HANDS-ON LAB: Using Wireshark Command Line Tools & Scripting

Sake Blok
Application Delivery Troubleshooter @ SYN-bit
sake.blok@SYN-bit.nl

SHARKFEST **'10**
Stanford University
June 14-17, 2010

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

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

Introductions

- Who are you?

- Who am I?
  - In Networking since 1995
  - Worked for: ISP, Large Bank, Reseller, SYN-bit
  - Use ethereal/wireshark since 1999
  - Develop for wireshark since 2006
    (GUI stuff, bug fixes, IP/TCP/HTTP/SSL)

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

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 :-)

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

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)
  - 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.cap
```

capinfos

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

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

File name: sharkfest-2.cap
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.cap tmp.cap 1-1000 2001-3000
editcap -A "2008-01-17 11:40:00" \
        -B "2008-01-17 11:49:59" example.cap tmp.cap
```

  - split file in chunks

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

  - remove duplicate packets

```
editcap -d example.cap tmp.cap
```

editcap (2)

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

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

  - change timetamps

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

  - change link layer type

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

  - change file type

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

mergecap

- used to merge capture files:
  - based on timestamps

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

  - or just append each file

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

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/)

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

Hands-On Wireshark CLI tools

- Get the files from: http://www.SYN-bit.nl/files/sharkfest-2010.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

Useful shell commands

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

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

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.cap -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.cap -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.cap -w cookie.cap \
  -R `tshark -r example.cap -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`.cap"
   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.cap tmp_*.cap
rm tmp_*.cap
```

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> …`

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

Hands-On Scripting

- Get the files from:
  http://www.SYN-bit.nl/files/sharkfest-2010.zip

- bash required
  (use Windows+cygwin, linux, MacOS/X, etc)

- Goal:
  Explore piping tshark output through other commands to create custom output formats

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

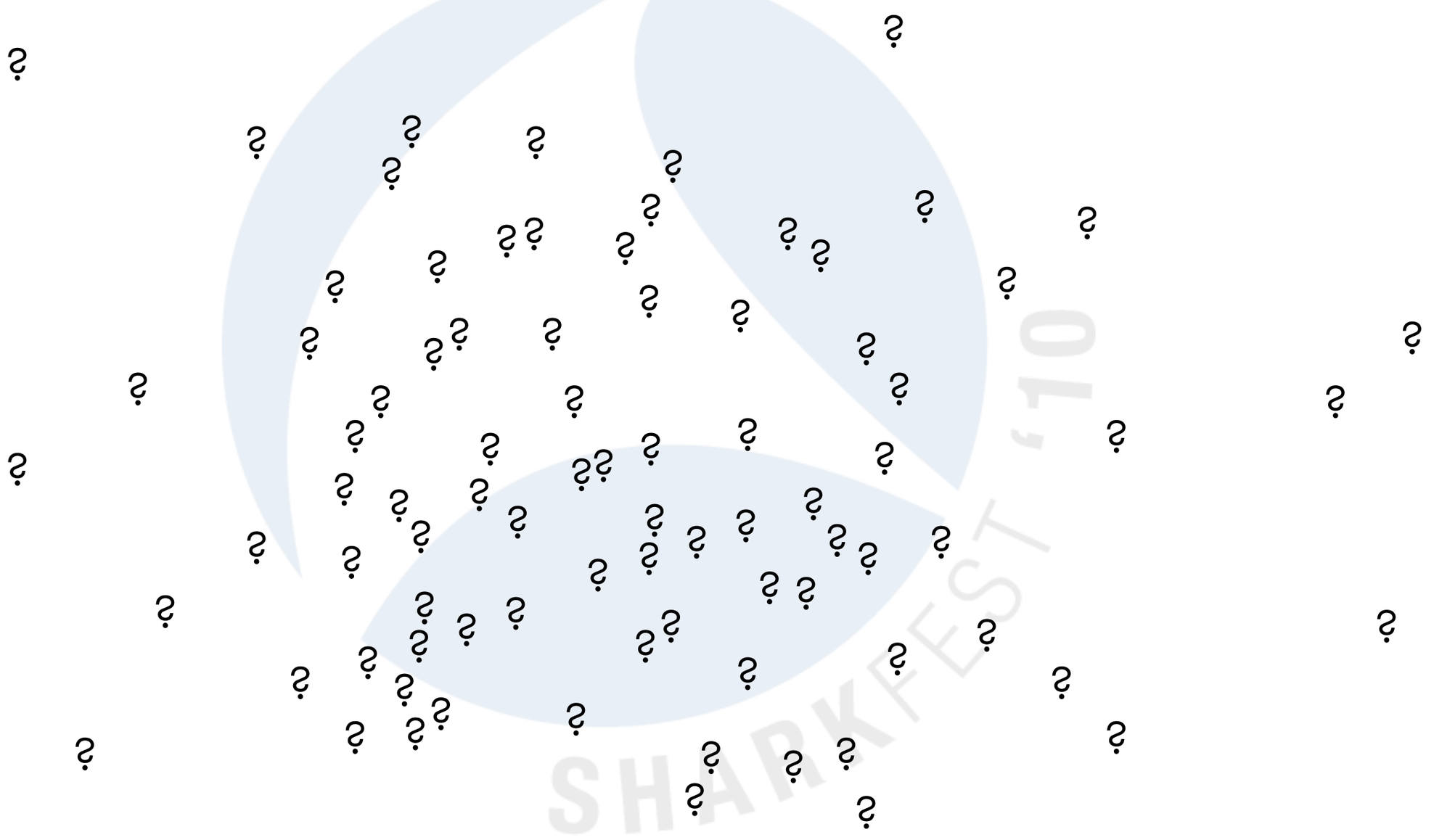
Summary

- tshark, dumpcap, capinfos, editcap, mergecap

- tshark+scripting can complement GUI

- use little building blocks and combine them

Q&A



FIN/ACK, ACK, FIN/ACK, ACK

Thanks!

e-mail: sake.blok@SYN-bit.nl

Hopefully you are triggered to experiment :-)