Intrusion Analysis with Wireshark

Learning outcome

1. Use Wireshark to capture traffic
2. Use Wireshark to investigate
3. Combine Snort and Wireshark
4. Correlate data to detect coordinated attacks
5. Subtle and stealthy attacks

Reading tasks and practice at home

1. https://www.youtube.com/watch?v=TkCSr30UojM

Overview

1. Intrusion Analysis Tools
2. Wireshark
3. Data Correlation
4. Subtle and Stealthy Attacks
5. Outgoing Packets

Analyst toolkit

- Tcpdump
- Ngrep
- Wireshark
- P0f
- Tcpflow
- Sancp
- PRADS
- Tcpreplay
- Netdude
- Cygnus HEX Editor
- Chaosreader
- Usniff

ngrep

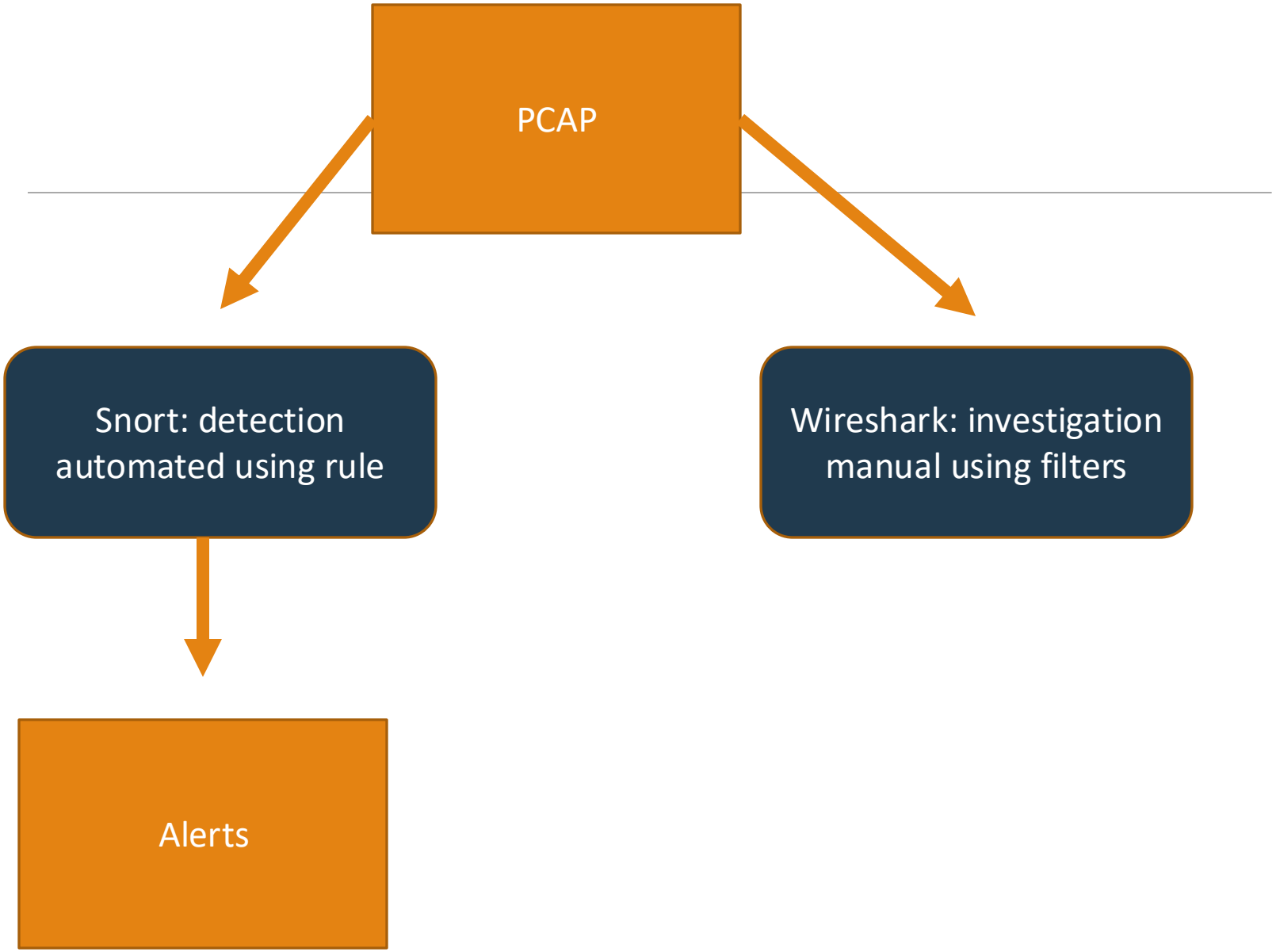
- Network grep
- Understands BPF filters
- Allows content pattern matching against common protocols HTTP, SMTP, FTP
- Uses regular expressions against data payload
- Examples
  - `ngrep -I pcapfile -d 2"user|pass" tcp port 110`
    Looks for traffic on tcp port 110, that match either "user" or "pass" in a regular expression

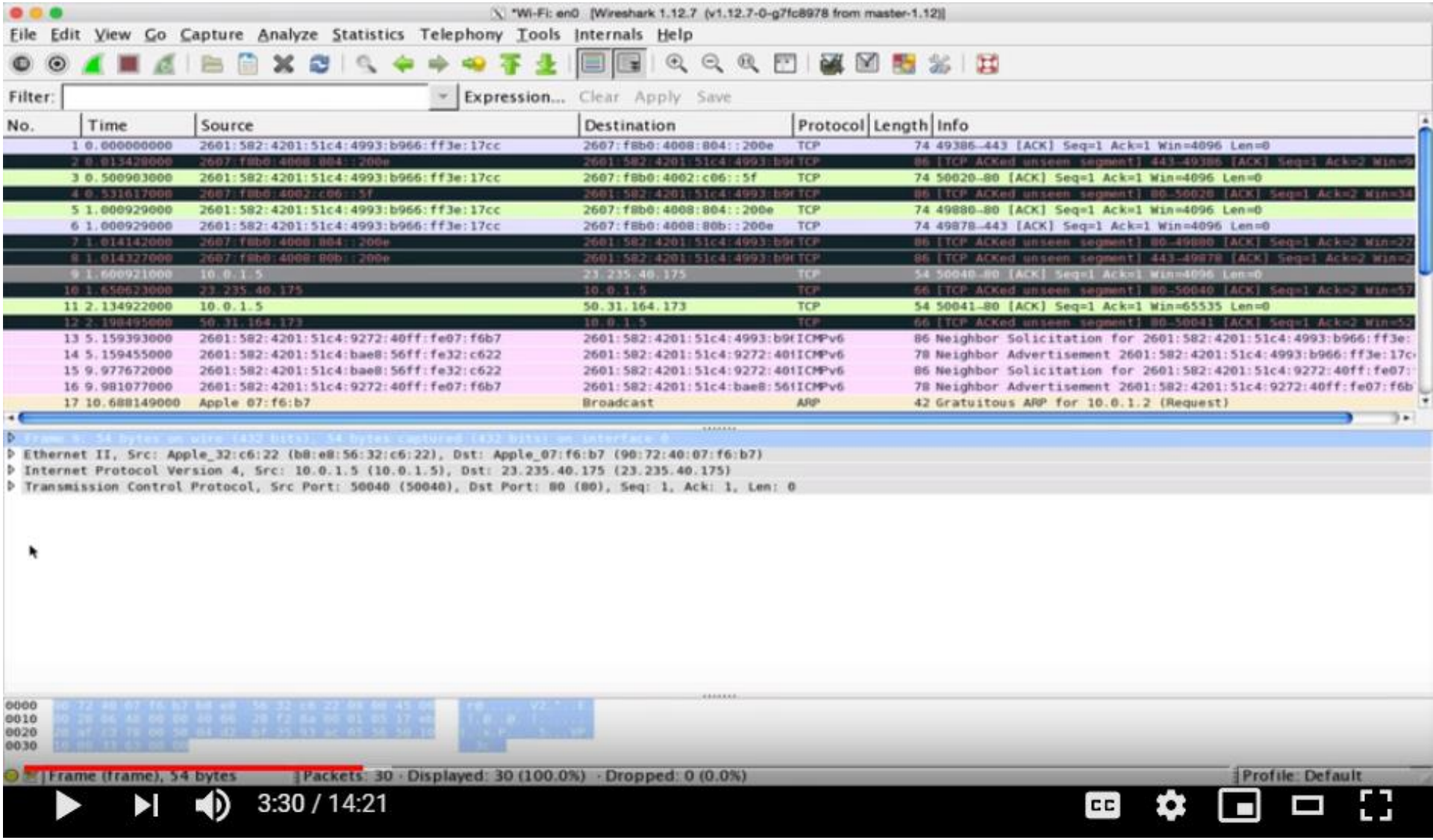
tcpdump

- Easy to use, faster than GUI alternatives
- Packets decoded in HEX, ASCII or both
- Uses BPF tcpdump filters
- Unix version uses single quotes for filters:
  - `'src host 192.168.0.103 and dst port 25'`
  - `src host 192.168.0 and 'tcp[13] & (0x12) = 0x12'`

WIRESHARK

- GUI !!!!
- Application layer decoding
- Easy to use display filters
  - Easy to use Expression Builder
- Protocol Hierarchy
  - Breakdown of protocols. Filters traffic based on these.
- Expert Info Composite
  - Displays uncommon network behaviour. Displays Errors, Warnings, etc
- Follow TCP Stream
  - Displays entire session in ASCII
  - Can carve files out of a packet stream (Save as Raw data)
- Export Objects
  - Can carve files out of HTTP, SMB

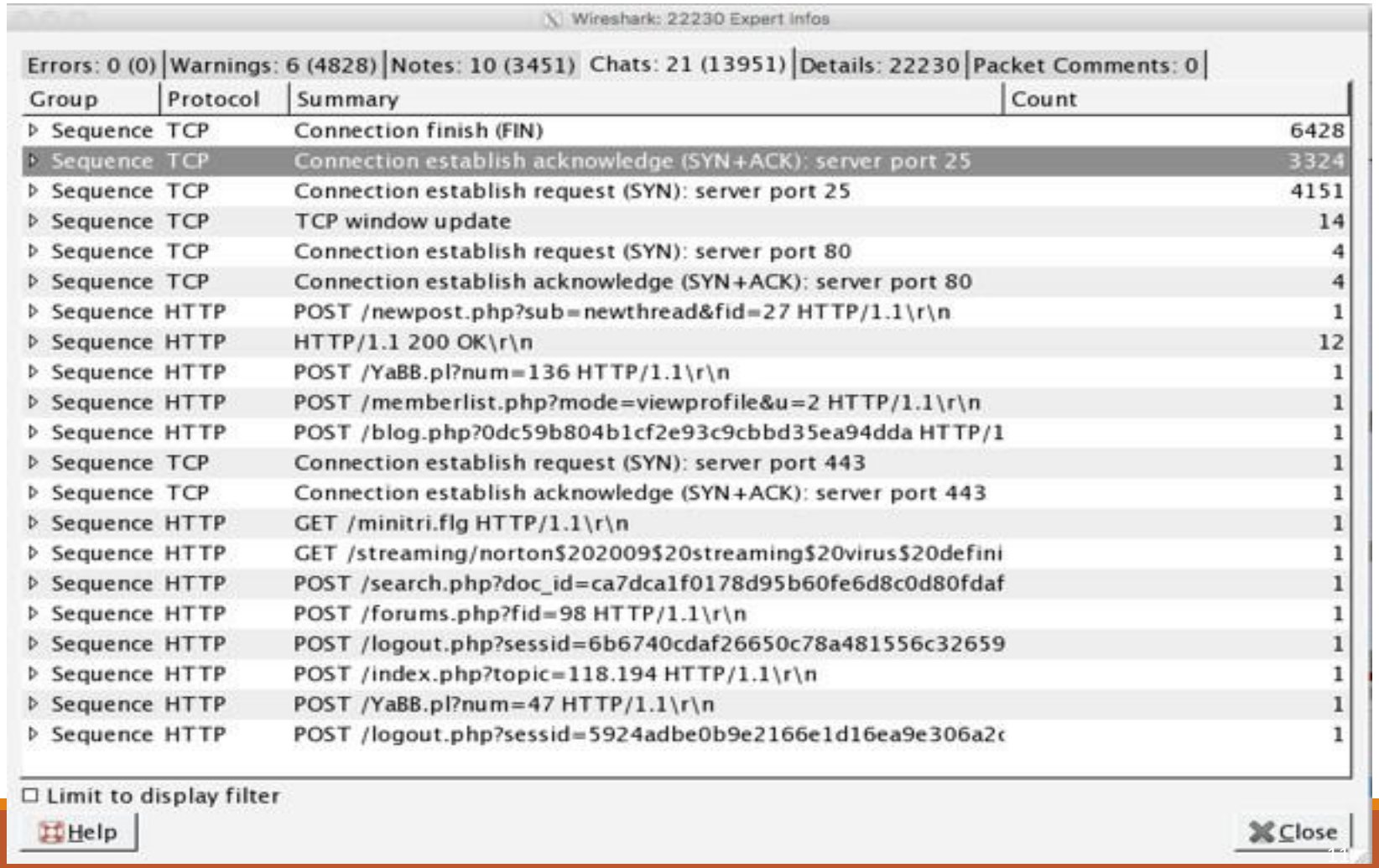

PCAP
Snort: detection automated using rule
Wireshark: investigation manual using filters
Alerts



https://www.youtube.com/watch?v=TkCSr30UojM

Lab Extra

Wireshark: Analyze ->Expert Information



Expert information

Errors : Serious problems can be Ethernet checksum errors, malformed packets, or missing fields in a protocol header.

Warnings : a problem in the application or in communications like TCP zero-window, TCP window-full, previous segment not captured, out-of-order segment, and any issue that is unnatural to the protocol behavior.

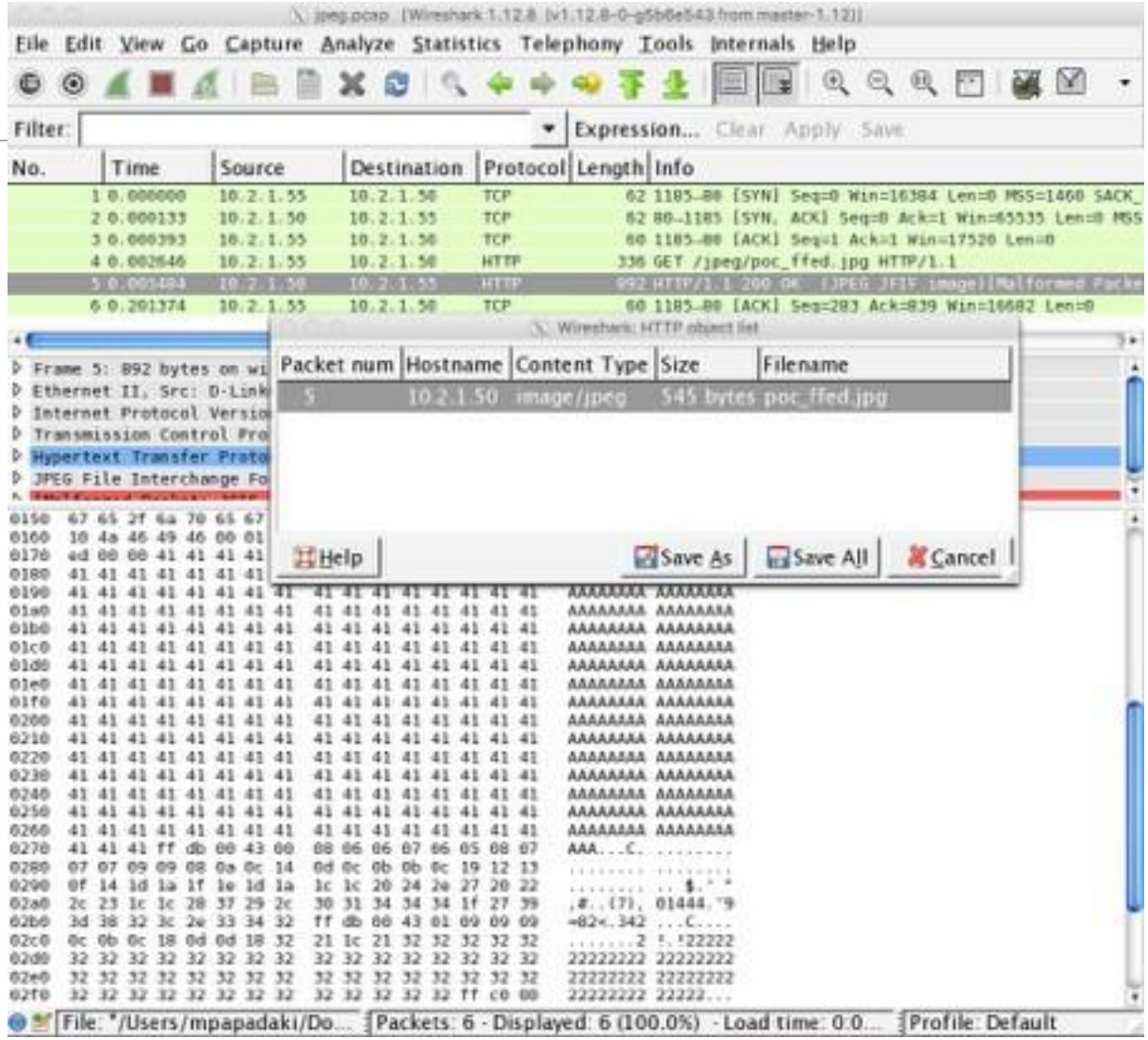
Notes : indicates an event that may cause a problem but is still inside the normal behavior of the protocol. TCP retransmission. Example: duplicate ACK, fast retransmission

Chats : provides information about usual workflow, for example TCP connection establish request (SYN), TCP connection establish acknowledge (SYN + ACK), connection reset (RST), HTTP GET, and HTTP POST:

Packet comments: You can manually add a comment to every packet. This will appear down to the chat on the expert information window.

Wireshark: File -> Export Objects

You need to examine items from network traffic



Export objects

1. Export objects (files)
2. DO NOT RUN any file in your machine
3. Examine the file type

file [option] [filename]

3. Check if it is a malware

- Calculate hash value

Sha256sum [filename]

- Check hash value against VirusTotal/ google

Exercise

1. Inside your VIRTUAL MACHINE, download example.zip from DLE
2. Unzip the file
3. Run wireshark: sudo wireshark
4. Inside wireshark, open example1.pcap

Download data for demonstration

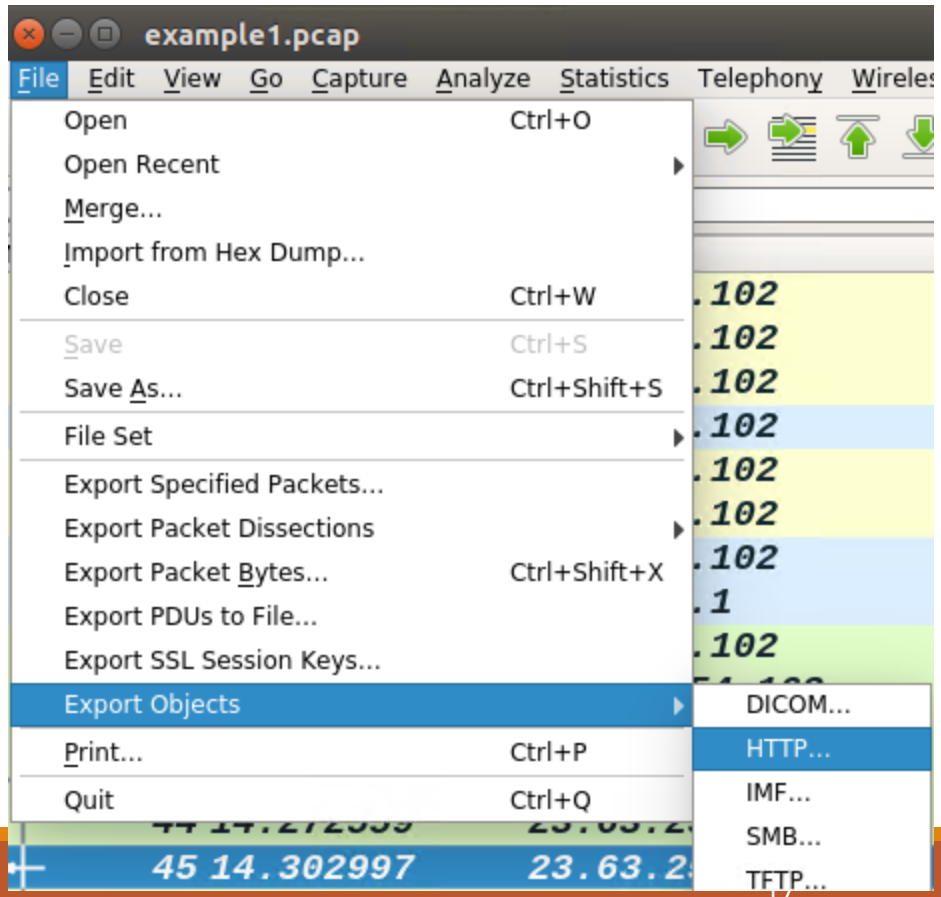
Go to DLE -> section “Lectures” -> download “Additional material-L07-Pcap examples”

Export objects from http traffic (http get request)

Choose File -> Export object -> HTTP

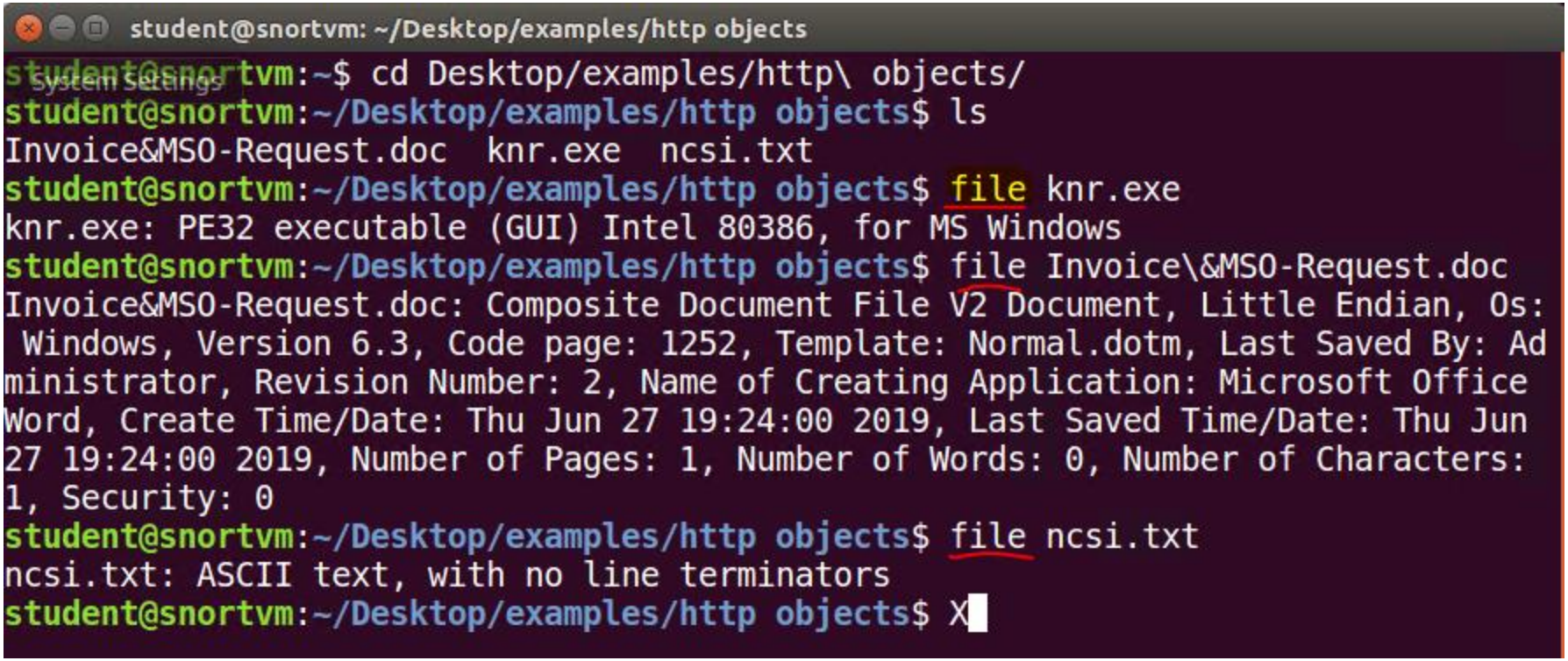
Demo: use

example1.pcap

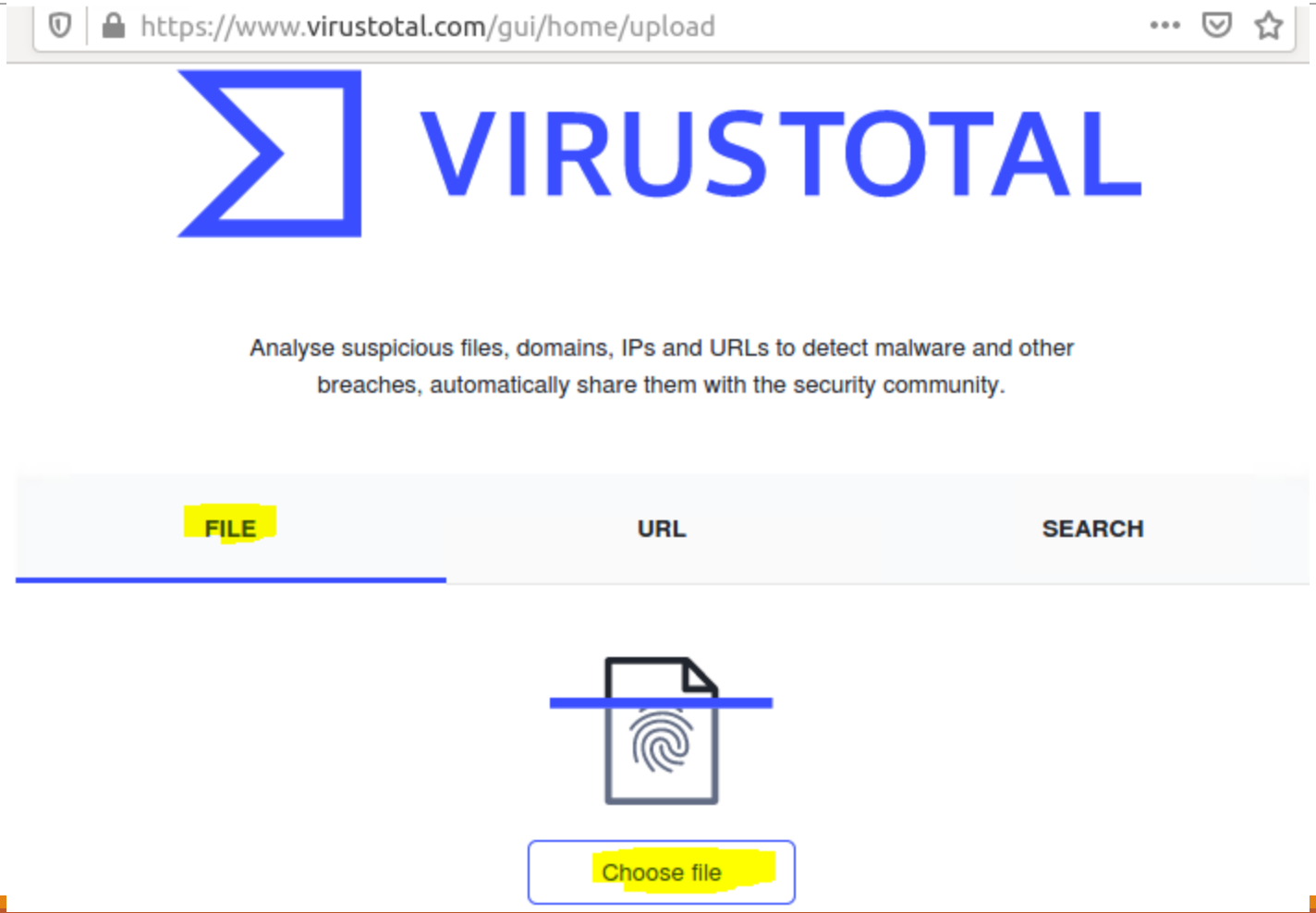


Examine file type

file [option] [filename]

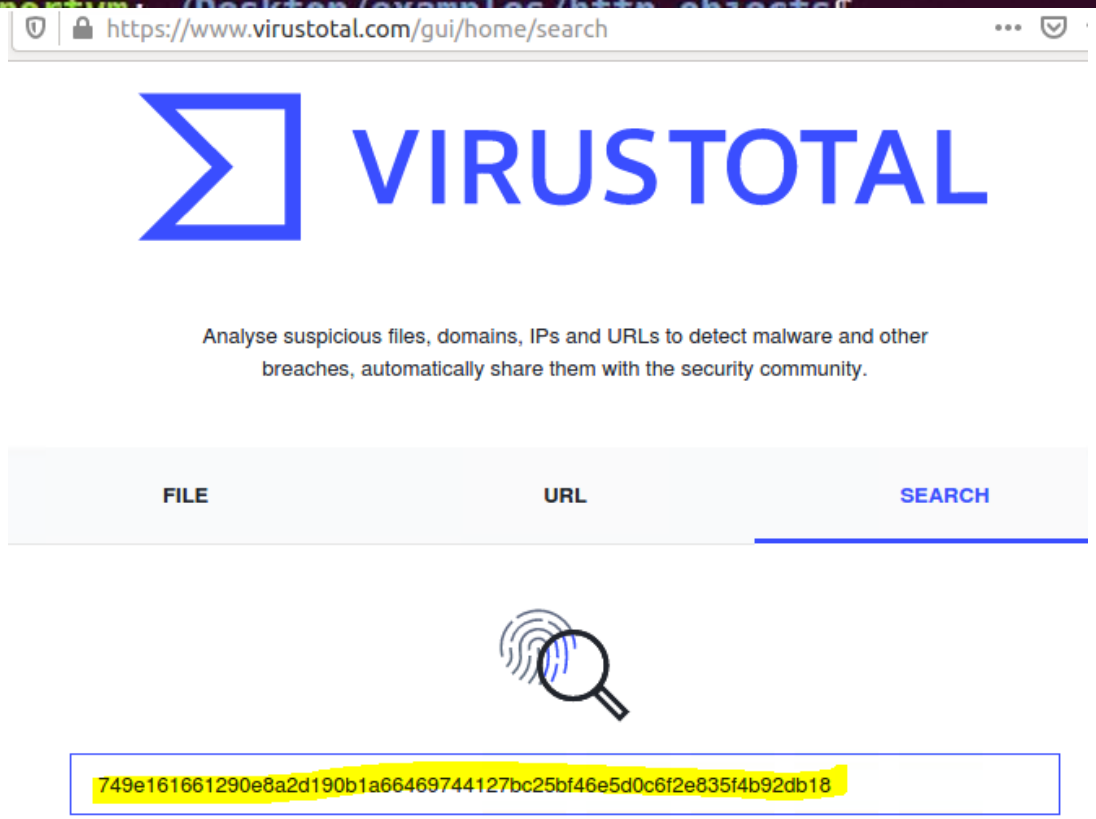


Check if it is a malware

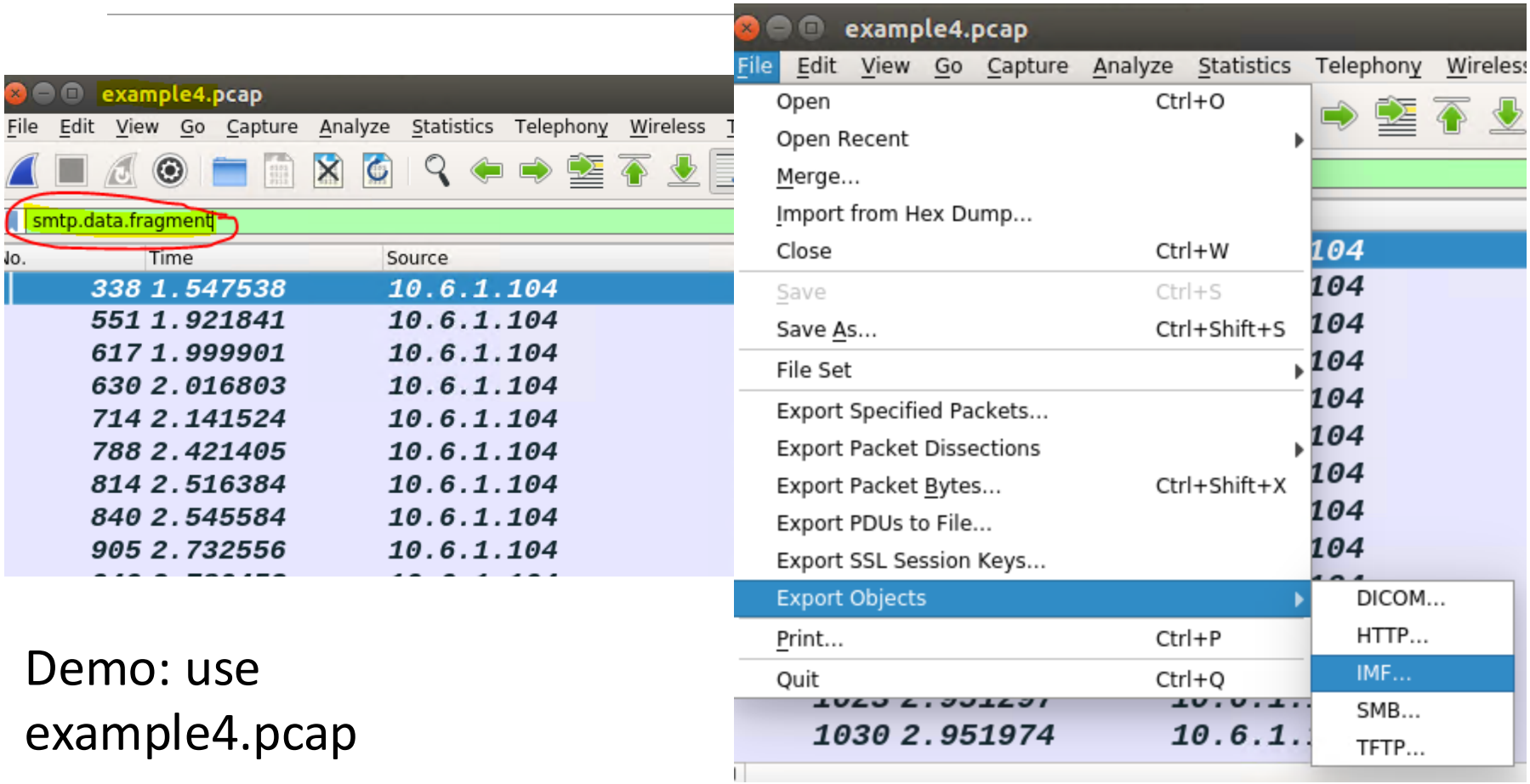


Check if it is a malware

```
student@snortvm:~/Desktop/examples/http objects$ sha256sum knr.exe
749e161661290e8a2d190b1a66469744127bc25bf46e5d0c6f2e835f4b92db18  knr.exe
```

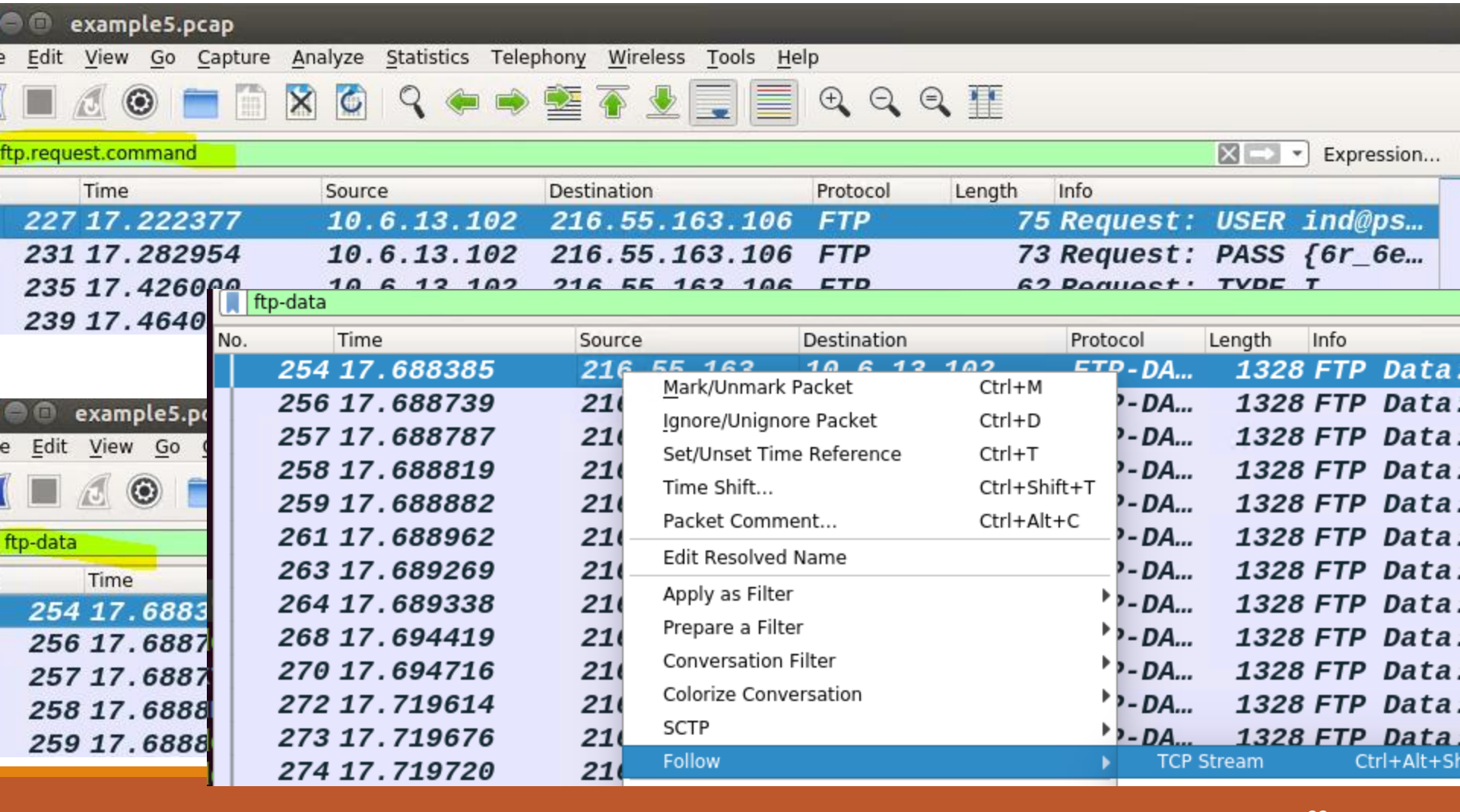


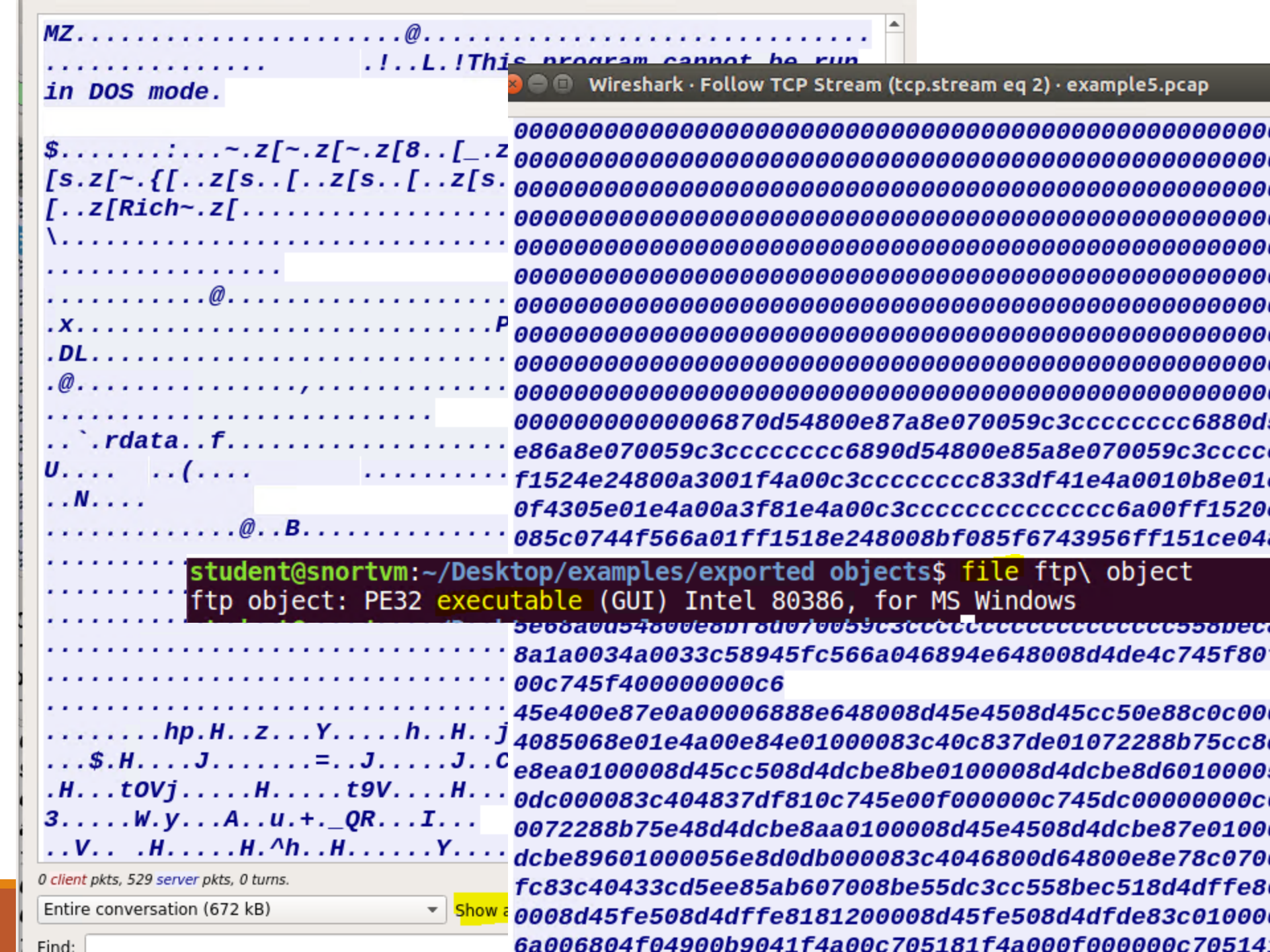
Export objects from smtp



Demo: use example4.pcap

Export objects from ftp (use example5.pcap)

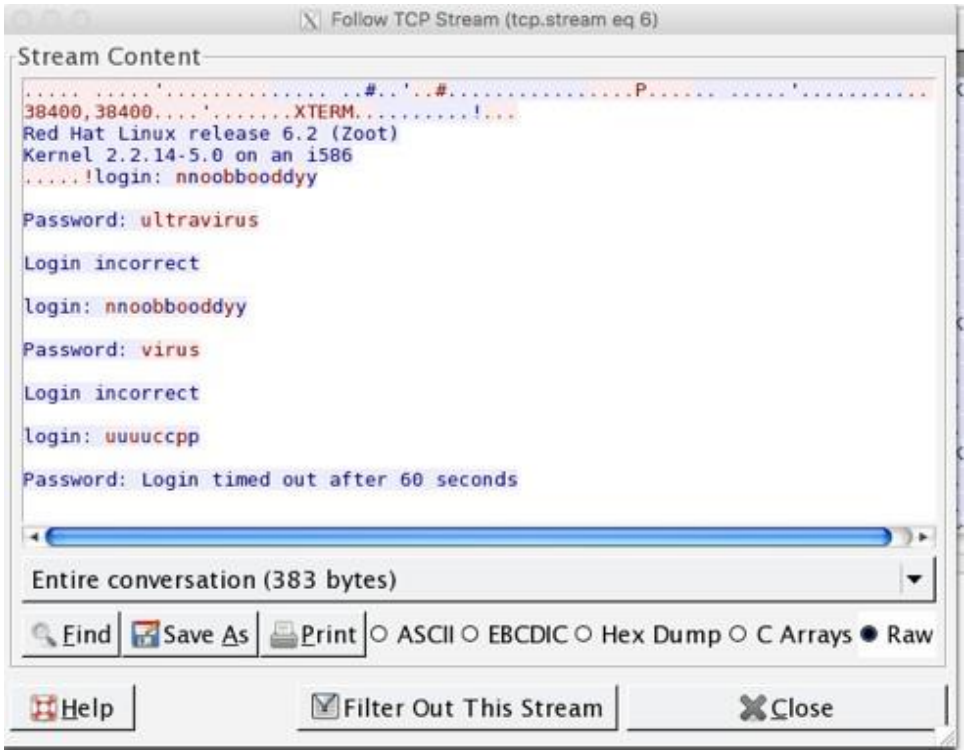



Wireshark · Follow TCP Stream (tcp.stream eq 2) · example5.pcap
student@snortvm:~/Desktop/examples/exported objects$ file ftp\ object
ftp object: PE32 executable (GUI) Intel 80386, for MS Windows
0 client pkts, 529 server pkts, 0 turns.
Entire conversation (672 kB)
Find:

Wireshark: Follow TCP Stream

Analyze, isolate and reassemble the TCP data from the first SYN-SYN/ACK/ACK handshake to the end of the connection into a human-readable format. Client requests are shown in red, and any text in blue denotes the response received from the server.

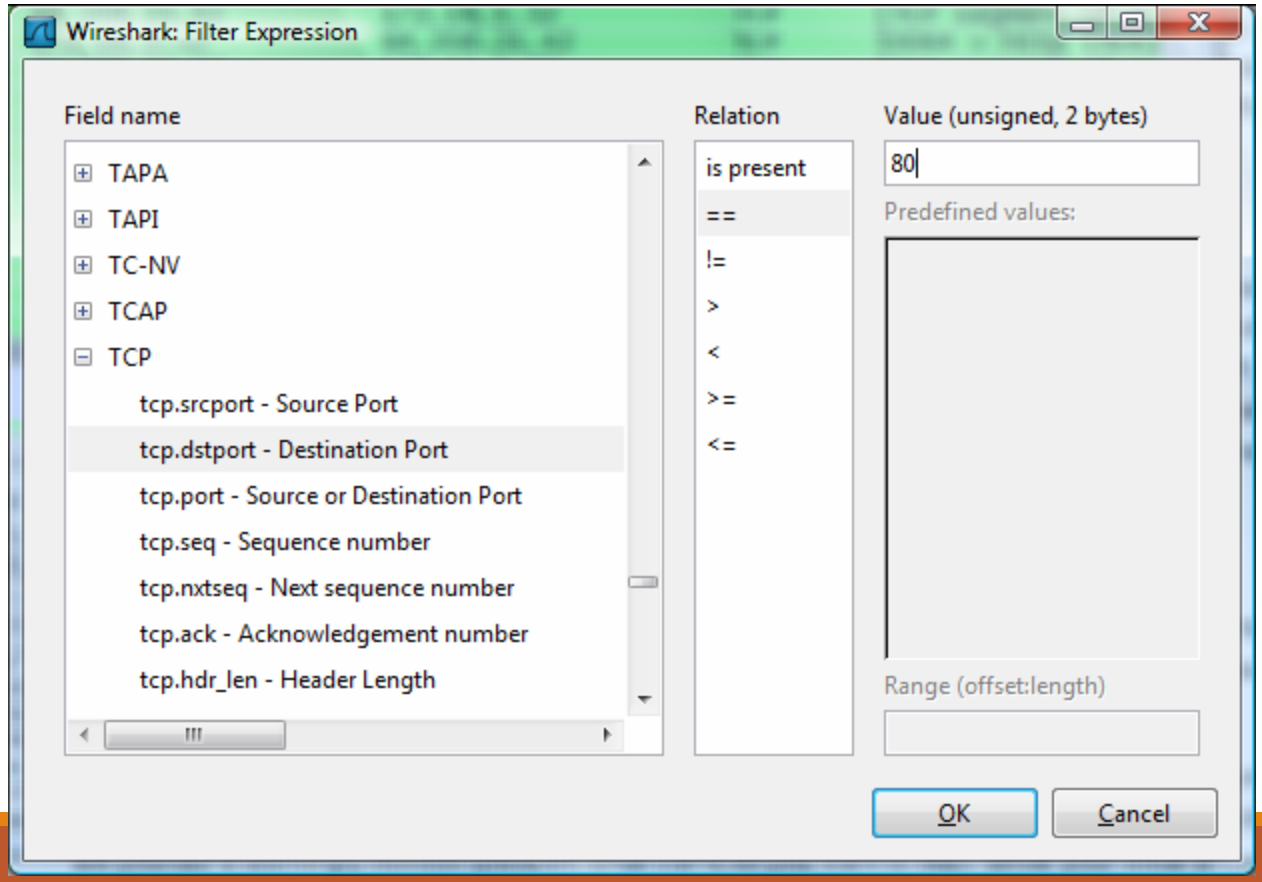
How: Select a packet. Choose Analyse -> Follow -> TCP stream



https://www.youtube.com/watch?v=ixVXfUBEciU

Writing filters method 1

To use expression builder utility, simply click the word "Expression" on the filter toolbar and fill in the fields in the pop-up window



Filter language

You can use English and C-like terms in the same way, they can even be mixed in a filter string.

Table 6.5. Display Filter comparison operators

English	C-like	Description and example
eq	==	Equal. `ip.src==10.0.0.5`
ne	!=	Not equal. `ip.src!=10.0.0.5`
gt	>	Greater than. `frame.len > 10`
lt	<	Less than. `frame.len < 128`
ge	>=	Greater than or equal to. `frame.len ge 0x100`

https://www.wireshark.org/docs/wsug_html_chunked/ChWorkBuildDisplayFilterSection.html

Filter language

Ethernet address

6 bytes separated by a colon (:), dot (.) or dash (-) with one or two bytes between separators:

eth.dst == ff:ff:ff:ff:ff:ff

eth.dst == ff-ff-ff-ff-ff-ff

eth.dst == ffff.ffff.ffff

IPv4 address

ip.addr == 192.168.0.1

Filter language – time slot

frame.time == "Oct 15, 2012 16:36:01.009638000"

Edit this display filter. Change the "==" to ">=" and change the time to the earliest time you want your display filter to show. For example:

frame.time >= "Oct 15, 2012 16:00:00"

Now right-click on Arrival Time again and select Prepare a Filter > And Selected. You will now have something like this:

(frame.time >= "Oct 15, 2012 16:00:00") && (frame.time == "Oct 15, 2012 16:36:01.009638000")

Now edit the second part of the filter. Change "==" to "<=" and change the time to the latest time you want your display filter to show. For example:

(frame.time >= "Oct 15, 2012 16:00:00") && (frame.time <= "Oct 15, 2012 17:00:00")

Combining expressions

Table 6.6. Display Filter Logical Operations

English	C-like	Description and example
and	&&	Logical AND. `ip.src==10.0.0.5 and tcp.flags.fin`
or	||	Logical OR. `ip.scr==10.0.0.5 or ip.src==192.1.1.1`
xor	^^	Logical XOR. `tr.dst[0:3] == 0.6.29 xor tr.src[0:3] == 0.6.29`
not	!	Logical NOT. `not llc`
[...]		See "Slice Operator" below.
in		See "Membership Operator" below.

Filters method examples

1. Monitor the network for connections using insecure protocols, such as telnet. The following filter will identify any TCP port 23 traffic that may contain cleartext authentication information: tcp.dstport == 23
2. Isolate traffic from a potentially compromised system. For example, you could use the following filter to monitor all traffic from 192.168.2.101: ip.src == 192.168.2.101
3. Investigate denial-of-service attacks. If your Web server exhibits the symptoms of a denial of service, you can use Wireshark to capture and analyze the relevant packets, allowing you to craft defenses against the threat. For example, the filter below captures all Web traffic headed to the server located at 192.168.3.134: ip.dst == 192.168.3.134 and tcp.dstport == 80

Exercise – Write filter

menti

1. Display only packets sent from or to specific MAC addresses 00:50:7f:cd:d5:38
2. Display only packets from specific IP addresse 192.168.1.1
3. Display only packets coming from network 192.168.1.0/24
4. Packets from network 10.0.0.0/24 to a website that contains the word packt
5. All packets that are not ICMP and not ARP

1. eth.src == 10:0b:a9:33:64:18 || eth.dst == 10:0b:a9:33:64:18
2. ip.addr == x.x.x.x
3. ip.src==192.168.1.0/24
4. ip.src == 10.0.0.0/24 and http.host contains "packt"
Packets from networks 10.0.0.0/24 to websites that end with .com: ip.addr == 10.0.0.0/24 and http.host matches ".com$"
5.All packets that are not ICMP and not ARP:
!arp || !icmp
not arp&&not icmp

Exercise – Write filter

Menti

7. Display all packets to the HTTP server
8. Display only TCP SYN packets
9. Packets with HTTP GET methods
10. All cookies sent by Google servers to your PC
11. All HTTP packets that contain a ZIP file

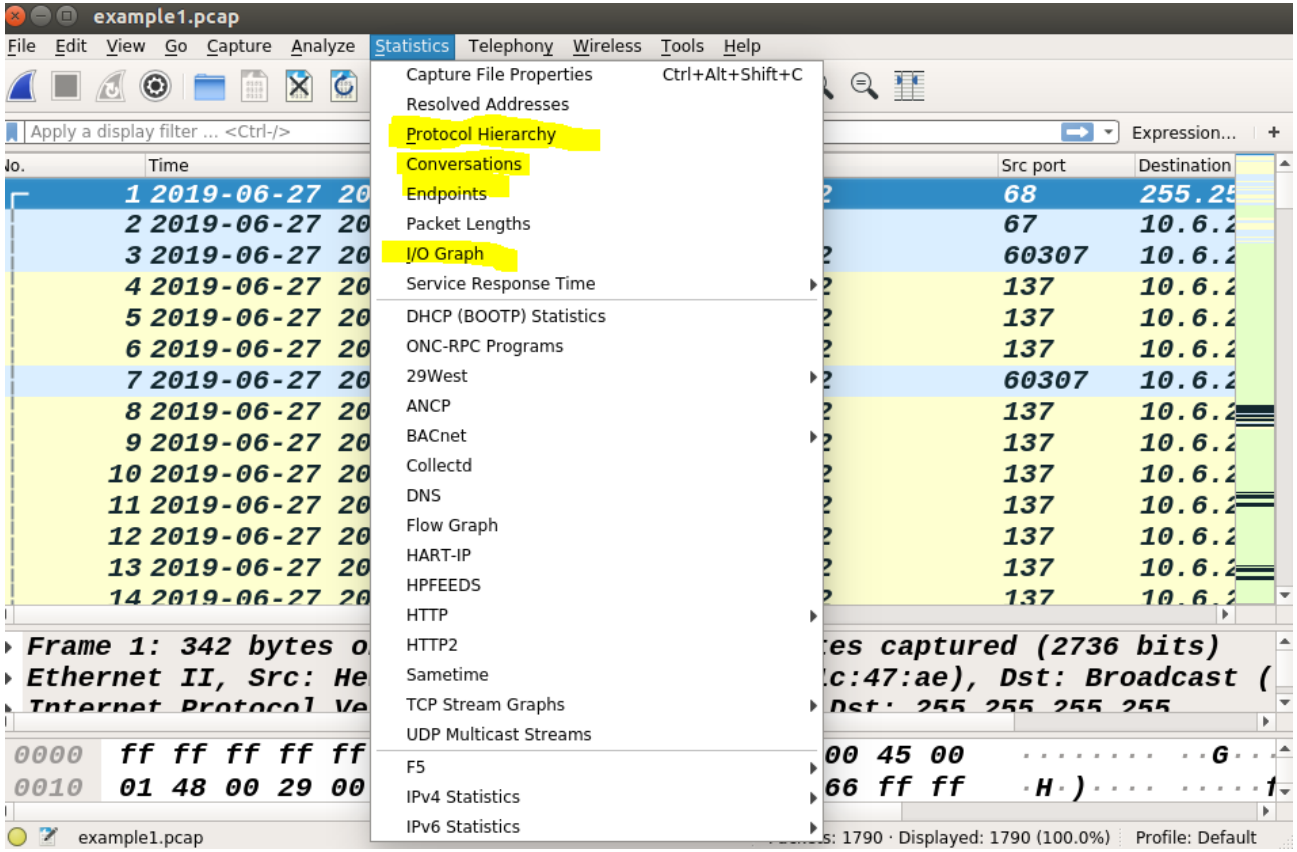
1. All packets to the HTTP server: tcp.dstport==80
2. Display only TCP SYN packets: tcp.flags.syn == 1
3. Packets with HTTP GET methods: http.request.method == "GET"
4. All cookies sent by Google servers to your PC: (http.set_cookie) && (http contains "google")
5. All HTTP packets that contain a ZIP file : http matches ".zip" && http.request.method == "GET"

Exercise

Open ”traffic-for-wireshark-column-setup.pcap“ (on DLE, “Additional material-L07-traffic for wireshark column setup (pcap file)”) in Wireshark

1. Display HTTP requests
2. Display domain names in https or ssl/tls packets
3. Filter out packets used to discover Plug & Play devices and not related with normal web traffic (i.e. Simple Service Discovery Protocol reques or SSDP requests). These packets have udp port 1900.
4. SYN packets
5. DNS activities
6. Ftp
7. Email communication
8. Email communication contains "From: "

Statistic information



Data Correlation

NETWORK MAPPING – DNS ZONE VARIATION – LOGS – SNORT DETECTION

Data Correlation

- Coordinated attacks. Example: each of the attackers scans a subset of victims
- Successful analysis requires:
  - Fusing observations from multiple types of sensors
  - Correlating observations from similar sensors
  - Building the answer one piece at a time
- Primary key: maintain situational awareness

External Network Mapping

Correlating data from multiple sources to the host machine

```
10:32:24.722 north.mappem.com.38758 > ns.target.net.33476: udp 12
10:32:24.756 north.mappem.com.38758 > ns.target.net.33477: udp 12
10:32:24.801 north.mappem.com.38758 > ns.target.net.33478: udp 12
10:32:24.833 north.mappem.com.38758 > ns.target.net.33479: udp 12
10:32:24.944 north.mappem.com.38758 > ns.target.net.33481: udp 12
10:32:24.975 north.mappem.com.38758 > ns.target.net.33482: udp 12

10:32:26.745 south.mappem.com.48412 > ns.target.net.33512: udp 12
10:32:26.837 south.mappem.com.48412 > ns.target.net.33513: udp 12
10:32:26.930 south.mappem.com.48412 > ns.target.net.33514: udp 12
10:32:27.033 south.mappem.com.48412 > ns.target.net.33515: udp 12
10:32:27.231 south.mappem.com.48412 > ns.target.net.33517: udp 12
10:32:27.436 south.mappem.com.48412 > ns.target.net.33519: udp 12
```

Simultaneous traceroutes from multiple locations within minutes. This provides timing data for multiple paths.

Often used by ISPs to determine the best route to a site so they can provide the best service next time you click a URL.

DNS Zone Variation (1)

Correlating data from different sensors (network traffic, running processes)

```
01:46:06.41 attacker1.23616 > target.domain:  S 4076745461:4076745461(0) win
512 <mss 1460>
01:46:06.42 target.domain > attacker1.23616:  S 2085251122:2085251122(0) ack
4076745462 win 17520 <mss 1460> (DF)
01:46:07.14 attacker1.23616 > target.domain:  . ack 1 win 31744 (DF)
01:46:07.34 attacker1.23616 > target.domain:  P 1:3(2) ack 1 win 31744 (DF)
01:46:07.51 target.domain > attacker1.23616:  . Ack 3 win 17520 (DF)
01:46:07.58 attacker1.23616 > target.domain:  . 3:1463(1460) ack 1 win 31744
(DF)
01:46:07.61 attacker1.23616 > target.domain:  P 1463:1563(100) ack 1 win 31744
(DF)
01:46:07.58 attacker1.23616 > target.domain:  F 1563:1563(0) ack 1 win 31744
```

One IP attacks, the second IP receives the data

Attacker1 scans the net until it finds a listening DNS server on TCP port 53. There is data transfer. Attacker2 (next slide) receives the data

DNS Zone Variation (2)

```
%strings ibm|grep bin
/usr/X11R6/bin/xterm -display Attacker2: 0
/usr/X11R6/bin/xterm -display Attacker2: 0
/usr/X11R6/bin/xterm -display Attacker2: 0
/usr/X11R6/bin/xterm -display Attacker2: 0
/usr/X11R6/bin/xterm -display Attacker2: 0
```

One IP attacks, the second IP receives the data

Cleaned up content from packets sent to the DNS server shows: Attacker2 opens up terminal access to the DNS server

Note: xterm is the standard terminal emulator for the X Window System. It allows users to run programs which require a command-line interface

Network Logger and Host Log

Correlating data from different sensors (network logger and log in the server)

```
Scanning for Proxies:

202.101.169.103 ; 3979 ; hxx.Triumf.CA ; 3128 ; Tue Oct 12 11:42:54 2002

202.101.169.103 ; 4171; mxx.Triumf.CA ; 3128 ; Tue Oct 12 11:43:53 2002


The one denied access to a real Squid server:

939753769.030  440 202.101.169.103  TCP_DENIED/400 517 GET
  http://www.s3.com - NONE/- -
```

Proxy Probe Correlation

The packets are clearly not being stopped by the perimeter defense, but the host did deny the access. It is critical to check for any systems that responded back with a SYN/ACK

Correlation with Snort Detection

Correlating data from different sensors (Snort and host log)

```
Mar 18 03:57:54 tempest snort: [1:0:0] IDS296/web-misc_http-whisker-
  splicing-attack-space [Classification: suspicious miscellaneous traffic]
  [Priority: 1]: {TCP} 218.62.92.153:4767 -> 195.82.120.100:80

Mar 18 03:57:54 tempest snort: [1:0:0] IDS243/web-cgi_http-cgi-pipe
  {Classification: system integrity attempt] [Priority: 11]: {TCP}
  218.62.92.153:4767 -> 195.82.120.100:80


218.62.92.153 - - [19/Mar/2003:15:34:24 +0000] "GET
  /data/snort_daily.html  HTTP/1.0" 200 41567 "-" "libwww-perl/5.21"
```

- Snort daily logs were shared through a perl script in this site.
- Source IP addresses of attacks that appear in the snort logs, match source IP addresses of hosts checking the daily logs the following day
  Not bad for free remote verification!

Subtle and Stealthy Attacks

INTRUDER TECHNIQUES

Subtle and Stealthy Attacks

- Attacks that typically fall below the noise threshold on a network and are difficult to detect
- Attacks that are embedded within a large volume of similar traffic that is too massive to realistically process

Stealthy Ping Mapping

```
Apr 21 ->
18:41:03.722 srn.org > 10.168.12.255: icmp: echo request
18:41:05.756 srn.org > 10.168.12.255: icmp: echo request
19:32:51.801 srn.org > 10.38.13.255: icmp: echo request
19:32:53.833 srn.org > 10.38.13.255: icmp: echo request
21:29:59.944 srn.org > 10.168.13.255: icmp: echo request
21:30:00.975 srn.org > 10.168.13.255: icmp: echo request

Apr 22 ->
01:11:07.745 srn.org > 10.38.15.255: icmp: echo request
01:11:09.837 srn.org > 10.38.15.255: icmp: echo request
02:11:40.930 srn.org > 10.168.15.255: icmp: echo request
02:11:42.033 srn.org > 10.168.15.255: icmp: echo request
```

Slow scan of 2 sites to Evade Detection
Most IDS systems don't hold connections for evaluation for more than 1 hour.

Inverse Network Mapping

```
02:58:05.722 srn.org.25984 > 172.30.69.23.2271: R 0:0(0) ack
  674719802 win 0
02:50:11.756 srn.org.50620 > 172.16.7.158.1050: R 0:0(0) ack
  674719802 win 0
02:59:20.801 srn.org.19801 > 192.168.184.174.1478: R 0:0(0) ack
  674719802 win 0
02:59:31.833 srn.org.7960 > 192.168.242.139.1728: R 0:0(0) ack
  674719802 win 0
03:00:58.944 srn.org.35124 > 192.168.182.171.1626: R 0:0(0) ack
  674719802 win 0
```

Using TCP RESET Packets: The attacker is sending TCP packets to close connections that were never established.

- If destination host does not exist;
- If packet is not filtered and reaches a router that has not been silenced;

Then the router will respond with ICMP host Unreachable error.

Inverse Network Mapping

```
07:36:32 stealth.com.domain > 172.29.185.48.54358: udp
07:57:01 stealth.com.domain > 10.242.221.165.13043: udp
09:55:28 stealth.com.domain > 192.168.203.163.15253: udp
10:38:53 stealth.com.domain > 192.168.126.131.39915: udp
10:40:37 stealth.com.domain > 192.168.151.126.19038: udp
10:44:28 stealth.com.domain > 172.29.96.220.8479: udp
11:35:40 stealth.com.domain > 192.168.7.246.44451: udp
```

Faking a DNS Response: The attacker is sending responses to questions that were never posed!

- The internal hosts never sent DNS queries in the first place
- If the packet is not filtered and reaches a router that has not been silenced, then the router will respond with ICMP Host Unreachable error.

Outgoing Packets

ICMP ERRORS – COMPROMISED TELNET

Outgoing Packets

- Often analysts focus on incoming traffic only
- It pays to inspect outgoing traffic as well
  - Someone using your network to launch attacks against other organisations
  - An internal infected system trying to propagate
  - An internal infected system reporting to their master

Inferring Outbound Traffic from Inbound ICMP Errors

```
12:58:47.697873 216.191.229.53 > 172.16.248.252: icmp: time
  exceeded in-transit
12:58:47.828249 205.235.142.158 > 172.16.248.252: icmp: time
  exceeded in-transit
12:58:47.858500 206.248.221.49 > 172.16.248.252: icmp: host
  129.97.175.143 unreachable – admin prohibited filter
12:58:47.918470 200.220.72.126 > 172.16.248.252: icmp:  host
  129.222.48.106 unreachable – admin prohibited filter
```

Unreachables reflect failed outbound attempts

- Searching the outbound traffic from host 172.16.248.252 would confirm this hypothesis.
- The payload of the ICMP Error messages will contain the IP header plus the first 8 bytes of the original datagram's payload that caused the error in the first place

Compromised Telnet Server

```
10.10.102.182.2828 > 206.132.179.136.21: S 809165349:809165349(0)
  win 512
206.132.179.136.21 > 10.10.102.182.2828: S 3182517100:3182517100(0)
  ack 809165350 win 17520 (DF)
10.10.102.182.2828 > 206.132.179.136.21: . ack 1 win 512
206.132.179.136.21 > 10.10.102.182.2828: P 1:101(100) ack 1 win
  17520
…
USER dadropsit
PASS amdhax0r
…
RETR
QUIT warez.tar
```

- 10.10.102.182 was known to offer telnet access. It was not known to generate outbound traffic, until it was compromised
- One day, outbound traffic was captured going to ftp server 206.132.179.136
- This change in its profile was enough to raise alarms. Recovering the payload shows the username and password they used

Compromised Telnet Server (2)

```
10.10.102.182.2647 > 199.3.235.130.6667: S
  2335338658:2335338658(0) win 512
199.3.235.130.6667 > 10.10.102.182.2647: S 4291582374:4291582374
  (0) ack 2335338659 win 17520 (DF)
10.10.102.182.2647 > 199.3.235.130.6667: P 1:11(10) ack 1 win
  32120 (DF)
199.3.235.130.6667 > 10.10.102.182.2647: P 1:46(45) ack 1 win
  17520 (DF)
```

- Apart from the ftp session, the hacker also visited IRC server 199.3.235.130
- If you have a server that initiates no outbound sessions, you could create a rule that will look for outbound initial SYN traffic from that server.

Thank You