Stepping up your packet analysis game...

...by leveraging the Wireshark CLI tools!

SYN-bit
deep traffic analysis

Sake Blok
Relational therapist for computer systems
sake.blok@SYN-bit.nl

Hands-On Wireshark CLI tools

- Get the files from:
  http://www.SYN-bit.nl/files/sf23-cli.zip

- Goal:
  Exploration of the Wireshark CLI tools tshark, editcap, mergecap and capinfos

Agenda

- Introductions
- Why use CLI tools?
  - ... and how?
- Wireshark CLI tools
  - Hands-On Wireshark CLI tools
- Useful shell commands
- Some Scripting Examples
  - Hands-On Scripting
- Q&A

$ whoami

EURONET INTERNET

ABN·AMRO

ion ip

SYN-bit
deep traffic analysis

Why use CLI tools?
… and how?

```
5464 16:44:57.225   0.014 10.0.0.108     10.0.0.10      SIP          605 Status: 200 OK |
5469 16:44:57.271   0.045 10.0.0.108     37.235.81.212  RTCP         130 Sender Report   Source description   Goodbye
5470 16:44:57.277   0.005 37.235.81.212  10.0.0.108     ICMP         158 Destination unreachable (Port unreachable)
5471 16:44:57.402   0.124 10.0.0.10      10.0.0.108     TCP           66 59519 → 5060 [ACK] Seq=7008 Ack=4384 Win=8760 Len=0
5472 16:44:57.459   0.057 10.0.0.108     10.0.0.10      TCP           74 48191 → 80 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK
5473 16:44:57.459   0.000 10.0.0.10      10.0.0.108     TCP           78 80 → 48191 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS
5474 16:44:57.460   0.000 10.0.0.108     10.0.0.10      TCP           66 48191 → 80 [ACK] Seq=1 Ack=1 Win=14600 Len=0 TSval=4
5475 16:44:57.486   0.026 10.0.0.108     10.0.0.10      HTTP         226 GET /aastraSipTerm.xml?requestType=disconnected HTTP
5476 16:44[illegible] 10.0.0.10      10.0.0.108     HTTP         64[illegible] HTTP/1.1 401 Unauthorized (text/ht[illegible]
5477 16:44:[illegible] 10.0.0.108 [illegible] TCP  [illegible] 4819[illegible] 80 [ACK] Seq=161 [illegible]k=583 Win=[illegible]600 Len=0 TSv
5478 16:44:5[illegible] 0.0[illegible] TCP  [illegible] 4819[illegible] → 8[illegible] Win=14600 Len=
5479 16:44:5[illegible] 0.0[illegible] TCP  [illegible] 80 → [illegible]8191 Len=0 TSva
5480 16:44:57.530   0.002 10.0.[illegible]08  10.0.0.10      HTTP         570 GET /aastraSipTerm.xml?requestType=disconnected HTTP
5481 16:44:57.533   0.002 10.0[illegible].10  10.0.0.108     HTTP/XML     365 HTTP/1.1 200 OK
5482 16:44:57.533   0.000 10.0.0.108     10.0.0.10      TCP           66 48191 → 80 [ACK] Seq=665 Ack=881 Win=14600 Len=0 TSv
5483 16:45:00.935   3.402 LiteonTe_dc:1c… Broadcast     ARP           60 Who has 10.0.0.[illegible]? Tell 10.0.0.100
5484 16:45:01.560   0.624 LiteonTe_dc:1c… Broadcast     ARP           60 Who has 10.0.0.22[illegible] Tell 10.0.0.100
5485 16:45:02.589   1.029 LiteonTe_dc:1c… Broa[illegible] [illegible]? Tell 10.0.0.100
5486 16:45:06.081   3.491 10.0[illegible]108 [illegible] [illegible]ITE [illegible]ip:0630302817@10.0.0.10:5060;user=
5487 16:45:06.083   0.002 10.0[illegible]10 [illegible] [illegible] 330 Status: 100 Trying |
5488 16:45:06.106   0.022 10.0.0.10      185.114.236.101 SIP/SDP    1053 Request: INVITE sip:0630302817@sip.bcom.nl |
5489 16:45:06.113   0.007 185.114.236.101 10.0.0.10     SIP          389 Status: 100 Trying. Please Hold... |
5490 16:45:06.117   0.004 185.114.236.101 10.0.0.10     SIP          528 Status: 407 Proxy Authentication Required |
5491 16:45:06.118   0.001 10.0.0.108     10.0.0.10      TCP           66 44011 → 5060 [ACK] Seq=10194 Ack=10097 Win=14600 Len
5492 16:45:06.120   0.001 10.0.0.10      185.114.236.101 SIP         396 Request: ACK sip:0630302817@sip.bcom.nl |
5493 16:45:06.124   0.003 10.0.0.10      185.114.236.101 SIP/SDP    1257 Request: INVITE sip:0630302817@sip.bcom.nl |
5494 16:45:06.131   0.007 185.114.236.101 10.0.0.10     SIP          389 Status: 100 Trying. Please Hold... |
5495 16:45:07.447   1.316 10.0.0.10      185.114.236.101 SIP         470 Request: OPTIONS sip:sip.bcom.nl |
5496 16:45:07.450   0.002 10.0.0.10      185.114.236.202 SIP         470 Request: OPTIONS sip:sip.bcom.nl |
```

Why use the CLI tools?

- When GUI is not available (shell access)
- Quick and Easy Analysis
- Postprocessing results
  - GUI is powerful & interactive, but fixed functionality
  - CLI combined with other tooling is very flexible
- Automation

CLI not only when GUI is unavailable

How?

- What information do I need?
  - visualize your output
- What (raw) data sources do I have?
  - Know the output formats of your data sources
- What tools are available?
  - What can they do, browse through manpages for unknown options

Practice & Experiment & be creative :-)

Wireshark CLI tools

```
5464  16:44:57.225  0.014  10.0.0.108       10.0.0.10        SIP       605  Status: 200 OK |
5469  16:44:57.271  0.045  10.0.0.108       37.235.81.212    RTCP      130  Sender Report   Source description   Goodbye
5470  16:44:57.277  0.005  37.235.81.212    10.0.0.108       ICMP      158  Destination unreachable (Port unreachable)
5471  16:44:57.402  0.124  10.0.0.10        10.0.0.108       TCP        66  59519 → 5060 [ACK] Seq=7008 Ack=4384 Win=8760 Len=0
5472  16:44:57.459  0.057  10.0.0.108       10.0.0.10        TCP        74  48191 → 80 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK
5473  16:44:57.459  0.000  10.0.0.10        10.0.0.108       TCP        78  80 → 48191 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS
5474  16:44:57.460  0.000  10.0.0.108       10.0.0.10        TCP        66  48191 → 80 [ACK] Seq=1 Ack=1 Win=14600 Len=0 TSval=4
5475  16:44:57.486  0.026  10.0.0.108       10.0.0.10        HTTP      226  GET /aastraSipTerm.xml?requestType=disconnected HTTP
5476  16:44:57.487  0.001  10.0.0.10        10.0.0.108       HTTP      647  HTTP/1.1 401 Unauthorized  (text/html)
5477  16:44:57.487  0.000  10.0.0.108       10.0.0.10        TCP        66  48191 → 80 [ACK] Seq=161 Ack=582 Win=14600 Len=0 TSv
5478  16:44:57.527  0.040  10.0.0.108       10.0.0.10        TCP        66  48190 → 80 [FIN, ACK] Seq=665 Ack=882 Win=14600 Len=
5479  16:44:57.527  0.000  10.0.0.10        10.0.0.108       TCP        66  80 → 48190 [ACK] Seq=882 Ack=666 Win=8191 Len=0 TSva
5480  16:4[illegible]  [illegible]002  10.0.0.108  [illegible]0.0.0.10  HTT[illegible]  [illegible]ET /a[illegible]traSi[illegible]Term.xml?requestTyp[illegible]disconnected HTTP
5481  16:4[illegible]  365  [illegible]TTP/1[illegible]  200
5482  16:4[illegible]  TCP  66  [illegible]8191 [illegible]80 [A[illegible]  Len=0 TSv
5483  16:45[illegible]  [illegible]40  Li[illegible]  [illegible]ro[illegible]ca[illegible]  ARP  [illegible]0  [illegible]ha[illegible]10.0.[illegible]
5484  16:45:01.560  0.624  LiteonTe_dc:1c…  Broadcast        ARP        60  Who has 10.0.0.222? Tell 10.0.0.100
5485  16:45:02.589  1.029  LiteonTe_dc:1c…  Broadcast        ARP        60  Who has 10.0.0.222? Tell 10.0.0.100
5486  16:45:06.081  3.491  10.0.0.108       10.0.0.10        SIP/SDP  1261  Request: INVITE sip:0630302817@10.0.0.10:5060;user=
5487  16:45:06.083  0.002  10.0.0.10        10.0.0.108       SIP       330  Status: 100 Trying |
5488  16:45:06.106  0.022  10.0.0.10        185.114.236.101  SIP/SDP  1053  Request: INVITE sip:0630302817@sip.bcom.nl |
5489  16:45:06.113  0.007  185.114.236.101  10.0.0.10        SIP       389  Status: 100 Trying. Please Hold... |
5490  16:45:06.117  0.004  185.114.236.101  10.0.0.10        SIP       528  Status: 407 Proxy Authentication Required |
5491  16:45:06.118  0.001  10.0.0.108       10.0.0.10        TCP        66  44011 → 5060 [ACK] Seq=10194 Ack=10097 Win=14600 Len
5492  16:45:06.120  0.001  10.0.0.10        185.114.236.101  SIP       396  Request: ACK sip:0630302817@sip.bcom.nl |
5493  16:45:06.124  0.003  10.0.0.10        185.114.236.101  SIP/SDP  1257  Request: INVITE sip:0630302817@sip.bcom.nl |
5494  16:45:06.131  0.007  185.114.236.101  10.0.0.10        SIP       389  Status: 100 Trying. Please Hold... |
5495  16:45:07.447  1.316  10.0.0.10        185.114.236.101  SIP       470  Request: OPTIONS sip:sip.bcom.nl |
5496  16:45:07.450  0.002  10.0.0.10        185.114.236.202  SIP       470  Request: OPTIONS sip:sip.bcom.nl |
```

Wireshark CLI tools

- tshark
- dumpcap
- capinfos
- editcap
- mergecap
- rawshark
  (not covered)

tshark (1)

- CLI version of wireshark
- Similar to tcpdump, but statefull / reassembly
  .... and MANY full protocol decodes
- uses dumpcap as capture engine
- standard options: -D, -i, -c, -n, -l, -f, -R, -s, -w, -r
- name resolving (-n)
- time stamps (-t<format>)
- decode as (-d tcp.port==8080,http)
- preferences (-o <pref>:<value>)

tshark (2)

- output formats (-V or -T <format>)
  - default: summary, uses column prefs
  - Verbose (-V), hex dump (-x)
  - PDML (-T pdml)
  - fields (-T fields -E <sep> -e <field1> -e <field2> …)

- statistics (-z …)
  - protocol hierarchy (-qz io,phs)
  - endpoints (-qz endpoints,eth , -qz endpoints,ip)
  - conversations (-qz conv,eth , -qz conv,tcp)
  - i/o statistics (-qz io,stat,10,ip,icmp,udp,tcp)

dumpcap

- used by (wire|t)shark
  ... for privilege separation
- can be used separately
- options similar to tshark
- fast! only network->disk
- stateless! so traces can run forever
- ring buffer feature extremely useful:

```
dumpcap -i 5 -s0 -b filesize:16384 -files:1024 -w ring.pcapng
```

capinfos

- display summary of a tracefile
- all info vs specific info

```
$ capinfos example.pcap
File name: example.pcap
File type: Wireshark/tcpdump/... - libpcap
File encapsulation: Ethernet
Number of packets: 3973
File size: 1431813 bytes
Data size: 1368221 bytes
Capture duration: 1299.436650 seconds
Start time: Thu Jan 17 11:37:16 2008
End time: Thu Jan 17 11:58:55 2008
Data rate: 1052.93 bytes/s
Data rate: 8423.47 bits/s
Average packet size: 344.38 bytes
```

```
$ capinfos -ae sharkfest-*.pcap
File name: example.pcap
Start time: Thu Jan 17 11:37:16 2008
End time: Thu Jan 17 11:58:55 2008

File name: sharkfest-2.pcap
Start time: Thu Jan 17 11:39:27 2008
End time: Thu Jan 17 12:02:52 2008
```

editcap (1)

- used to **select** packets in a capture file

  - select frame ranges or time ranges

```
editcap -r example.pcap tmp.pcap 1-1000 2001-3000
editcap -A "2008-01-17 11:40:00" \
      -B "2008-01-17 11:49:59" example.pcap tmp.pcap
```

  - split file in chunks

```
editcap -c 1000  example.pcap tmp.pcap
editcap -i 60 example.pcap tmp.pcap
```

  - remove duplicate packets

```
editcap -d example.pcap tmp.pcap
```

editcap (2)

- used to **change** (packets in) a capture file
  - change snaplen

```
editcap -s 96  example.pcap tmp.pcap
```

  - change timetamps

```
editcap -t -3600 example.pcap tmp.pcap
```

  - change link layer type

```
editcap -T user0 example.pcap tmp.pcap
```

  - change file type

```
editcap -F ngsniffer example.pcap tmp.pcap
```

mergecap

- used to merge capture files:
  - based on timestamps

    ```
    mergecap -w out.pcap in-1.pcap in-2.pcap
    ```

  - or just append each file

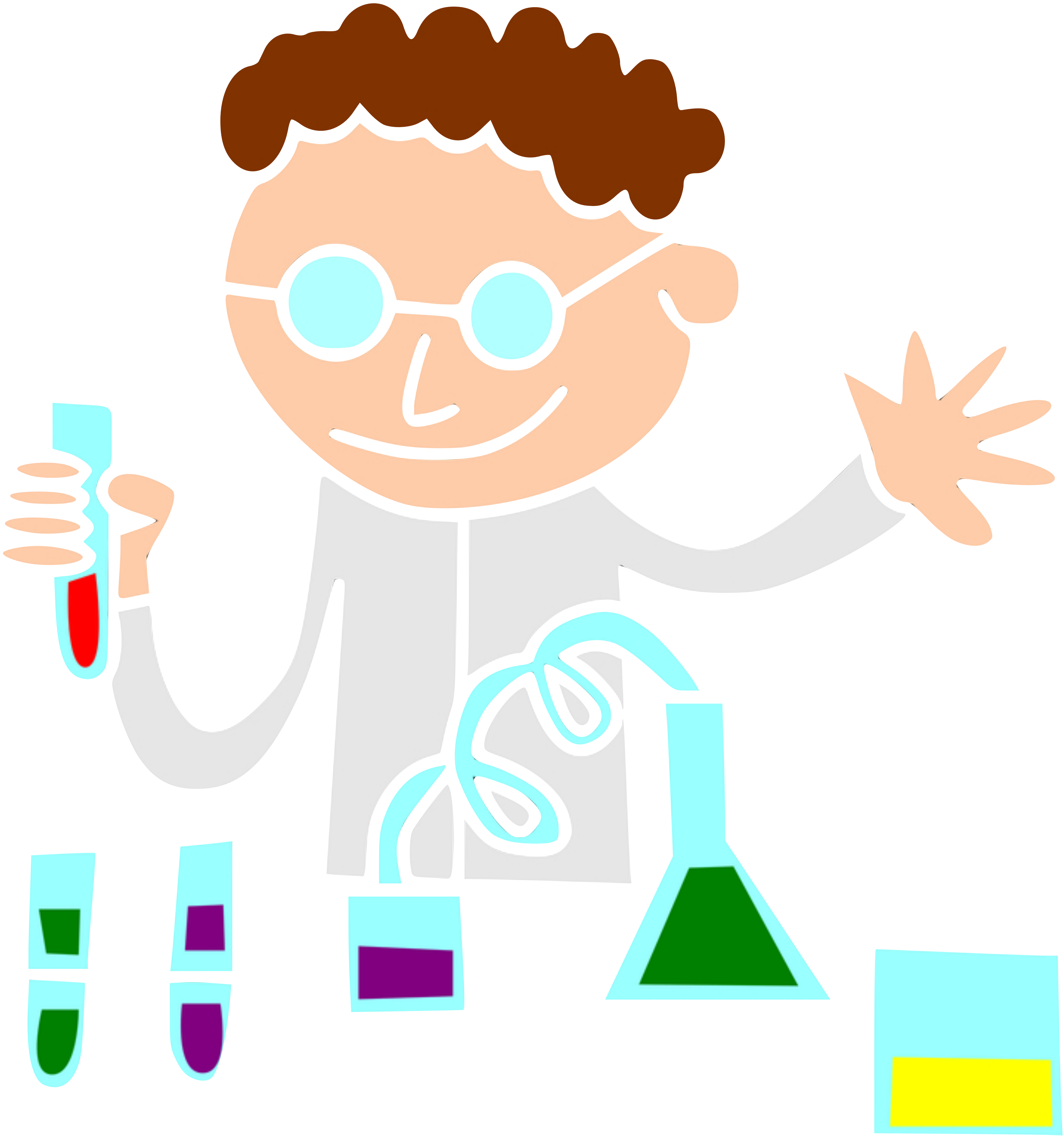
    ```
    mergecap -a -w out.pcap in-1.pcap in-2.pcap
    ```

Wireshark CLI tools

- Use "<command> -h" for options
  ... check once-in-a-while for new features

- Read the man-pages for in-depth guidance
  (see: http://www.wireshark.org/docs/man-pages/)

Hands-On Wireshark CLI tools



Useful shell commands

Useful shell commands

- bash internals:
  |, >, for … do … done, `<command>`
- cut
- sort
- uniq
- tr
- sed
- awk
- scripting (sh/python/…)

|, >, for … do … done

- Command piping with '|'

```
ls -1t | head
```

- Output redirection with '>'

```
ls -1t | head > 10-newest-files.txt
```

- Looping with for … do … done

```
for word in 'one' 'two' 'three'; do echo $word;
done
```

`<command>` , variable assignments

- Command evaluation with backtics (``)

```
for file in `ls -1t | head`
do
   echo $file
   head -1 $file
   echo ""
done > firstlines.txt
```

- Variable assignments

```
backupfile=`echo ${file}.bak`
```

cut

- By character position (-c <range>)

```
cut -c1-10 /etc/passwd
```

- By field (-f<index> [-d '<delimiter>']

```
cut -d ':' -f1 /etc/passwd
```

sort

- General alphabetic sort (no option)

```
sort names.txt
```

- Reverse sorting (-r)

```
sort -r names.txt
```

- Numerical (-n)

```
sort -n numbers.txt
```

- Or combined: `du -ks * | sort -rn | head`

uniq

- De-duplication (no option)

```
sort names.txt | uniq
```

- Show only 'doubles' (-d)

```
sort names.txt | uniq -d
```

- Count occurrences (-c)

```
sort names.txt | uniq -c
```

tr

- Translate a character(set)

```
echo "one two" | tr " " "_"
echo "code 217" | tr "[0-9]" "[A-J]"
echo "What is a house?" | tr "aeiou" "eioua"
```

- Delete a character(set)

```
echo "no more spaces" | tr -d " "
echo "no more vowels" | tr -d "aeiou"
cat dosfile.txt | tr –d "\015" > unixfile.txt
```

sed

- stream editor
- Very powerful 'editing language'
- Some simple examples:
  - deleting text:

```
sed -e 's/<deleteme>//'
```

  - replacing text:

```
sed -e 's/<replaceme>/<withthis>/'
```

  - extracting text:

```
sed -e 's/^.*\(<keepme>\).*\(<andme>\).*$/\1 \2/‘
```

awk

- pattern scanning and processing language
- Also a very powerful language
- Some simple examples:

```
netstat -an | \
awk '$1~"tcp" {print $4}' | \
sort | uniq –c

… | awk '{printf(" %s tcp.port==%s",sep,$1);sep="||"}'
```

scripting

- parsing output when command piping is not enough
- automate execution of tshark/dumpcap/mergecap etc
- use your own favorite language (sh/perl/python/etc)

do anything you want :-)

Some Scripting Examples

```
5464 16:44:57.225  0.014 10.0.0.108      10.0.0.10       SIP       605 Status: 200 OK |
5469 16:44:57.271  0.045 10.0.0.108      37.235.81.212   RTCP      130 Sender Report   Source description   Goodbye
5470 16:44:57.277  0.005 37.235.81.212   10.0.0.108      ICMP      158 Destination unreachable (Port unreachable)
5471 16:44:57.402  0.124 10.0.0.10       10.0.0.108      TCP        66 59519 → 5060 [ACK] Seq=7008 Ack=4384 Win=8760 Len=0
5472 16:44:57.459  0.057 10.0.0.108      10.0.0.10       TCP        74 48191 → 80 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK
5473 16:44:57.459  0.000 10.0.0.10       10.0.0.108      TCP        78 80 → 48191 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS
5474 16:44:57.460  0.000 10.0.0.108      10.0.0.10       TCP        66 48191 → 80 [ACK] Seq=1 Ack=1 Win=14600 Len=0 TSval=4
5475 16:44:57.486  0.026 10.0.0.108      10.0.0.10       HTTP      226 GET /aastraSipTerm.xml?requestType=disconnected HTTP
5476 16:44:57.487  [illegible] 10.0.0.10       10.0.0.108      [illegible] [illegible] HTTP/1.1 [illegible] Unauthorized  (text/html)
5477 16:44:57.487  [illegible]
5478 16:44:57.527  [illegible]
5479 16:44:57.527  [illegible]
5480 16:44:57.530  [illegible]
5481 16:44:57.533  0.002 10.0.0.10       10.0.0.108      HTTP/XML  [illegible] HTTP/1.1 200 OK
5482 16:44:57.533  0.000 10.0.0.108      10.0.0.10       TCP        66 48191 → 80 [ACK] Seq=665 Ack=881 Win=14600 Len=0 TSv
5483 16:45:00.935  3.402 LiteonTe_d[illegible] Broadcast  ARP  [illegible] Who has 10.0.0.222? Tell 10.0.0.100
5484 16:45:01.560  0.624 LiteonTe_d[illegible] Broadcast  [illegible] Who has [illegible].0.0.222? Tell 10.0.0.100
5485 16:45:02.589  1.029 LiteonTe_d[illegible] [illegible] [illegible].0.222? Tell 10.0.0.100
5486 16:45:06.081  3.491 10.0.0.108      [illegible] [illegible] sip:0630302817@10.0.0.10:5060;user=
5487 16:45:06.083  0.002 10.0.0.10       [illegible] Trying |
5488 16:45:06.106  0.022 10.0.0.10       185.114.236.101 SIP/SDP  1053 Request: INVITE sip:0630302817@sip.bcom.nl |
5489 16:45:06.113  0.007 185.114.236.101 10.0.0.10       SIP       389 Status: 100 Trying. Please Hold... |
5490 16:45:06.117  0.004 185.114.236.101 10.0.0.10       SIP       528 Status: 407 Proxy Authentication Required |
5491 16:45:06.118  0.001 10.0.0.108      10.0.0.10       TCP        66 44011 → 5060 [ACK] Seq=10194 Ack=10097 Win=14600 Len
5492 16:45:06.120  0.001 10.0.0.10       185.114.236.101 SIP       396 Request: ACK sip:0630302817@sip.bcom.nl |
5493 16:45:06.124  0.003 10.0.0.10       185.114.236.101 SIP/SDP  1257 Request: INVITE sip:0630302817@sip.bcom.nl |
5494 16:45:06.131  0.007 185.114.236.101 10.0.0.10       SIP       389 Status: 100 Trying. Please Hold... |
5495 16:45:07.447  1.316 10.0.0.10       185.114.236.101 SIP       470 Request: OPTIONS sip:sip.bcom.nl |
5496 16:45:07.450  0.002 10.0.0.10       185.114.236.202 SIP       470 Request: OPTIONS sip:sip.bcom.nl |
```

Some Examples

- Using command piping
  - Counting http response codes
  - Top 10 URL's
  - All TCP sessions which contain session-cookie XXXX

- Using scripting
  - All sessions for user XXXX (shell script)

Example 1:
Counting http response codes (1)

- Problem
  - I need an overview of http response codes

- Output
  - table with http response codes & counts

- Input
  - Capture file with http traffic

Example 1:
Counting http response codes (2)

- Steps to take
  - print only http response code
  - count
  - make (sorted) table

Example 1:
Counting http response codes (3)

- Command:

```
tshark -r example.pcap -R
http.response
       -T fields -e http.response.code
|\
sort | uniq -c
```

- New tricks learned:

  -T fields -e <field>

  | sort | uniq -c

Example 2:
Top 10 requested URL's (1)

- Problem
  - I need a list of all URL's that have been visited

- Output
  - Sorted list with requested URL's and count

- Input
  - Capture file with http traffic

Example 2:
Top 10 requested URL's (2)

- Steps
  - Print http.host and http.request.uri
  - Strip everything after “?”
  - Combine host + uri and format into normal URL
  - count url’s
  - make top 10

Example 2:
Top 10 requested URL's (3)

- Command:

```
tshark -r example.pcap -R http.request \
  -T fields -e http.host -e http.request.uri |\
sed -e 's/?.*$//' |\
sed -e 's#^\(.*\)\t\(.*\)$#http://\1\2#' |\
sort | uniq -c | sort -rn | head
```

- New tricks learned:

  remove unnecessary info : `sed -e 's/?.*$//'`

  transform : `sed -e 's#^\(.*\)\t\(.*\)$#http://\1\2#'`

  top10 : `| sort | uniq -c | sort -rn | head`

Example 3:
All sessions with cookie XXXX (1)

- Problem
  - I know in which “session” a problem exists, but I need all data from that session to work it out

- Output
  - New capture file with whole tcp sessions that contain cookie PHPSESSID=c0bb9d04cebbc765bc9bc366f663fcaf

- Input
  - Capture file with http traffic

Example 3:
All sessions with cookie XXXX (2)

- Steps
  - select packets that contain the cookie
  - print the port numbers
  - create new filter based on port numbers
  - use filter to extract tcp sessions
  - save packets to a new capture file

Example 3:
All sessions with cookie XXXX (3)

- Command:

```
tshark -r example.pcap -w cookie.pcap \
  -R `tshark -r example.pcap -T fields -e tcp.srcport
            -R "http.request and http.cookie contains \

"PHPSESSID=c0bb9d04cebbc765bc9bc366f663fcaf\"" |\
           awk '{printf(“%stcp.port==%s",sep,1);sep="||"}‘
`
```

- New tricks learned:

```
tshark -R `<other command that generated filter>`
awk '{printf("%stcp.port==%s",sep,$1);sep="||"}'
```

Example 4:
All sessions for user XXXX (1)

- Problem
  - A particular user has multiple sessions and I need to see all sessions from that user
- Output
  - New capture file with all data for user xxxx
- Input
  - Capture file with http data

Example 4:
All sessions for user XXXX (2)

- Steps
  - print all session cookies for user XXXX
  - create new capture file per session cookie (see example 3)
  - merge files to new output file

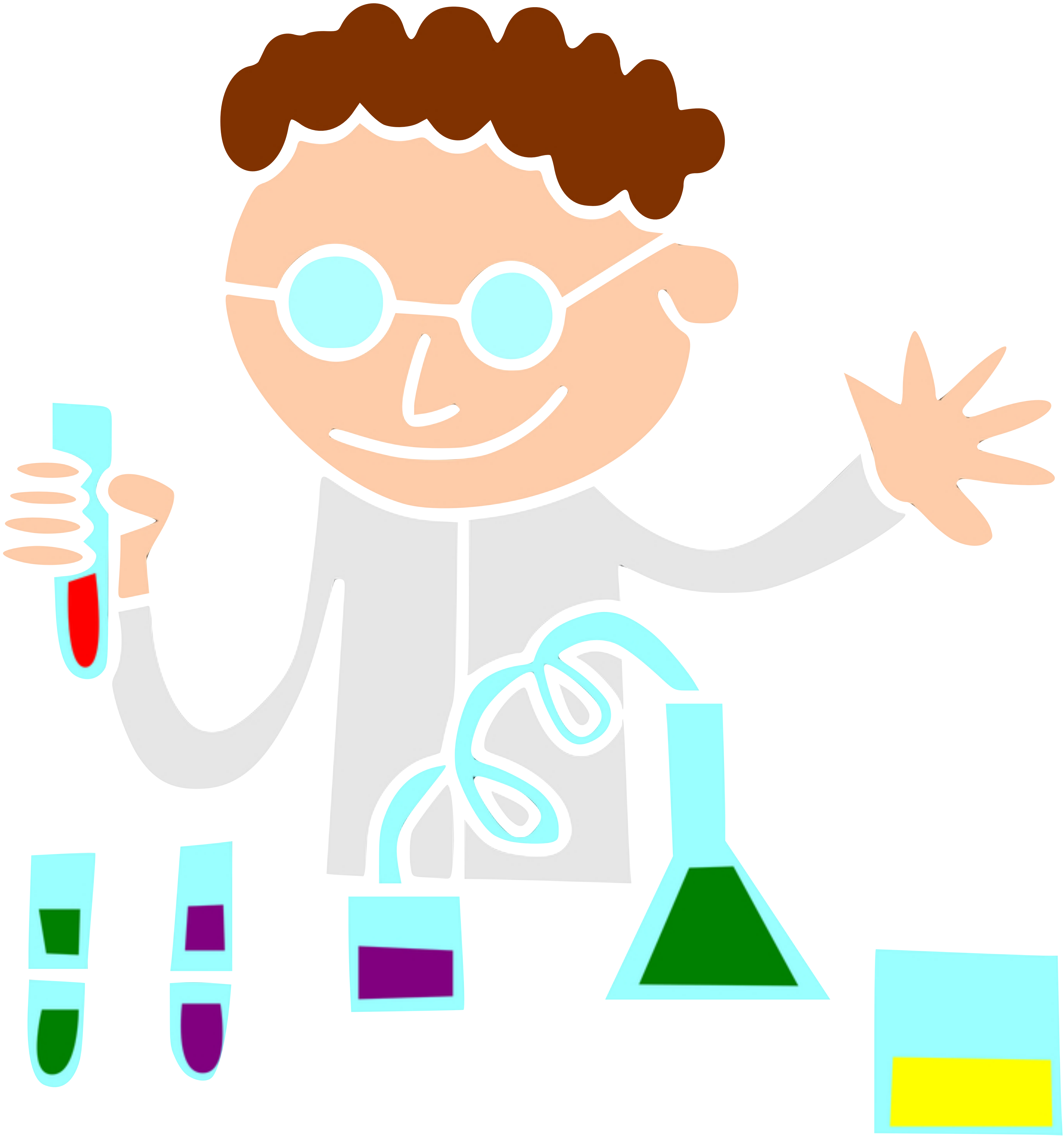
Example 4:
All sessions for user XXXX (3)

```
#!/bin/bash

file=$1
user=$2

for cookie in `tshark -r $file -R "http.request and http contains $user" -T
fields -e http.cookie | cut -d ' ' -f2`
do
   tmpfile="tmp_`echo $cookie | cut -d '=' -f 2`.pcap"
   echo "Processing session cookie $cookie to $tmpfile"

   tshark -r $file -w $tmpfile -R `tshark -r $file -T fields -e tcp.srcport \
      -R "http.request and http.cookie contains \"$cookie\"" | \
      awk '{printf("%stcp.port==%s",sep,$1);sep="||"}'`
done

mergecap -w $user.pcap tmp_*.pcap
rm tmp_*.pcap
```

Example 4:
All sessions for user XXXX (4)

- New tricks learned:
  - `for … do … done`
  - ``<var>=`echo … | …` ``
  - `cut -d <FS> -f <x>`
  - `mergecap -w <outfile> <infile1> <infile2>`
  - …

Hands-On Scripting



Hands-On Scripting

- Get the files from:
  http://www.SYN-bit.nl/files/sf23-cli.zip
- bash required
  (use Windows+cygwin, linux, MacOS/X, etc)
- Goal:
  Explore piping tshark output through other commands to create custom output formats

Takeaways

- Great fun analysing a CVE
- Take precautions to do it safe!
- Make packet captures…. DUH!
- Make use of:
  - Wireshark profiles
  - Display filter buttons
  - Packet coloring

Summary

- tshark, dumpcap, capinfos, editcap, mergecap
- tshark+scripting can complement GUI
- use little building blocks and combine them

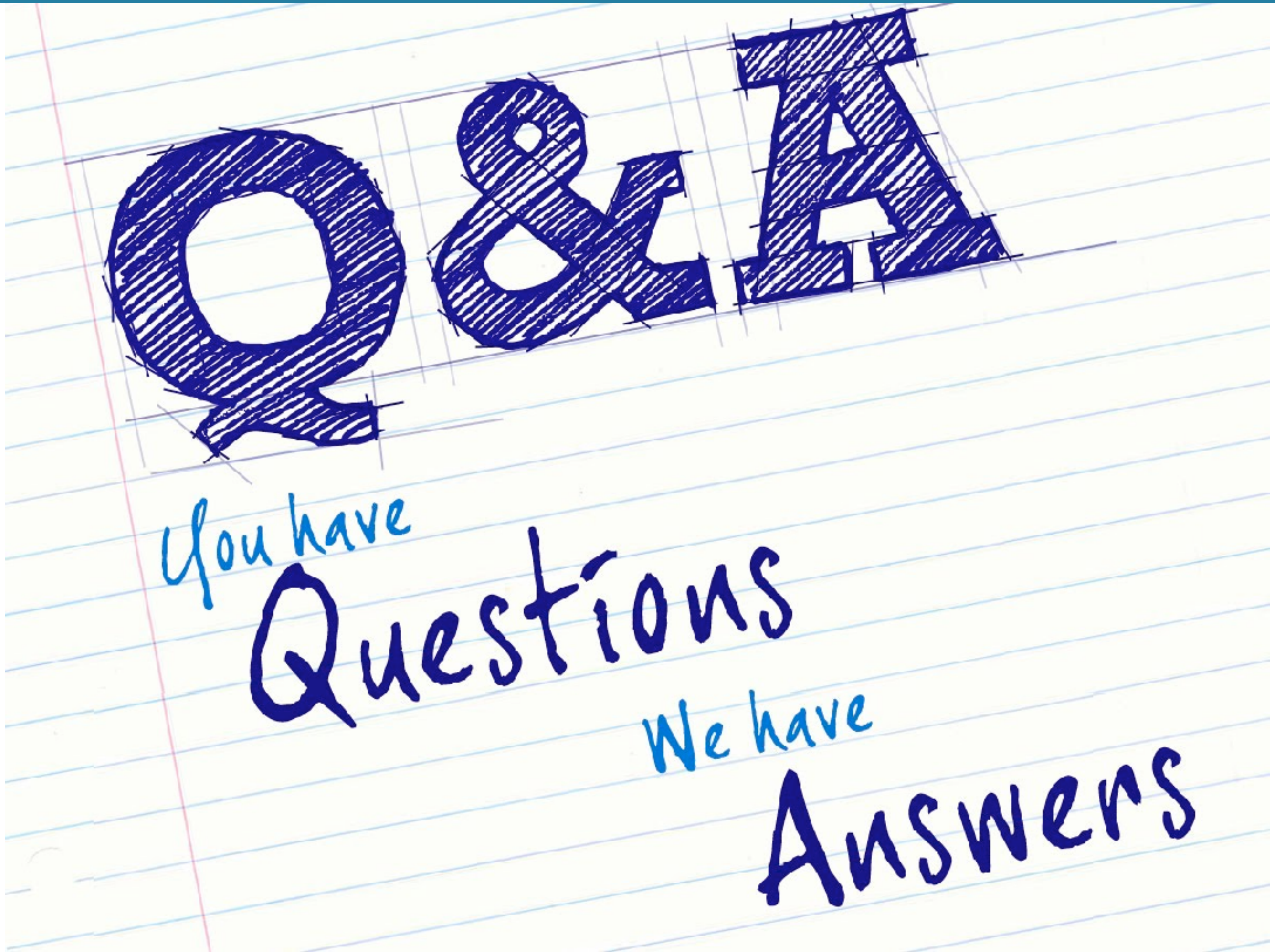
SYN-bit
deep traffic analysis

Application and network troubleshooting

Protocol and packet analysis

Training (Wireshark, TCP, SSL)

www.SYN-bit.nl


Q&A
You have
Questions
We have
Answers