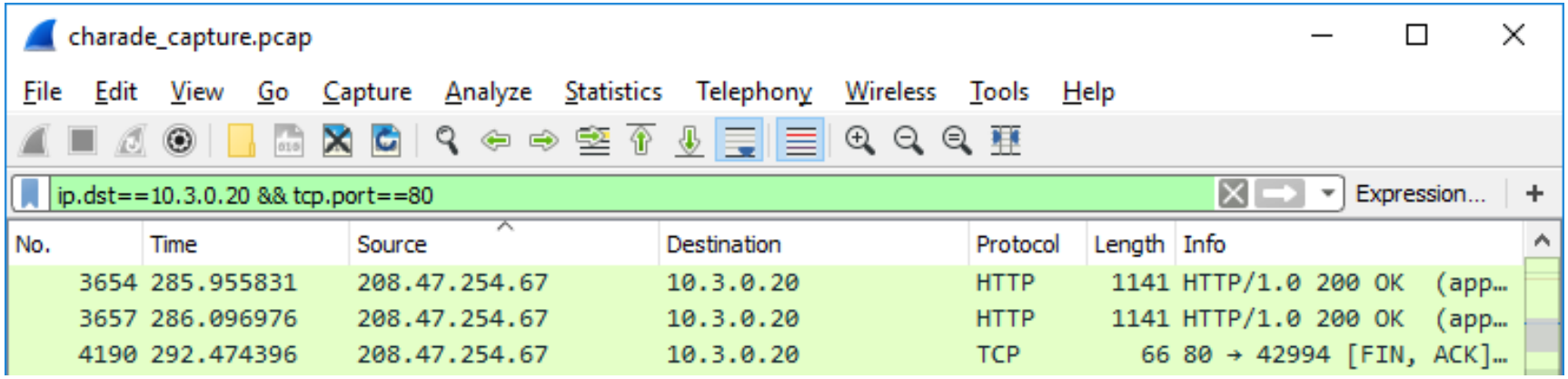
Fun with Wireshark

Basics

- Where to get it?
  - https://www.wireshark.org/download.html
  - Don't just Google and download – you'll end up with adware and other crap you don't need/want
- What is it?
  - Full-featured tool for network traffic capture and protocol analysis
  - Easy to use, but lots of advanced features
  - Started in 1998 with Ethereal (credit to Gerald Combs)
- Features
  - Free and open source! Distributed under GNU GPLv2
  - Extensible – write your own plug-ins and protocol dissectors

Currently on Version 2.0.X

Wireshark Display Filters



- Enter filters in textbox
  - Use Expression button to get help creating filters
  - Filter box is green for valid filter, red otherwise
- Click Apply to apply filter
- Click Clear to clear filter

More Display Filters . . .

- **Boolean Expressions in Filters:**
  - The symbol for logical **AND** in TCP filters is **&&** (you can use **and** and **&&** interchangeably)
  - The symbol for logical **OR** is **||** (you can use **or** and **||** interchangeably)
  - Use parenthesis to form more specific Boolean expressions
  - Wireshark generally doesn't care about case except with matching a specific string value.
- Some Examples:

Packets from 192.168.1.1	ip.src==192.168.1.1
Packets to and from port 80	tcp.port==80
From 10.10.3.2 to 10.10.3.40	ip.src==10.10.3.2 && ip.dst==10.10.3.40
To/from 10.10.3.2 on port 443	ip.addr==10.10.3.2 && tcp.port==443

More Example Filters

Filter	Description
`HTTP`	All HTTP protocol packets
`FTP`	All FTP protocol packets
`TCP`	All TCP packets
`ip.src == 192.168.1.1`	All packets with a source IP address of 192.168.1.1
`ip.dst == 10.10.1.1`	All packets with a destination address of 10.10.1.1
`ip.addr == 172.14.5.5`	All packets with 172.14.5.5 in source or destination address
`ip.src == 192.168.1.1 && ip.dst == 10.10.1.1`	All packets with source IP 192.168.1.1 and destination IP 10.10.1.1
`tcp.port == 80`	All TCP packets going to or coming from port 80
`ip.dst == 10.10.4.1 && tcp.port == 80`	All packets destined for port 80 on host 10.10.4.1
`tcp contains "www.cnn.com"`	All tcp packets containing the string "www.cnn.com"

Real-time vs. Stored Traffic

- Real-time traffic analysis
  - Tcpdump and Wireshark can capture and display traffic in real time.
  - Must have ***root*** or ***Administrator*** permissions to put interface in promiscuous mode.
  - Must use careful filtering to make sense of network traffic – way too much for real-time analysis
- Packet captures (.pcap)
  - Tcpdump can efficiently capture packets and write to .pcap file for later analysis.
  - Must have root/Admin rights to capture traffic, but not required to load captures in tools.
  - Allows for more detailed analysis of traffic; however, incidents may not be identified until long after the fact.

Stored Packet Capture Options

- Full packet capture
  - Best possible scenario, but not always possible
  - Must have network device that can ‘keep up’ with traffic
  - Must have significant storage space (possible to ‘rotate’ captures and age-out old files)
  - Obvious privacy concerns (depending on the context)
- Headers only
  - TCP/IP headers provide lots of data for forensic investigators; but payload data is also very useful.
  - Tcpdump’s ‘snaplen’ option allows configuration of amount of per-packet data to grab
- Filtered packet capture
  - To limit overhead, may want to only capture traffic to/from certain devices on the network; filtering can achieve this.

Capture Filters: Berkeley Packet Filter (BPF) syntax

- Wireshark capture is based on libpcap, developed in the 1980's at the Lawrence Berkeley Laboratory
- Capture filter language uses Berkeley packet filter (BPF)
  - An expression consists of one or more primitives.
  - Primitives usually consist of an id (name or number) preceded by one or more qualifiers
    - Type: *host, net, port, portrange*, etc. (if no qualifier, "*host*" is assumed)
    - Direction: *src, dst, src or dst* (if no qualifier, "*src or dst*" is assumed)
    - Protocol: *ether, wlan, ip, ip6, arp, rarp, tcp, udp*, etc.
  - Primitives can be combined using logical syntax: *and, or*
  - Examples:
    - dst host 192.168.1.15
    - src 192.168.1.33 and dst port 80

For more BPF syntax: http://biot.com/capstats/bpf.html

Packet Coloring

Wireshark · Coloring Rules · Default

Name	Filter
Bad TCP	tcp.analysis.flags && !tcp.analysis.window_update
HSRP State Change	hsrp.state != 8 && hsrp.state != 16
Spanning Tree Topology Change	stp.type == 0x80
OSPF State Change	ospf.msg != 1
ICMP errors	icmp.type eq 3 || icmp.type eq 4 || icmp.type eq 5 || icmp.type eq 11 || icmpv6.type eq 1 || icmpv6.type eq 2 || icmpv6.type eq 3 || icmpv
ARP	arp
ICMP	icmp || icmpv6
TCP RST	tcp.flags.reset eq 1
SCTP ABORT	sctp.chunk_type eq ABORT
TTL low or unexpected	(! ip.dst == 224.0.0.0/4 && ip.ttl < 5 && !pim && !ospf) || (ip.dst == 224.0.0.0/24 && ip.dst != 224.0.0.251 && ip.ttl != 1 && !(vrrp || ca
Checksum Errors	eth.fcs_bad==1 || ip.checksum_bad==1 || tcp.checksum_bad==1 || udp.checksum_bad==1 || sctp.checksum_bad==1 || mstp.checksur
SMB	smb || nbss || nbns || nbipx || ipxsap || netbios
HTTP	http || tcp.port == 80 || http2
IPX	ipx || spx
DCERPC	dcerpc
Routing	hsrp || eigrp || ospf || bgp || cdp || vrrp || carp || gvrp || igmp || ismp
TCP SYN/FIN	tcp.flags & 0x02 || tcp.flags.fin == 1
TCP	tcp
UDP	udp
Broadcast	eth[0] & 1

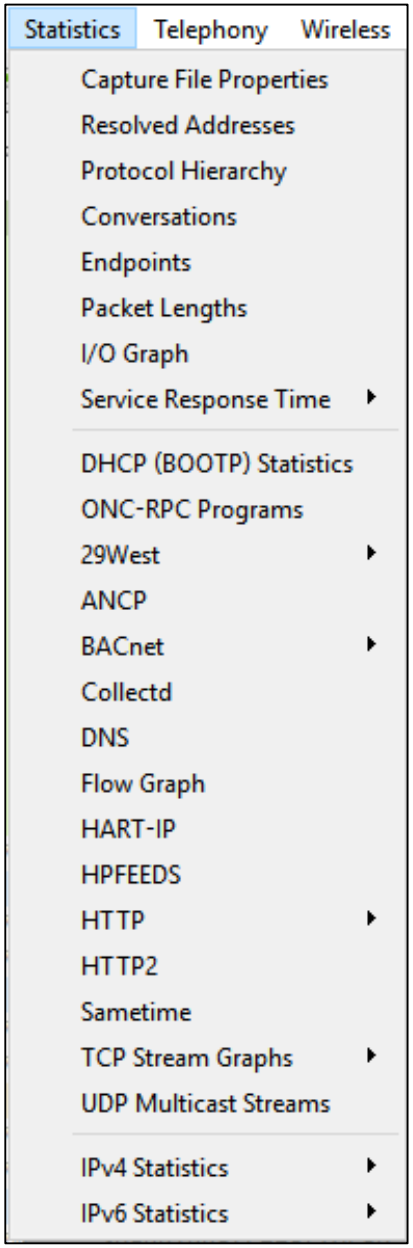
Double click to edit. Drag to move. Rules are processed in order until a match is found.

OK | Cancel | Import... | Export... | Help

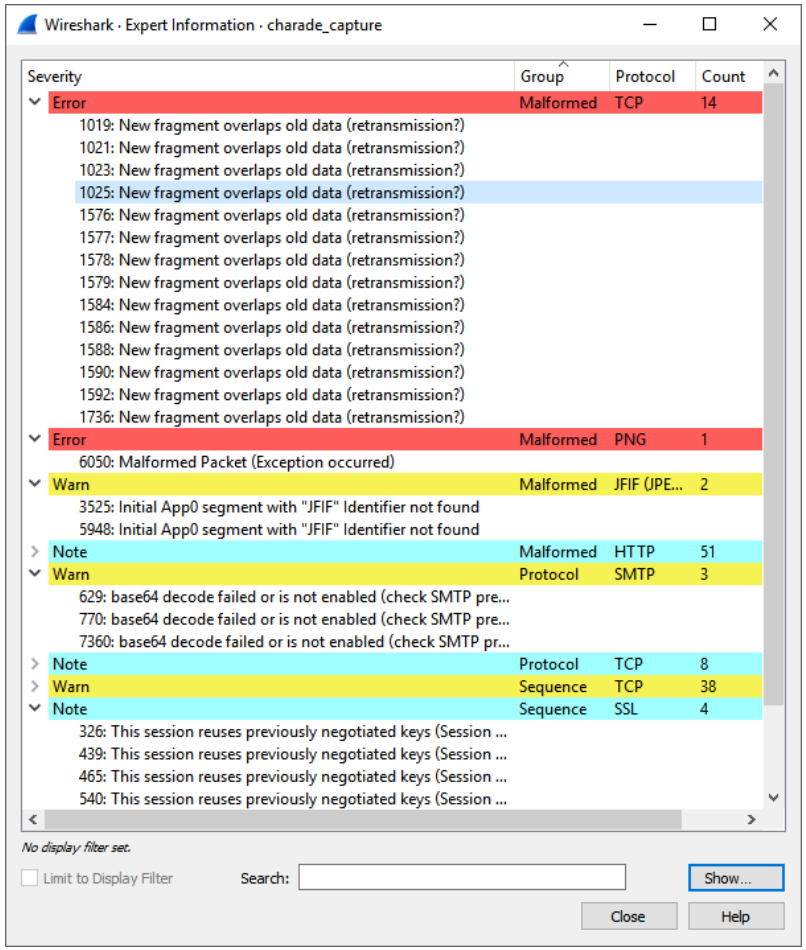
- Menu: View→Colorizing Rules …

Capture Statistics

- From Statistics menu item
- Capture File Properties
  - Start/stop time, number packets, OS, interface, etc.
- Resolved Addresses
  - All hostnames and resolved IPs in capture
- Protocol Hierarchy – percentage of packets by protocol
- Conversations
  - Source/destination IP, #packets, #bytes, start/end time
- Graphs
  - TCP/UDP flows,



Expert Information



- Analyze→Expert Info
- List of anomalies found in capture file
- Quick and easy way to show ‘uncommon’ network behavior
  - Useful in troubleshooting AND in identifying intentionally malformed packets
- Packets are grouped by severity and by anomaly type

Expert Info (More)

Severity	Group	Protocol	Count
Error	Malformed	TCP	14
Error	Malformed	PNG	1
Warn	Sequence	TCP	38
Warn	Protocol	SMTP	3
Warn	Malformed	JFIF (JPEG) image	2
Note	Sequence	SSL	4
Note	Sequence	TCP	178
Note	Protocol	TCP	8
Note	Malformed	HTTP	51
Chat	Sequence	TCP	905
Chat	Sequence	HTTP	748

- Severity
  - Error (red): serious problem, e.g. malformed packet
  - Warn (yellow): warning, e.g. app returned ‘unusual’ error code
  - Note (cyan): notable, e.g. an app returned ‘usual’ error code
  - Chat (blue?): info about usual workflow (TCP stream start/stop)
- Group
  - Checksum: invalid checksum
  - Sequence: non-contiguous or repeat
  - Response code: unusual response code
  - Undecoded: dissector incomplete or otherwise insufficient
  - Reassemble: problems with reassembly
  - Malformed: packet malformed or can’t be interpreted

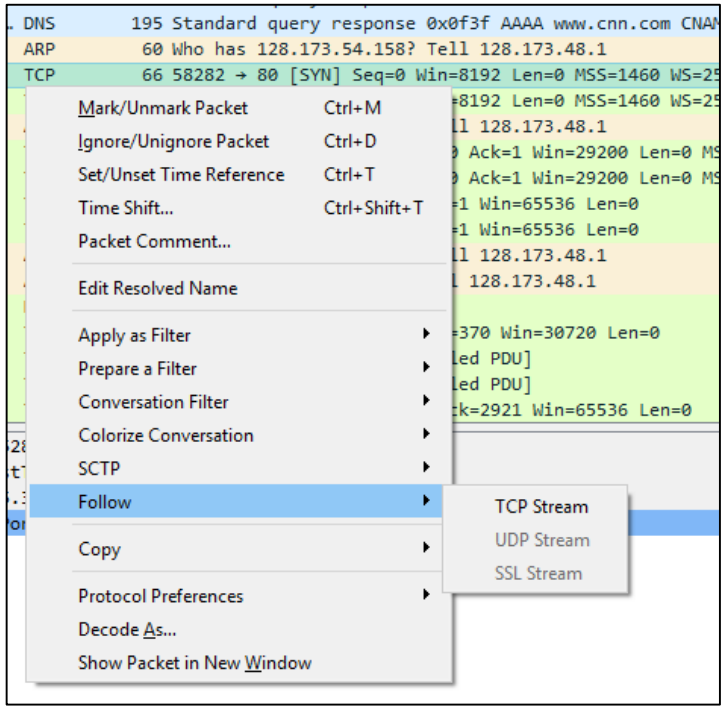
Hands on

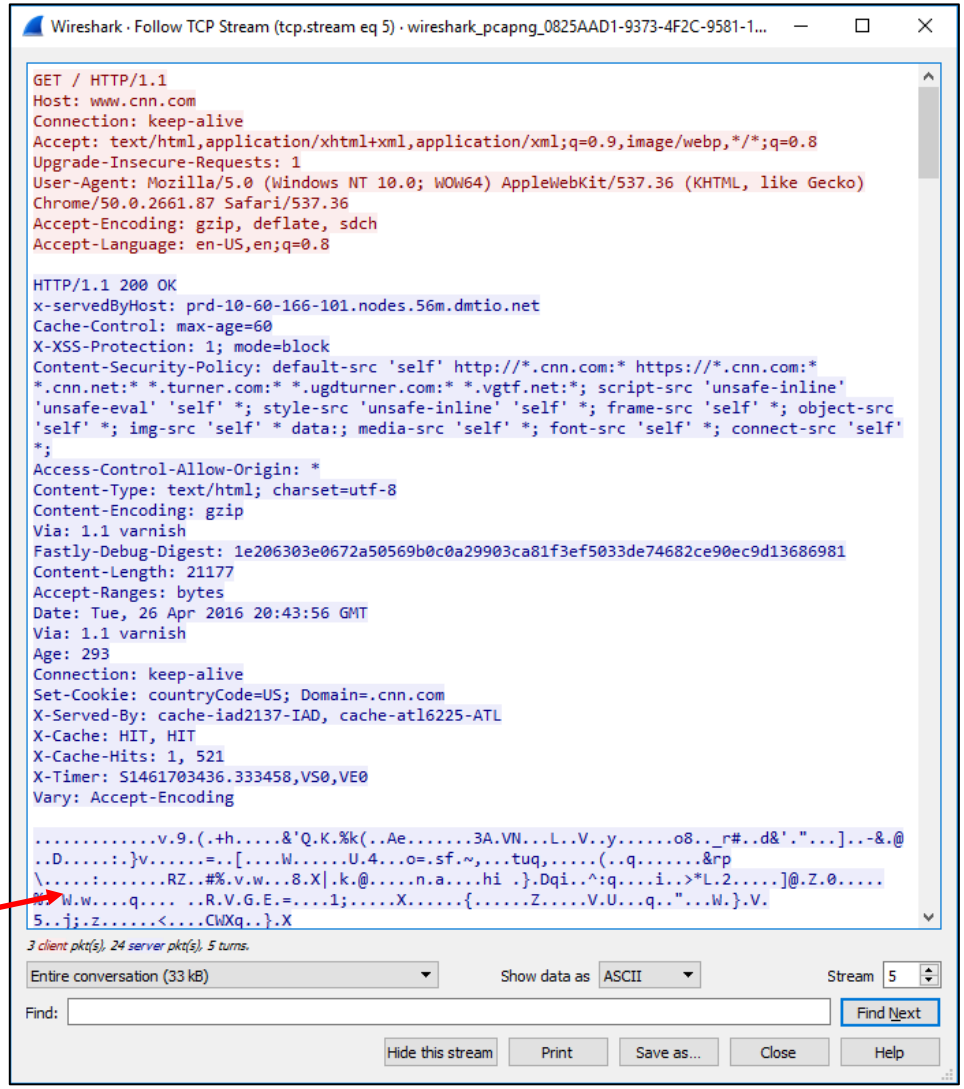
- Download PCAPs:
  - http://www.dnomyard.com/pcap/pcaps.zip

- Start with ftp_attack.pcap
  - Can you find FTP control and data traffic?
  - Who logged on to the FTP server?
  - What files did they download?
  - Can you identify the chain of events and what was exfiltrated?

Intermediate Features

- Follow TCP Streams



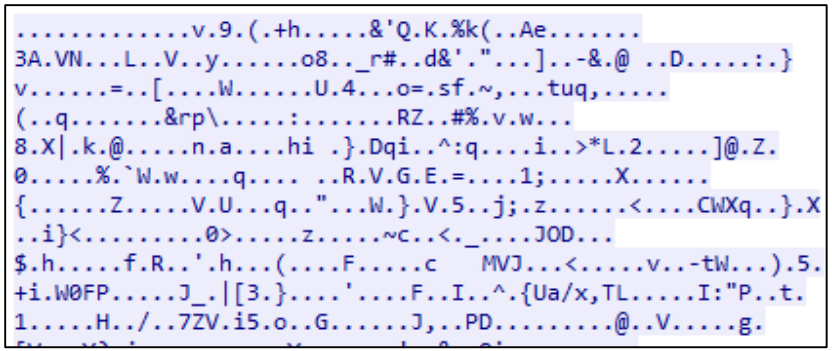


Zipped!

Expanding Compressed Content

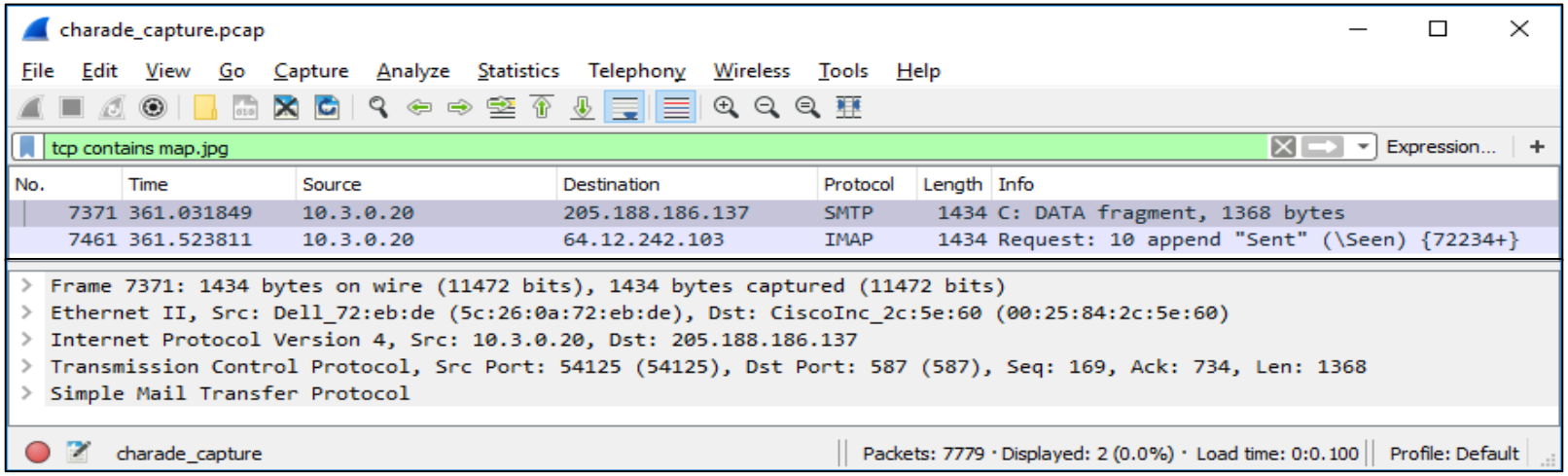
1. Follow TCP Stream
2. Note name of object in ‘TCP Stream’ window
3. From main window, choose

   File → Export Objects → HTTP
4. Select the object (find filename from Step 2)
5. Choose save, then give filename with appropriate extension



Basic File Carving

- If you want to follow along . . .
  - Open **charade_capture.pcap**
  - Find TCP stream containing ‘map.jpg’
    - In filter bar: “**tcp contains map.jpg**”
  - Follow TCP Stream
    - (Take a look at the TCP stream - what is going on?)



Magic Numbers

- File types are identified by a series of bytes at the beginning and/or end of the file – often called a 'magic number' or file signature
- Usually represented in Hex (as in below examples)

File type	Magic number
Compiled Java class files	CA FE BA BE
GIF images	47 49 46 38 39 61 ("GIF89a")
JPG images	FF D8 (start) FF D9 (end)
Unix/Linux scripts	25 21 (#!) followed by path to interpreter
ELF Executables	7F E L F
MS PE (executable) files	4D 5A (MZ initials for Mark Zbikowsi)

Lots more at https://en.wikipedia.org/wiki/Magic_number_(programming)

Format Conversion

```
--------------070302050006050209080901
Content-Type: image/jpeg;
 name="map.jpg"
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
 filename="map.jpg"

/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDABALDA4MChAODQ
PTo0OThBSV5QQUVZRjg5Um9TWWFkaWppP09ze3Jmel5naW

ZWVlZWVlZWVlZWVlZWVlZWVlZWVlZWVlZWVlZWVlZW
CAH9ApkDASIAAhEBAxEB/8QAGwAAAgMBAQEAAAAAAAAAA
BAMEBgQLBQcDAgcBAgMABBESIQUTMUEiUWEUcYGRBjJSob
otJDRGOTo8LxJYOENTaUByZkdMOz/8QAGQEBAQEBAQEAAA
```

Copy/paste from TCP stream

www.asciitohex.com

BASE64

```
/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDABALDA4
MChAODQ4SERATGCkbGBYWGDIkJh4pOzQ+
PTo0OThBSV5QQUVZRjg5Um9TWWFkaWppP09ze3
Jmel5naWX/2wBDARESEhgVGDAbGzBlQzlD

ZWVlZWVlZWVlZWVlZWVlZWVlZWVlZWVlZWVlZWVl
ZWVlZWVlZWVlZWVlZWVlZWX/wAAR
CAH9ApkDASIAAhEBAxEB/8QAGwAAAgMBAQEAA
```

Convert | Copy to Clipboard

"Convert"

Hexadecimal

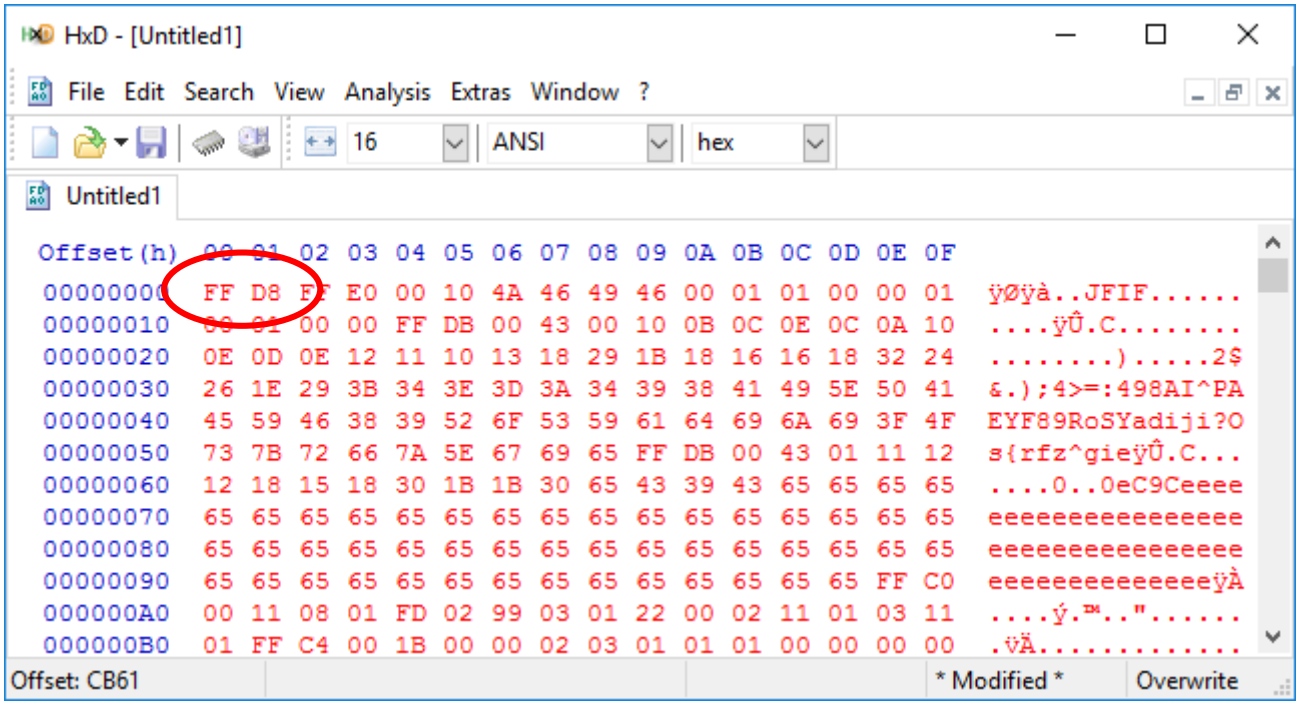
```
50 63 51 25 dc e0 9f da e9 d8 ec 05 38 92 26 02 90
41 03 7d a9 14 8b 5c f2 36 a2 08 91 ba 7b e8 8b 71
24 53 6e 43 75 1b 8a 96 35 29 c3 8e a0 82 3d 28
64 82 db 1a a2 5e 17 46 cc 6a 0e 36 23 b7 4a a2 c8
5a 40 31 8a cd 8d 4a 60 81 db ad 70 6c 77 aa 17
20 81 55 2e 73 f9 d4 c5 1c 9c f9 55 83 64 6d 4a 06
27 af 7a 3a 8c 0c 83 42 2f 8c 9e 95 3e 55 cc 6d d6
b9 a4 54 1f ff d9
```

Convert | Copy to Clipboard

"Copy to Clipboard"

Conversion (cont)

- Open HxD or other hex editor
  - File → New
- Paste hex bytes
- “Save As” desired filetype and open file



Wireshark Utilities

- We'll discuss just a sampling – there are more!

tshark

- Command-line wireshark!
- Provides the command-line functionality of tcpdump/windump with protocol decoders of Wireshark!

draymond — -bash — 82×56

```
Ds-MacBook-Pro:~ draymond$ tshark -h
TShark (Wireshark) 2.0.2 (v2.0.2-0-ga16e22e from master-2.0)
Dump and analyze network traffic.
See https://www.wireshark.org for more information.

Usage: tshark [options] ...

Capture interface:
  -i <interface>           name or idx of interface (def: first non-loopback)
  -f <capture filter>      packet filter in libpcap filter syntax
  -s <snaplen>             packet snapshot length (def: 65535)
  -p                       don't capture in promiscuous mode
  -I                       capture in monitor mode, if available
  -B <buffer size>         size of kernel buffer (def: 2MB)
  -y <link type>           link layer type (def: first appropriate)
  -D                       print list of interfaces and exit
  -L                       print list of link-layer types of iface and exit
```

tshark: Common Options

-h	Display help and exit
-D	Print list of interfaces
-i <interface>	Name of interface to capture on
-f <capture filter>	Capture filter in libpcap syntax
-Y <display filter>	Display filter in Wireshark syntax
-p	Don't use promiscuous mode
-I	Capture in monitor mode if available
-c <count>	Stop after 'count' packets
-r <infile>	Read from 'infile'
-w <outfile>	Write to 'outfile'
-F <outfile type>	Default = pcapng
-V	Verbose mode

tshark: More Options

-n	Disable all name resolution
-N <flags>	Specify name resolution (mntd)
-C <config profile>	Start with specified config profile
-a <autostop cond.>	duration:NUM - stop after NUM secs filesize:NUM - stop after NUM KB files:NUM - stop after NUM files
-O <protocols>	Only show specified protocols, comma separated
-x	Output hex and ASCII dump
-T fields	Format output
-E <fieldsoption>	Options for output fields/separators
-t a|ad|d|dd|e|r|u|ud	Output format for timestamps
-z <statistics>	Various statistics on capture

Aside: Configuration Files

preferences

```
# Configuration file for Wireshark 2.0.2.
#
# This file is regenerated each time preferences are saved within
# Wireshark. Making manual changes should be safe, however.
# Preferences that have been commented out have not been
# changed from their default value.

####### User Interface ########

# Open a console window (Windows only)
# One of: NEVER, AUTOMATIC, ALWAYS
# (case-insensitive).
#gui.console_open: NEVER
```

- Text files saved by Wireshark upon config changes; default configs are at
  - MacOS: /Users/<user>/.config/wireshark/*
  - Linux: /home/<user>/.config/wireshark/*
  - Windows: c:\Users\<user>\AppData\Roaming\Wireshark*
- Can be edited by hand in a text editor
- Use in **`tshark`** using **`-C`** flag and provide path to configs

tshark: Examples

- Standard capture (like **tcpdump -nnvi en0**)

```
# tshark -Vni en0
```

- Capture headers only to file (assumes wired, IPv4 headers!)

```
# tshark -ni en0 -s 54
```

- Capture and display DNS traffic only (Wireshark display filter syntax)

```
# tshark -ni en0 -Y 'udp.port==53 || tcp.port==53'
```

- Same as above using BPF syntax

```
# tshark -ni en0 -f 'port 53'
```

tshark: Read/Write Files

- Write to file

```
# tshark –i en0 –w outfile.pcap
```

- Write to file for 1 hour; write file until it reaches 1MB

```
# tshark –i en0 –w outfile.pcap –a duration:360
# tshark –i en0 –w outfile.pcap –a filesize:100
```

- Capture to ring buffer (5 files): 1 MB files

```
# tshark –i en0 –w file.pcap -b filesize:1000 -b files:5
```

- Read from file

```
# tshark –Vn –r infile.pcap
```

editcap

```
editcap [ -a <frame:comment> ] [ -A <start time> ]
  [ -B <stop time> ] [ -c <packets per file> ]
  [ -C [offset:]<choplen> ][ -E <error probability> ]
  [ -F <file format> ] [ -h ] [ -i <seconds per file> ]
  [ -o <change offset> ] [ -L ] [ -r ] [ -s <snaplen> ]
  [ -S <strict time adjustment> ] [ -t <time adjustment> ]
  [ -T <encapsulation type> ] [ -v ] infile outfile
  [ packet#[-packet#] ... ]
```

- Reads all (or specified) packets from infile, optionally mangles them, and writes them to outfile.
- Can be used to remove duplicates, ensure chronological ordering, shift time, anonymize captures, etc.

editcap Examples

- To see more detailed description of the options use:
  `editcap -h`
- To limit a capture file to packets from number 200 to 750 (inclusive) use:
  `editcap -r capture.pcap small.pcap 200-750`
- To get all packets from number 1-500 (inclusive) use:
  `editcap -r capture.pcap first500.pcap 1-500`
- To advance the timestamps of each packet forward by 3.0827 seconds:
  `editcap -t 3.0827 capture.pcap adjusted.pcap`
- To ensure all timestamps are in strict chronological order:
  `editcap -S 0 capture.pcap adjusted.pcap`
- To remove vlan tags from all packets within an Ethernet-encapsulated capture:
  `editcap -L -C 12:4 vlan.pcap no_vlan.pcap`

mergecap

mergecap [**-a**] [**-F** <*file format*>] [**-h**] [**-I** <*IDB merge mode*>]
[**-s** <*snaplen*>] [**-v**] [**-V**] **-w** <*outfile*>|- <*infile*> [<*infile*> ...]

- Combines multiple saved capture files into single output
- Reads .pcap from various utilities (tcpdump, windump, dumpcap, etc.)
- Can write to various output formats
  `# mergecap -F` for file formats
- Packets are merged according to timestamp (unless –a specified)

randcap

```
randpkt [ -b <maxbytes> ] [ -c <count> ]
     [ -t <type> ] <filename>
```

- Creates a pcap full of random packets
- Used to test packet sniffers and applications to see how they handle malformed packets
- Types include
  - Arp, bgp, dns, eth, fddi, ip, llc, m2m, sctp, syslog, tcp, udp, usb, tr, and more

Plug-ins

- If that's not enough, you can write your own packet dissectors in Lua or C!
- **C dissector**: build a full dev environment and grab the Wireshark source
  - Compiled into Wireshark; very efficient
- **Lua dissector**:
  - Loaded at startup and interpreted; less efficient
  - Details and examples on the Wireshark wiki at https://wiki.wireshark.org/Lua

More Hands-on?

- In **charade_capture.pcap**
  - Find the first DHCP request in the capture
    - Can you identify the manufacturer of the requesting system?
    - What is the lease duration?
  - What websites are visited?
    - Any indication that the user was looking for a new job?
  - Find the three emails from in5id3r.thr34t@aol.com
    - What is the name of the apparent owner of that alias?
    - Who are the recipients of the emails?
    - What is the address contained in the image file?

  - ***Any guesses what is going on here?***

Questions?