported sort-fields are:

stime	record start time <default>
dur	record total duration.
saddr[/cidr]	source IP addr, with optional cidr specification for IPv4 addresses.
daddr[/cidr]	destination IP addr, with optional cidr specification for IPv4 addresses.
sport	source port number.
dport	destination port number.
bytes	total transaction bytes.
sbytes	src -> dst transaction bytes.
dbytes	dst -> src transaction bytes.
pkts	total transaction packet count.
spkts	src -> dst packet count.
dpkts	dst -> src packet count.

```
rafilteraddr   [-f address.file] [-v] [options]
               [-- filter-expression]
```

-v Invert the logic and print flows that don't match any of the addresses.

EXAMPLES

List all flows to/from the class C network 217.195.49.0/24 in chronological order based on start time:

```
racluster -R * -w - -- net 217.195.49.0/24 |
rasort -m stime -n
```

List all flows to/from 192.168.0.53, where the remote IP is not listed in ip_whitelist.txt. Sort flows based on bytes sent from the server:

```
rafilteraddr -R * -v -f /usr/local/etc/
ip_whitelist.txt -w - -- host 192.168.0.53 |
racluster -w - | rasort -m dbytes -n
```

== TCPDUMP ==

```
tcpdump [ -n ] [ -c count ] [ -i interface ] [ -r file ] [ -w file ] [ filter-expression ]
```

-c Exit after receiving count packets.

-i Sniff packets from interface.

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

-r Read packets from file.

-w Write the raw packets to file rather than parsing and printing them out.

EXAMPLES

Sniff and print DNS packets to stdout:

```
tcpdump -i eth0 -n port 53
```

Capture 100 packets from eth0 to sniffed.pcap:

```
tcpdump -i eth0 -c 100 -w sniffed.pcap
```

Filter a PCAP file to only include traffic to/from 217.195.49.146 into a new PCAP file:

```
tcpdump -r snort.log.1426118407 -w /var/
tmp/217.195.49.146.pcap host 217.195.49.146
```

== TCPFLOW ==

```
Tcpflow [-BcC] [-AH] [-b max_bytes] [-i iface]
    [-r file1.pcap] [expression]
```

-B Force binary output even when printing to console with -C or -c.

-b Capture no more than max_bytes bytes per flow.

-c Console print (stdout), without storing any captured data to files

-C Console print without the packet source and destination details being printed.

-AH Perform HTTP post-processing ("After" processing) to extract HTTP payloads.

-i Capture packets from the network interface named iface.

-r Read from PCAP file.

EXAMPLE

Extract contents of POP3 sessions (TCP 110):

```
tcpflow -r emails.pcap port 110
```

== TSHARK ==

```
tshark [ -c <packet count> ] [ -e <field> ] [ -n ] [ -q ] [ -r <infile> ] [ -R <read (display)
filter> ] [ -T fields ][ -w <outfile>|- ] [ -x ]
[ -z <statistics> ]
```

-c <packet count>
Set the maximum number of packets to read.

-e <field>
Add a field to the list of fields to display if -T fields is selected.

-n Disable network object name resolution (such as hostname, TCP and UDP port names).

-q Don't print packet information; this is useful if you're using a -z option to calculate statistics and don't want the packet information printed, just the statistics.

-r <infile>
Read packet data from infile.

-R <read (display) filter>
Cause the specified filter to be applied.

-T fields
Set the format of the output when viewing decoded packet data. The values of fields specified with the -e option.

-w <outfile> | -
Write raw packet data to outfile or to the standard output if outfile is '-'.

-x Cause TShark to print a hex and ASCII dump of the packet data after printing the summary or details.

-z <statistics>
Get TShark to collect various types of statistics and display the result after finishing reading the capture file. Use the -q flag if you're reading a capture file and only want the statistics printed.

EXAMPLES

Print client IP and HTTP URI for all HTTP requests containing the string "index.html":

```
tshark -r dump.pcap -R "http.request.uri contains index.html" -T fields -e ip.src -e
http.request.uri
```

TCPDUMP

packetlife.net

Command Line Options

`-A`	Print frame payload in ASCII	`-q`	Quick output
`-c <count>`	Exit after capturing **count** packets	`-r <file>`	Read packets from **file**
`-D`	List available interfaces	`-s <len>`	Capture up to **len** bytes per packet
`-e`	Print link-level headers	`-S`	Print absolute TCP sequence numbers
`-F <file>`	Use **file** as the filter expression	`-t`	Don't print timestamps
`-G <n>`	Rotate the dump file every n seconds	`-v[v[v]]`	Print more verbose output
`-i <iface>`	Specifies the capture interface	`-w <file>`	Write captured packets to **file**
`-K`	Don't verify TCP checksums	`-x`	Print frame payload in hex
`-L`	List data link types for the interface	`-X`	Print frame payload in hex and ASCII
`-n`	Don't convert addresses to names	`-y <type>`	Specify the data link type
`-p`	Don't capture in promiscuous mode	`-Z <user>`	Drop privileges from root to **user**

Capture Filter Primitives

`[src|dst] host <host>`	Matches a host as the IP source, destination, or either
`ether [src|dst] host <ehost>`	Matches a host as the Ethernet source, destination, or either
`gateway host <host>`	Matches packets which used **host** as a gateway
`[src|dst] net <network>/<len>`	Matches packets to or from an endpoint residing in **network**
`[tcp|udp] [src|dst] port <port>`	Matches TCP or UDP packets sent to/from **port**
`[tcp|udp] [src|dst] portrange <p1>-<p2>`	Matches TCP or UDP packets to/from a port in the given range
`less <length>`	Matches packets less than or equal to **length**
`greater <length>`	Matches packets greater than or equal to **length**
`(ether|ip|ip6) proto <protocol>`	Matches an Ethernet, IPv4, or IPv6 protocol
`(ether|ip) broadcast`	Matches Ethernet or IPv4 broadcasts
`(ether|ip|ip6) multicast`	Matches Ethernet, IPv4, or IPv6 multicasts
`type (mgt|ctl|data) [subtype <subtype>]`	Matches 802.11 frames based on type and optional subtype
`vlan [<vlan>]`	Matches 802.1Q frames, optionally with a VLAN ID of **vlan**
`mpls [<label>]`	Matches MPLS packets, optionally with a label of **label**
`<expr> <relop> <expr>`	Matches packets by an arbitrary expression

Protocols

`arp`	`ip6`	`slip`
`ether`	`link`	`tcp`
`fddi`	`ppp`	`tr`
`icmp`	`radio`	`udp`
`ip`	`rarp`	`wlan`

Modifiers

`!` or `not`
`&&` or `and`
`||` or `or`

Examples

`udp dst port not 53`	UDP not bound for port 53
`host 10.0.0.1 && host 10.0.0.2`	Traffic between these hosts
`tcp dst port 80 or 8080`	Packets to either TCP port

TCP Flags

`tcp-urg`	`tcp-rst`
`tcp-ack`	`tcp-syn`
`tcp-psh`	`tcp-fin`

ICMP Types

`icmp-echoreply`	`icmp-routeradvert`	`icmp-tstampreply`
`icmp-unreach`	`icmp-routersolicit`	`icmp-ireq`
`icmp-sourcequench`	`icmp-timxceed`	`icmp-ireqreply`
`icmp-redirect`	`icmp-paramprob`	`icmp-maskreq`
`icmp-echo`	`icmp-tstamp`	`icmp-maskreply`

by Jeremy Stretch