

## Contents

- [1 tcpdump](#)
  - ◆ [1.1 Capturing A Specific User's Calls](#)
  - ◆ [1.2 Using Wireshark to Analyze pcap files](#)
- [2 ngrep](#)
- [3 sipgrep](#)
- [4 pcapsipdump](#)
- [5 tshark aka tethereal](#)
- [6 HOMER Sip Capture](#)
- [7 Truncate / Cut a captured pcap file](#)
  - ◆ [7.1 With The Wireshark gui](#)
  - ◆ [7.2 CLI: via editcap](#)
- [8 Analyze a packet capture with SIP TLS on port 5061](#)
- [9 Analyze RTP Quality](#)
- [10 Related](#)

## tcpdump

SRC: <http://www.tcpdump.org/>

Use tcpdump if you want a pcap to open up in wireshark later. Else, use tshark if you want a "text only" view of the SIP traffic without all the headers and extra information.

Examples:

Real-time traffic dump (full packets):

```
tcpdump -nq -s 0 -A -vvv -i eth0 port 5060
```

Dump to file:

```
tcpdump -nq -s 0 -i eth0 -w /tmp/dump.pcap port 5060
```

Save a new time-stamped file approximately once per hour on the specified port

```
tcpdump -nq -s 0 -i eth0 -G3600 -w /tmp/trace/sip-%F--%H-%M-%S.pcap port 5060
```

Daemonize and log 2 ports, rotate every hour.

```
nohup tcpdump -nq -s 0 -i eth0 -G3600 -w /tmp/trace/sip-%F--%H-%M-%S.pcap port 5080 or port 5060
```

Daemonize and log 2 ports, rotate every hour, and place into hierarchical directory structure.

```
#!/bin/bash
TD=`pidof tcpdump`
if [ -n "$TD" ]; then
  kill "$TD"
```

## Packet\_Capture

```
fi
```

```
DIRS="/var/spool/pcap/`/bin/date +%Y`/ /var/spool/pcap/`/bin/date +%Y`/`/bin/date +%m`/ /v  
for DIR in $DIRS  
do  
  if [ ! -d "$DIR" ]; then  
    mkdir "$DIR"  
  fi  
done
```

```
nohup tcpdump -nq -s 0 -i eth0 -G3600 -w '/var/spool/pcap/%Y/%m/%d/%H%M%S.pcap' port 5060 or port
```

This should be run from cron / init services first min of new day.

## Capturing A Specific User's Calls

```
sipstatus profile $profile user $user_id
```

to get the remote ip/and port, then use:

```
tcpdump -i $INTERFACE -s 1500 -A host $IPADDRESS and port $SIPPORT
```

## Using Wireshark to Analyze pcap files

Wireshark has some nice tools for analyzing your packet captures. Jason Garland (IRC: Cherebrum) did two presentations on using Wireshark (and some other tools) for checking your network and your VoIP:

NOTE: The douchebags at viddler went non-free, even for OSS projects. I'm currently working on an alternative with youtube.com. Stay tuned.

The second presentation is here: <http://www.viddler.com/explore/cluecon/videos/33/>

The first presentation got cut short, but it's still good: <http://www.viddler.com/explore/cluecon/videos/8/>

**There was a good howto on the web that disappeared.**

- <http://www.panoramisk.com/151/analyzing-voip-with-wireshark/en/>
  - ◆ <http://web.archive.org/web/20101204200659/http://www.panoramisk.com/151/analyzing-voip-with-wireshark/en/>
  - ◆ [Here's a scrape](#) in case archive.org goes down.

## ngrep

SRC: <http://ngrep.sourceforge.net/>

```
USAGE:usage: ngrep <-hNXViqpbevxlDtTRM> <-IO pcap_dump> <-n num> <-d dev> <-A num>
```

```
tcpdump
```

## Packet\_Capture

```
<-s snaplen> <-S limitlen> <-W normal|byline|single|none> <-c cols>
<-P char> <-F file> <match expression> <bpf filter>
-h is help/usage
-V is version information
-q is be quiet (don't print packet reception hash marks)
-e is show empty packets
-i is ignore case
-v is invert match
-R is don't do privilege revocation logic
-x is print in alternate hexdump format
-X is interpret match expression as hexadecimal
-w is word-regex (expression must match as a word)
-p is don't go into promiscuous mode
-l is make stdout line buffered
-D is replay pcap_dumps with their recorded time intervals
-t is print timestamp every time a packet is matched
-T is print delta timestamp every time a packet is matched
-M is don't do multi-line match (do single-line match instead)
-I is read packet stream from pcap format file pcap_dump
-O is dump matched packets in pcap format to pcap_dump
-n is look at only num packets
-A is dump num packets after a match
-s is set the bpf caplen
-S is set the limitlen on matched packets
-W is set the dump format (normal, byline, single, none)
-c is force the column width to the specified size
-P is set the non-printable display char to what is specified
-F is read the bpf filter from the specified file
-N is show sub protocol number
-d is use specified device instead of the pcap default
```

### EXAMPLES:

```
ngrep -qt -W byline port 5060
ngrep -d any port 5060 -W byline > outfile.txt
ngrep -q '8005551212' -W byline port 5060 #<swk>: only shows packets on 5060 with 8005551212 in
```

For a more in depth tutorial on using ngrep check out this [post](#) by Jonathan Manning. VIM users may be interested in this [syntax highlighter](#).

## sipgrep

SRC: [http://cvs.berlios.de/cgi-bin/viewcvs.cgi/ser/sip\\_router/utils/sipgrep/sipgrep](http://cvs.berlios.de/cgi-bin/viewcvs.cgi/ser/sip_router/utils/sipgrep/sipgrep)

DESC: Perl with number matching in -t or -f 5554443333 style. colored output.

Usage: sipgrep <-h> <-f number> <-t number> <-a> <-l file> <-V> <-p> <-T> <-n|-c>

```
-h      Displays this help message.
-f ARG  Search ARG in From field.
-t ARG  Search ARG in To field.
-a      Search the ARG from '-f' and '-t' parameters in To and From fields.
-l ARG  Debug file name.
-V      Displays the current version.
-p      Port for ngrep.
-T      Parameter for ngrep. Indicating the delta between packet matches.
-c      Allow colors in debug file.
-n      Not allow colors in STDOUT.
```

## Packet\_Capture

```
Example: sipgrep -f 0123456 -t 0654321 -l debug.sip
or
sipgrep -f 0123456 -a -l debug.sip
```

## pcapsipdump

SRC: <http://sourceforge.net/projects/pcapsipdump/>

DESC: pcapsipdump is a tool for dumping SIP sessions (+RTP traffic, if available) to disk in a fashion similar to "tcpdump -w" (format is exactly the same), but one file per sip session (even if there is thousands of concurrent SIP sessions).

```
example: pcapsipdump -i eth0 -d /tmp/ « keep all SIP sessions on tmp folder
```

```
pcapsipdump version 0.1.4-trunk
```

```
Usage: pcapsipdump [-fpU] [-i <interface>] [-r <file>] [-d <working directory>] [-v level]
-f Do not fork or detach from controlling terminal.
-p Do not put the interface into promiscuous mode.
-U Make .pcap files writing 'packet-buffered' - slower method,
  but you can use partially written file anytime, it will be consistent.
-v Set verbosity level (higher is more verbose).
-n Number-filter. Only calls to/from specified number will be recorded
-t T.38-filter. Only calls, containing T.38 payload indicated in SDP will be recorded
```

## tshark aka tethereal

SRC: <http://www.wireshark.org>

DESC: Dump and analyze network traffic.

```
TShark 0.99.4
Dump and analyze network traffic.
See http://www.wireshark.org for more information.
```

```
Copyright 1998-2006 Gerald Combs <gerald@wireshark.org> and contributors.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

```
Usage: tshark [options] ...
```

Capture interface:

```
-i <interface>          name or idx of interface (def: first non-loopback)
-f <capture filter>    packet filter in libpcap filter syntax
-s <snaplen>           packet snapshot length (def: 65535)
-p                     don't capture in promiscuous mode
-y <link type>         link layer type (def: first appropriate)
-D                     print list of interfaces and exit
-L                     print list of link-layer types of iface and exit
```

Capture stop conditions:

```
-c <packet count>      stop after n packets (def: infinite)
-a <autostop cond.> ... duration:NUM - stop after NUM seconds
                       filesize:NUM - stop this file after NUM KB
                       files:NUM - stop after NUM files
```

Capture output:

```
-b <ringbuffer opt.> ... duration:NUM - switch to next file after NUM secs
```

## Packet\_Capture

filesize:NUM - switch to next file after NUM KB  
files:NUM - ringbuffer: replace after NUM files

Input file:  
-r <infile> set the filename to read from (no pipes or stdin!)

Processing:  
-R <read filter> packet filter in Wireshark display filter syntax  
-n disable all name resolutions (def: all enabled)  
-N <name resolve flags> enable specific name resolution(s): "mntC"  
-d <layer\_type>==<selector>,<decode\_as\_protocol> ...  
"Decode As", see the man page for details  
Example: tcp.port==8888,http

Output:  
-w <outfile|-> set the output filename (or '-' for stdout)  
-F <output file type> set the output file type, default is libpcap  
an empty "-F" option will list the file types  
-V add output of packet tree (Packet Details)  
-x add output of hex and ASCII dump (Packet Bytes)  
-T pdml|ps|psml|text output format of text output (def: text)  
-t ad|a|r|d output format of time stamps (def: r: rel. to first)  
-l flush output after each packet  
-q be more quiet on stdout (e.g. when using statistics)  
-X <key>:<value> eXtension options, see the man page for details  
-z <statistics> various statistics, see the man page for details

Miscellaneous:  
-h display this help and exit  
-v display version info and exit  
-o <name>:<value> ... override preference setting

Filter with tshark then separate them per call into different pcap files with pcapsipdump:  
EXAMPLE: 'sip.uri contains "soemname" or rtp or rtcp' -w -|pcapsipdump -

Capture SIP, RTP, ICMP, DNS, RTCP, and T38 traffic in a ring buffer capturing 100 50MB files cont  
EXAMPLE: tshark -i eth0 -o "rtp.heuristic\_rtp: TRUE" -w /tmp/capture.pcap -b filesize:51200 -b fi

Filter on RTCP packets reporting any packet loss or jitter over 30ms  
EXAMPLE: tshark -i eth0 -o "rtp.heuristic\_rtp: TRUE" -R 'rtcp.ssrc.fraction >= 1 or rtcp.ssrc.jit

Filter on SIP and all RTP packets  
EXAMPLE: tshark -S -w capture.pcap -f "(udp port sip) or (udp[1] & 1 != 1 && udp[3] & 1 != 1 && u

Capture all SIP on specified port and switch files every hour  
tshark -nq -i eth0 -b duration:3600 -w /tmp/trace/sip.pcap port 5080

## HOMER Sip Capture

SRC: <http://www.sipcapture.org>

DESC: SIP capturing server with HEP and IP-proto-4 (IPIP) & Monitoring Application with CallFlows, PCAP extraction, powerful search tools, statistics and API. Native HEP [capture agent](#) integrated in FreeSWITCH

## Truncate / Cut a captured pcap file

### With The Wireshark gui

Open the pcap then click "save as". Look at the options - from frame \$x to \$y, the marked ones, from the first marked one to the last marked one, etc. To mark packets, you can right click them in the viewer.

### CLI: via editcap

If you have a large PCAP from any of the above methods and want to share part of it, you can use the "editcap" command line program that comes with wireshark. Read the [full manual](#).

In short, if you want just packets \$x to \$y, use:

```
editcap -r $source_infile $outfile $x-$y
(-r means "only include x-y" otherwise, this command would cut out x-y).
```

If you want to drop some packets, then skip the -r and list the ones to drop, e.g. to drop packet 1000 to 3000:

```
editcap $source_infile $outfile 1000-3000
```

## Analyze a packet capture with SIP TLS on port 5061

Replace 4.2.2.2 with your own IP address.

```
EXAMPLE: wireshark -o "ssl.desegment_ssl_records: TRUE" \
-o "ssl.desegment_ssl_application_data: TRUE" \
-o "ssl.keys_list: 4.2.2.2,5061,sip,/usr/local/freeswitch/conf/ssl/agent.pem" \
-o "ssl.debug_file: /tmp/wireshark.log" \
-i eth0 -f "tcp port 5061"
```

```
EXAMPLE: tshark -o "ssl.desegment_ssl_records: TRUE" \
-o "ssl.desegment_ssl_application_data: TRUE" \
-o "ssl.keys_list: 4.2.2.2,5061,sip,/opt/freeswitch/conf/ssl/agent.pem" \
-o "ssl.debug_file:/tmp/tshark.log" \
-i eth0 \
-f "tcp port 5061"
```

Analyze RTP events

## Packet\_Capture

```
tshark -o "rtp.heuristic_rtp: TRUE" -R rtpevent  
or with tethereal:  
tethereal -o "rtp.heuristic_rtp: TRUE" -R rtpevent
```

## Analyze RTP Quality

```
sudo tshark -q -f 'udp portrange 16384-32768' -o rtp.heuristic_rtp:TRUE -z rtp,streams
```

If you're doing long-term captures, you may want to get a bit more paranoid about security:

```
sudo setuid 4755 /usr/bin/dumpcap  
dumpcap -f 'udp portrange 16384-32768' -i eth0 -w /tmp/qos.pcap  
tshark -qr /tmp/qos.pcap -o rtp.heuristic_rtp:TRUE -z rtp,streams
```

## Related

- [RTP Issues](#)
- [Jitterbuffer](#)
- [Jason Garland's LAN debugging video](#)