Network Analysis and Forensics Cheatsheet

Introduction

While analysing data supplied in the NCAC, I compiled a set of tools that are helpful in parsing artifacts and searching for indicators. Here, I document how these tools are used at a basic level. I also include commands that have been useful to be which you may adopt as well.

Built-in command line tools

Look for all cars of the "Corolla" model

```
cat data.txt | grep Toyota | less
```

Get the first column of every row in a file

```
cat data.txt | awk '{print $1}' | less
```

Get the first two columns of every row, separated by a space

```
cat data.txt | awk '{print $1 " " $2}' | less
```

Sort the car type in order of occurrence (histogram)

```
cat data.txt | awk '{print $1 " " $2}' | sort  | uniq -c | sort -rn
```

Recursively search for the word "estate" in the current directory

```
grep -irF "Estate" . | less
```

Use hexdump to view a binary file (don't do this for evt files)

```
hexdump -Cv file.exe | less
```

Network tools

Wireshark

Find all network traffic associated with an IP address.

```
ip.addr == 10.0.0.1
```

Set a conversation filter between the two defined IP addresses

```
ip.addr==10.0.0.1 && ip.addr==10.0.0.2
```

Set a filter to display all http and dns. (You can also use smtp, or other protocols by name)

```
http or dns
```

Search for a url that someone has browsed. You can replace ".com" with your desired domain or keep it to look for all domains with that tld.

```
http.host contains ".com"
```

Set a filter for any TCP packet with 25 as a source or dest port. Good for filtering for email.

```
tcp.port == 25
```

Match HTTP requests where the last characters in the uri are the characters "gl=se"

```
http.request.uri matches "gl=se$"
```

Search for the string "helloworld" in any tcp packet. This is a very broad search.

```
tcp contains "helloworld"
```

Additional Resources
- https://packetlife.net/media/library/13/Wireshark*Display*Filters.pdf - https://wiki.wireshark.org/CaptureFilters - https://networksecuritytools.com/list-wireshark-display-filters/

Tshark

Tshark is the command line version of Wireshark. I use it to parse through files a little bit faster. It is also useful for carving out parts of a pcap file for further analysis (useful for large files).

Basic usage: Use the `-r` flag to specficy the network pcap file. Use the `-2 -R` followed by quotes, you can use any of the tcp filters from Wireshark above.

Search for all packets containing the string "helloworld"

```
tshark -r myfile.pcap -2 -R "tcp contains helloword" | less
```

Follow all streams in a file. Careful... this can be noisy.

```
tshark -r myfile.pcap -q -z conv,tcp
```

Parse out all ftp data and write to a file.

```
tshark -r myfile.pcap -2 -R “ftp” -w output.pcap
```

Tcpdump

Tcpdump uses the burkeley packet filters (different from Wireshark and t-shark).

Get all ftp traffic.

```
tcpdump -r myfile.pcap “tcp (port 20 or 21)”
```

Get all traffic involving a host and write it to a file.

```
tcpdump -r myfile.pcap “host x.x.x.x” -w output.pcap
```

Additonal resources * https://yaleman.org/2013/09/11/berkeley-packet-filter-bpf-syntax/

Ngrep

Ngrep slices through pcap files very efficiently, but the results are spit out in text.

Find all packets going over port 25

```
ngrep -q -I  test.pcap -w port 25 | less
```

Packets sent or received by 209.242.181.34

```
ngrep -q -I  test.pcap -w host 209.242.181.34 | less
```

Packets sent by 209.242.181.34

```
ngrep -q -I  test.pcap -w src 209.242.181.34 | less
```

Packets sent by 74.125.226.196 to 209.242.181.34

```
ngrep -q -I  test.pcap -w src 74.125.226.196 and dst 209.242.181.34 | less
```

Packets sent to (received by) 209.242.181.34

```
ngrep -q -I  test.pcap -w dst 209.242.181.34 | less
```

Packets containing the word "Google"

```
ngrep -q -I test.pcap 'Google'   | less
```

Email packets involving "foo@bar.com"

```
ngrep -q -I  test.pcap 'foo@bar.com' port 25 | less
```

Same but view in one line (-W single and less -S)

```
ngrep -q -W single -I  test.pcap 'foo@bar.com' port 25 | less -S
```

Same but force column width to make reading easier (-c 90) ngrep -q -c 90 -I test.pcap 'foo@bar.com' port 25 l less -s

Same but print the timestamp (-t)

```
ngrep -q -t  -c 90 -I  test.pcap 'foo@bar.com' port 25 | less -s
```

Forensics tools

Volatility

The first thing you need to do is to identify the profile

```
vol -f memdump.mem imageinfo
```

Identify connections to IP addresses made by the machine

```
vol -f memdump.mem --profile=WinXPSP2x86 connections
```

Print a list of process in the image (I also like pstree)

```
vol -f memdump.mem --profile=Win7SP0x64 pslist
```

Carve out the master file table and write to to a file

```
vol -f memdump.mem --profile=Win7SP0x64 mftparser >> mft.txt
```

View all the sites browed by the user. can be further grepped

```
vol -f memdump.mem --profile=Win7SP0x64 iehistory | grep Location | less
```

Convert a hybernation file to a windows image for analysis

```
vol imagecopy -f hiberfil.sys -O win7.img
```

** Use the help command for the full list of things you can get from volatility

Sleuthkit

Step 1:

```
$ img_stat file.raw

IMAGE FILE INFORMATION
--------------------------------------------
Image Type: raw

Size in bytes: 1073741824
Sector size: 512
```

Step 2: Get the layout of the disk image

```
$ mmls file.raw

DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

      Slot      Start        End          Length       Description
000:  Meta      0000000000   0000000000   0000000001   Primary Table (#0)
001:  -------   0000000000   0000002047   0000002048   Unallocated
002:  000:000   0000002048   0031459327   0031457280   NTFS / exFAT (0x07)
003:  -------   0031459328   0031463423   0000004096   Unallocated
```

Step 3: List out contents on disk

```
$ fls -o 0000002048 file.raw

r/r 4-128-4: $AttrDef
r/r 8-128-2: $BadClus
r/r 8-128-1: $BadClus:$Bad
r/r 6-128-1: $Bitmap
r/r 7-128-1: $Boot
d/d 11-144-4: $Extend
r/r 2-128-1: $LogFile
r/r 0-128-1: $MFT
r/r 1-128-1: $MFTMirr
r/r 9-128-16: $Secure:$SDS
r/r 9-144-17: $Secure:$SDH
r/r 9-144-18: $Secure:$SII
r/r 10-128-1: $UpCase
r/r 3-128-3: $Volume
r/r 35-128-1: AUTOEXEC.BAT
r/r 36-128-1: boot.ini
r/r 37-128-1: CONFIG.SYS
d/d 38-144-6: Documents and Settings
r/r 2521-128-1: IO.SYS
r/r 2522-128-1: MSDOS.SYS
d/d 11983-144-1: MSOCache
r/r 2523-128-3: NTDETECT.COM
r/r 2524-128-3: ntldr
r/r 19393-128-1: pagefile.sys
d/d 2525-144-6: Program Files
d/d 25626-144-1: QUARANTINE
d/d 25043-144-1: RECYCLER
d/d 3435-144-6: System Volume Information
d/d 3438-144-6: WINDOWS
d/- * 0: WINDOWS
V/V 32480: $OrphanFiles
```

Step 4: Output the contents of a file based on its inode number.

```
$ icat -o  0000002048 file.raw 32451-128-1  | less
```