# Using Wireshark Command Line Tools & Scripting
# HANDS-ON LABGUIDE

**Case 1 : showing the content of a tracefile in different formats (use file http.cap)**

a) First use '`tshark -r http.cap`'
b) Show full decodes (use '`tshark -r http.cap -V`')
c) Show PDML (XML) decodes (use '`tshark -r http.cap -T pdml`')
d) Do a, b and c again, but now pipe the output through the command wc (word count), like '`tshark -r http.cap | wc`'. How much output is generated with each output format? How large was the file http.cap to begin with?

**Case 2 : using "decode as..." in tshark (use file port-1234.cap)**

a) Display the contents of the with tshark. What protocol is recognized for port 1234?
b) Use the option '`-x`' to view hex/ascii output too. What protocol is transported over tcp port 1234?
c) Now use '`tshark -r port-1234.cap -d tcp.port==1234,http`' to decode tcp port 1234 as http. Is it possible to filter on http now?

**Case 3 : using preferences on the command line (use file ssl.cap)**

a) Display the contents of file ssl.cap with tshark, do you see http traffic?
b) Use '`-o ssl.keys_list:192.168.3.3,443,http,key.pem`', do you see http traffic now?
c) Which version of OpenSSL is used by the webserver (use '`-V`' and look at the "`Server: <xxx>`" http header)

**Case 4 : create a new capture file with a selection of packets (use file http.cap)**

a) Use tshark with option '`-o tcp.desegment_tcp_streams:TRUE`' and filter on http
b) Now use tshark with option '`-o tcp.desegment_tcp_streams:FALSE`' and filter on http. How is this output different from the output in 4a?
c) Do 4a and 4b again, but now use '`-w`' to write the output to 4a.cap and 4b.cap respectively. Read 4a.cap and 4b.cap with tshark, can you explain the difference?

**Case 5 : use the tshark -z options (use file mail.cap)**

a) Create a protocol hierarchy with '`-qz io,phs`', which protocols are present in the file?
b) Create a ip conversation list with '`-qz conv,ip`'
c) Create a tcp conversation list with '`-qz conv,tcp`'
d) Create some io statistics with '`-qz io,stat,60,ip,tcp,smtp,pop`'
e) Did the previous commands give you an overview of the contents of mail.cap?

**Case 6 : use editcap to split a tracefile and mergecap to join tracefiles (use file mail.cap)**

a) Execute the command '`editcap -i 60 mail.cap tmp.cap`'. How many files are created?
b) Use '`capinfos -Tcae tmp*`' to display a summary of these new files. Why are the timestamps not exactly 60 seconds apart?
c) Remove the 'tmp*' files
d) Execute the command '`editcap -c 1000 mail.cap tmp.cap`'. How many files are created?
e) Use '`capinfos -Tcae tmp*`' to display a summary of these new files.
f) Use '`mergecap -w mail-new.cap tmp*`'. Is the resulting file exactly the same as mail.cap (tip: use '`cmp <file1> <file2>`')?

**Case 7 : use editcap to adjust timestamps (use file mail.cap)**

a) Use '`editcap -t <delta>`' to create a new tracefile (tmp.cap) where the first packet arrived exactly at 11:39:00 (tip: use '`-V -c1`' to see the exact timestamp of the first packet). What is your '<delta>'?
b) What is the timestamp of the last packet in the new file? Are all packets adjusted with the same '<delta>'?

**Case 8 : Create a new trace file for a specific pop user that contains only his pop sessions. (use mail.cap)**

a) First get an idea of a typical POP session, use :
```
tshark -r mail.cap -R 'tcp.port==64315 and tcp.len>0'
```

b) Use the following steps to create a list of tcp ports used by user 'sake-test2':
   1. Use the filter '`pop.request.parameter=="sake-test2"` ' to only show sessions of user sake-test2
   2. Add '`-T fields -e tcp.srcport`' to the command to just show the tcp ports.
   3. Add `| awk '{printf("%stcp.port==%s",sep,$1);sep="||"}'` to create a display filter that will only display packets belonging to the sessions for user sake-test2.

c) Now use the output of the previous command between backticks to create the new file:
```
tshark -r mail.cap -w sake-test2.cap -R `<previous command>`
```

d) Use '`tshark -r sake-test2.cap -R pop.request.command==USER`' to verify that the new file only contains sessions of user sake-test2. Did we succeed? What went wrong? How can we fix it?

**Case 9 : Extend on case 8 by creating a new trace file for each user automatically. (use the file mail.cap)**

a) Delete the file sake-test2.cap

b) Create a list of users with the following steps:
   1. Use a filter to only select the packets where the pop command was "USER" and use '`-T fields`' to only print the username.
   2. Use '`| sort | uniq`' to create a list of unique usernames

c) Loop through the list of usernames and create the file per user with:

```
for user in `<command from 9b>`
do
  echo $user
  <command from case 8c with $user as variable>
done
```

**Case 10 : CHALLENGE (use file mail.cap)**

Create a shell script [or a one-liner ;-)] that produces the following output:

```
Mail check times for : sake-test1
11:39:43 : 1 message (2833 octets)
11:40:00 : 0 messages (0 octets)
11:42:33 : 7 messages (25958 octets)
11:45:04 : 6 messages (21538 octets)
11:47:37 : 5 messages (17480 octets)
11:50:09 : 8 messages (32297 octets)
11:52:40 : 5 messages (17017 octets)
11:55:13 : 6 messages (21075 octets)
11:57:46 : 6 messages (20859 octets)
12:00:28 : 7 messages (25416 octets)
12:02:49 : 1 message (3677 octets)

Mail check times for : sake-test2
11:39:44 : 5 messages (14512 octets)
11:40:01 : 6 messages (16811 octets)
11:42:34 : 5 messages (17568 octets)
11:45:05 : 4 messages (8551 octets)
11:47:38 : 6 messages (16337 octets)
11:50:10 : 2 messages (5396 octets)
11:52:42 : 7 messages (20601 octets)
11:55:14 : 5 messages (12089 octets)
11:57:46 : 4 messages (14463 octets)
12:00:22 : 5 messages (15016 octets)
12:02:50 : 4 messages (14805 octets)
```

**Send your solutions and/or questions to**

**sake.blok@SYN-bit.nl**