

"I wish Wireshark" - add the missing pieces with Lua



Chuck Craft
Wireshark Core

● `set_plugin_info(presenter_info)`

```
local presenter_info =  
{  
    version = "SF23US San Diego",  
    author = "Chuck Craft",  
    description = "Wireshark Core",  
    repository = "https://www.linkedin.com/in/cpu4coffee"  
}
```

(yep - that's a valid version string. Try it in your Lua code.)

● Example Lua tasks/solutions

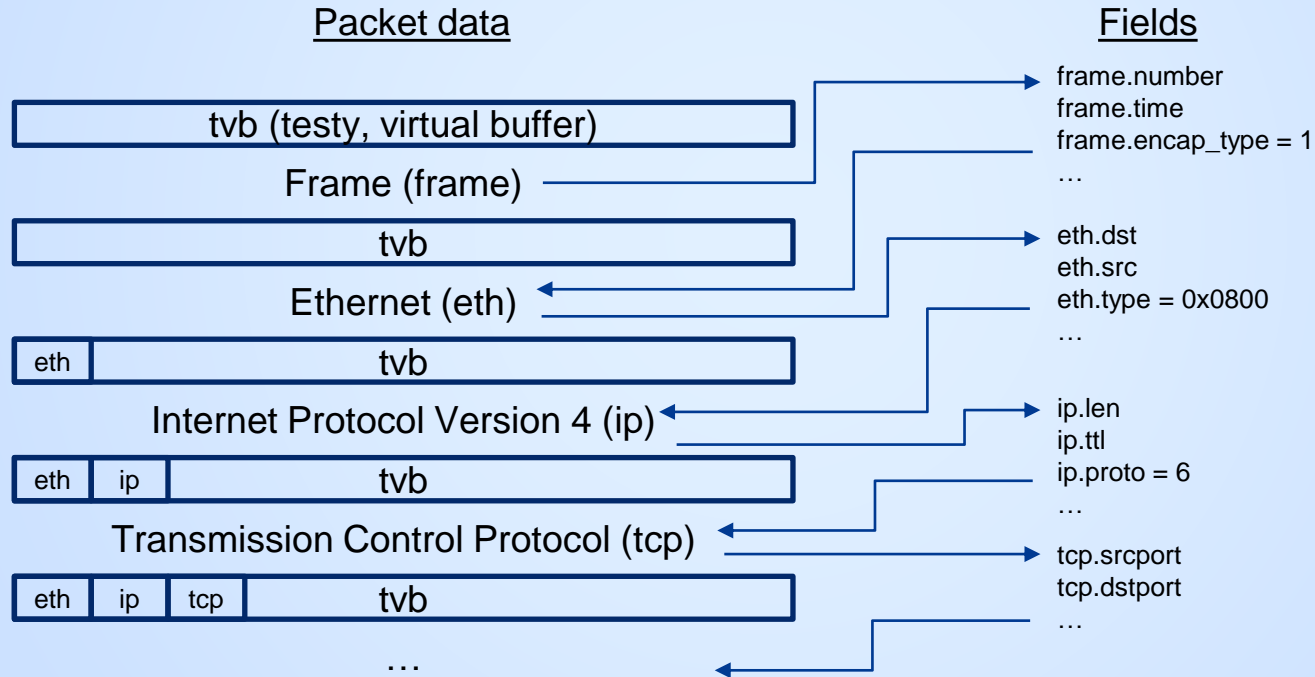
Dissectors

- An existing field in a different format
- New fields
- Dissecting an unsupported protocol

Taps/Listeners

- Relate data across multiple packets
- Custom statistics
- Add menu items/utilities

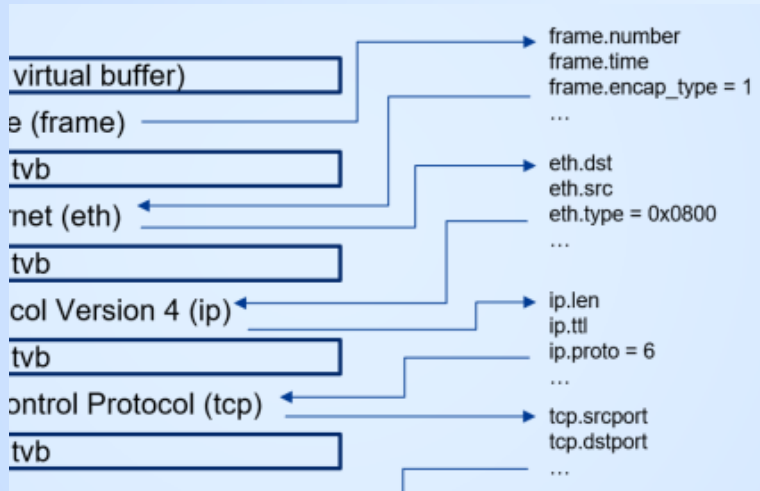
Dissector handoff



DRAFT



Lua - Dissector



Dissectors are meant to analyze some part of a packet's data. Only get called when the packet matches or when the user forces it using "Decode As".

View->Internals->Dissector Tables

wtap_encap: 1 = Ethernet
ethertype: 0x0800 = IPv4
ip.proto: 6 = TCP

DRAFT

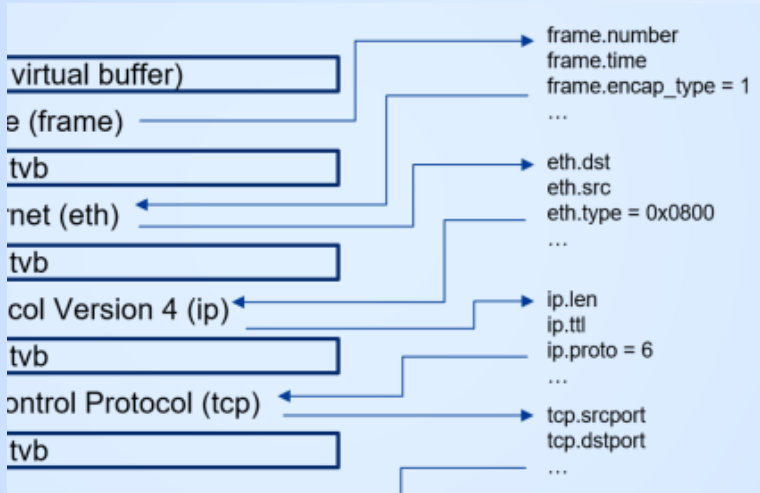
```
dissectortable:add(pattern, dissector)  
proto:register_heuristic(listname, func)
```

● Lua - Post-dissector

A dissector registered to be called after every other dissector has been called. These are handy as all protocol fields are available so they can be accessed, and they can add items to the dissection tree (Packet Details).

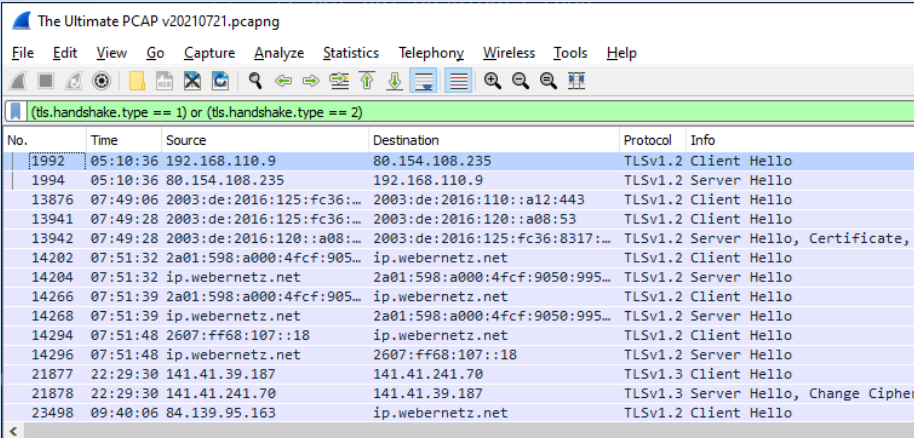
Not very efficient but easy to implement.

`register_postdissector(proto, [allfields])` DRAFT



Post-dissector runs after all other dissectors

Lua - Listener (Tap)



The Ultimate PCAP v20210721.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

(tls.handshake.type == 1) or (tls.handshake.type == 2)

No.	Time	Source	Destination	Protocol	Info
1992	05:10:36	192.168.110.9	80.154.108.235	TLSv1.2	Client Hello
1994	05:10:36	80.154.108.235	192.168.110.9	TLSv1.2	Server Hello
13876	07:49:06	2003:de:2016:125:fc36:...	2003:de:2016:110::a12:443	TLSv1.2	Client Hello
13941	07:49:28	2003:de:2016:125:fc36:...	2003:de:2016:120::a08:53	TLSv1.2	Client Hello
13942	07:49:28	2003:de:2016:120::a08:...	2003:de:2016:125:fc36:8317:...	TLSv1.2	Server Hello, Certificate, ...
14202	07:51:32	2a01:598:a000:4fcf:905...	ip.webernetz.net	TLSv1.2	Client Hello
14204	07:51:32	ip.webernetz.net	2a01:598:a000:4fcf:9050:995...	TLSv1.2	Server Hello
14266	07:51:39	2a01:598:a000:4fcf:905...	ip.webernetz.net	TLSv1.2	Client Hello
14268	07:51:39	ip.webernetz.net	2a01:598:a000:4fcf:9050:995...	TLSv1.2	Server Hello
14294	07:51:48	2607:ff68:107:18	ip.webernetz.net	TLSv1.2	Client Hello
14296	07:51:48	ip.webernetz.net	2607:ff68:107:18	TLSv1.2	Server Hello
21877	22:29:30	141.41.39.187	141.41.241.70	TLSv1.3	Client Hello
21878	22:29:30	141.41.241.70	141.41.39.187	TLSv1.3	Server Hello, Change Cipher
23498	09:40:06	84.139.95.163	ip.webernetz.net	TLSv1.2	Client Hello

local tap = Listener.new("tls", "(tls.handshake.type == 1) or (tls.handshake.type == 2)")

A `Listener` is called once for every packet that matches a certain filter or has a certain tap. It can read the tree, the packet's Tvb buffer as well as the tapped data, but it cannot add elements to the tree.

Called once every few seconds to redraw the GUI objects; in TShark this function is called only at the very end of the capture file.

DRAFT

Listener.new([tap], [filter], [allfields])

Existing field in a different format

- arp_host.lua
- EASYPOST.lua template



arp_host.lua

#sf23us

```
3 0.110617 cpe-24-166-172-1.kc.res.rr.com Broadcast ARP 60 24.166.173.161
4 0.211791 cpe-65-28-78-1.kc.res.rr.com Broadcast ARP 60 65.28.78.76
```

> Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface unknow
> Ethernet II, Src: Cisco251_af:f4:54 (00:07:0d:af:f4:54), Dst: Broadcast (ff:ff:ff:ff:f
v Address Resolution Protocol (request)
Hardware type: Ethernet (1)
Protocol type: IPv4 (0x0800)
Hardware size: 6
Protocol size: 4
Opcode: request (1)
Sender MAC address: cpe-65-28-78-1.kc.res.rr.com (00:07:0d:af:f4:54)
Sender IP address: cpe-24-166-172-1.kc.res.rr.com (24.166.172.1)
Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
Target IP address: cpe-24-166-173-159.kc.res.rr.com (24.166.173.159)

< Target IP address (arp.dst.proto_ipv4), 4 bytes

DRAFT

220703_arp-storm.pcapng (<https://wiki.wireshark.org/SampleCaptures#arp-rarp>)



arp_host.lua

“Is there a display filter that can be used to apply as column, the resolved or mapped host name for an ARP target IP address?

This string value is shown in the packet details window.”

<https://ask.wireshark.org/question/22016/resolved-or-mapped-arp-target-ip-address/>

DRAFT

arp_host.lua

Source field:

```
local arp_target_f = Field.new("arp.dst.proto_ipv4")
```

Destination field:

```
target_host = ProtoField.string("arp_host.target",  
                                "ARP target (resolved)")
```

Transformation:

```
subtree:add(pf.target_host, v.display)
```

WSLUARM: fieldinfo.display

“The string display of this field as seen in GUI.”

DRAFT

Download: <https://wiki.wireshark.org/Contrib> -> Post-Dissectors



EASYPOST.lua template

220703_arp-storm.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Info
1	09:01:05	Cisco251_af:f4:54	Broadcast	ARP	Who has 24.166.173.159? Tell 24.166.172.1
2	09:01:05	Cisco251_af:f4:54	Broadcast	ARP	Who has 24.166.172.141? Tell 24.166.172.1
3	09:01:05	Cisco251_af:f4:54	Broadcast	ARP	Who has 24.166.173.141? Tell 24.166.172.1

Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface unknown, id 0
Section number: 1
> Interface id: 0 (unknown)
Encapsulation type: Ethernet (1)
Arrival Time: Oct 5, 2004 09:01:05.275344000 Central Daylight Time
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 1096984865.275344000 seconds
[Time delta from previous captured frame: 0.000000000 seconds]
[Time delta from previous displayed frame: 0.000000000 seconds]
[Time since reference or first frame: 0.000000000 seconds]
Frame Number: 1
Frame Length: 60 bytes (480 bits)
Capture Length: 60 bytes (480 bits)
[Frame MD5 Hash: 2841a69b277550768a9e7bf77d72fe6a]
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: eth:ethertype:arp]
[Coloring Rule Name: ARP]
[Coloring Rule String: arp]
> Ethernet II, Src: Cisco251_af:f4:54 (08:07:0d:af:f4:54), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
> Address Resolution Protocol (request)
Important EASYPOST Protocol
EASYPOST data: ETH:ETHERTYPE:ARP
EASYPOST data (easypost.payload)

EASYPOST.lua is a starting point (cookbook steps) for Lua plugins.

Out of the box, it builds a new field – uppercase of `frame.protocols`.

<https://wiki.wireshark.org/lua#examples>

1. Download / save to
Personal Lua Plugins folder
2. Analyze -> Reload Lua Plugins

DRAFT

EASYPOST.lua template

Source field:

```
easypost_payload_f = Field.new("frame.protocols")
```

Destination field:

```
payload = ProtoField.string("easypost.payload", "EASYPOST data")
```

Transformation:

```
local field_data = string.format("%s", v):upper()  
subtree:add(pf.payload, field_data)
```

Programming in Lua: (<https://www.lua.org/pil/20.html>)
20 – The String Library

DRAFT

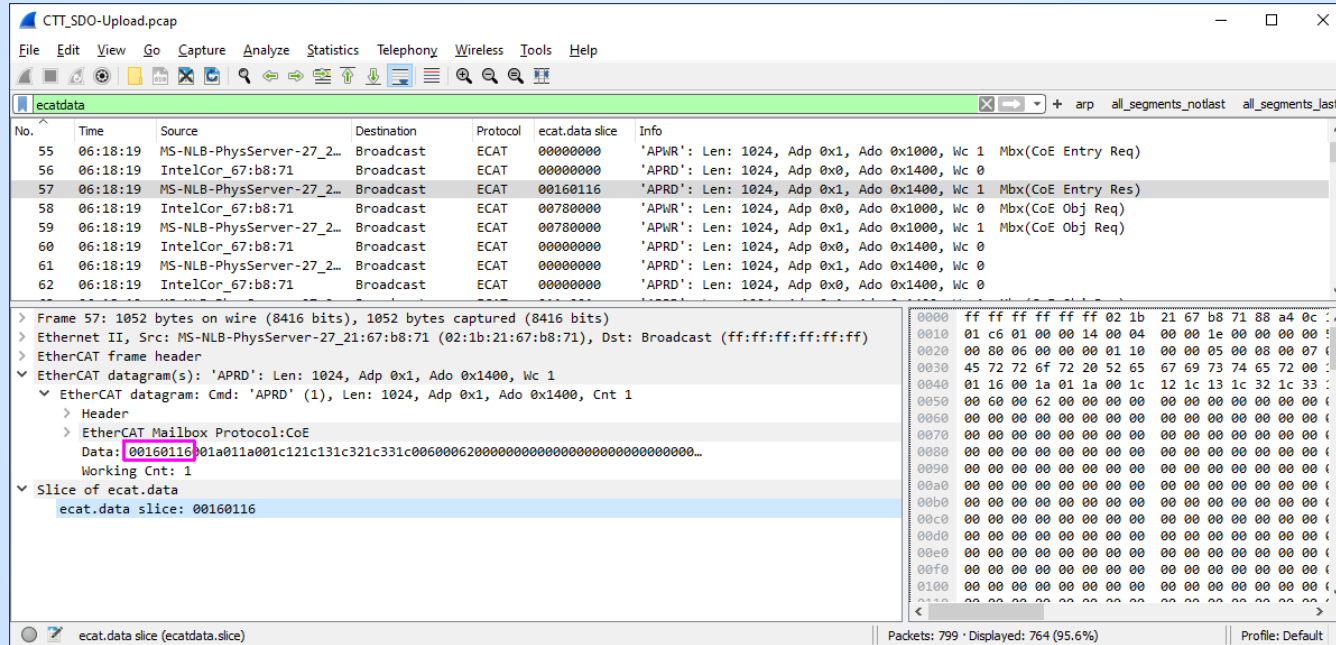
Download: <https://wiki.wireshark.org/lua#examples>



New fields

- ecatdata.lua -
<https://ask.wireshark.org/question/27207/how-to-display-slice-as-a-filter-in-column/>
- filtcols --
<https://wiki.wireshark.org/Lua/Examples/filtcols/>

ecatdata.lua



No.	Time	Source	Destination	Protocol	ecat.data slice	Info
55	06:18:19	MS-NLB-PhysServer-27_2...	Broadcast	ECAT	00000000	'APWR': Len: 1024, Adp 0x1, Ado 0x1000, Wc 1 Mbx(CoE Entry Req)
56	06:18:19	IntelCor_67:b8:71	Broadcast	ECAT	00000000	'APRD': Len: 1024, Adp 0x0, Ado 0x1400, Wc 0
57	06:18:19	MS-NLB-PhysServer-27_2...	Broadcast	ECAT	00160116	'APRD': Len: 1024, Adp 0x1, Ado 0x1400, Wc 1 Mbx(CoE Entry Res)
58	06:18:19	IntelCor_67:b8:71	Broadcast	ECAT	00780000	'APWR': Len: 1024, Adp 0x0, Ado 0x1000, Wc 0 Mbx(CoE Obj Req)
59	06:18:19	MS-NLB-PhysServer-27_2...	Broadcast	ECAT	00780000	'APWR': Len: 1024, Adp 0x1, Ado 0x1000, Wc 1 Mbx(CoE Obj Req)
60	06:18:19	IntelCor_67:b8:71	Broadcast	ECAT	00000000	'APRD': Len: 1024, Adp 0x0, Ado 0x1400, Wc 0
61	06:18:19	MS-NLB-PhysServer-27_2...	Broadcast	ECAT	00000000	'APRD': Len: 1024, Adp 0x1, Ado 0x1400, Wc 0
62	06:18:19	IntelCor_67:b8:71	Broadcast	ECAT	00000000	'APRD': Len: 1024, Adp 0x0, Ado 0x1400, Wc 0

> Frame 57: 1052 bytes on wire (8416 bits), 1052 bytes captured (8416 bits)
> Ethernet II, Src: MS-NLB-PhysServer-27_21:67:b8:71 (02:1b:21:67:b8:71), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
> EtherCAT frame header
EtherCAT datagram(s): 'APRD': Len: 1024, Adp 0x1, Ado 0x1400, Wc 1
 < EtherCAT datagram: Cmd: 'APRD' (1), Len: 1024, Adp 0x1, Ado 0x1400, Cnt 1
 > Header
 > EtherCAT Mailbox Protocol:CoE
 Data: 00160116001a011a001c121c131c321c331c00600062000000000000000000000000
 Working Cnt: 1
 Slice of ecat.data
 ecat.data slice: 00160116

```
0000 ff ff ff ff ff ff 02 1b 21 67 b8 71 88 a4 0c ...
0010 01 c6 01 00 00 14 00 04 00 00 1e 00 00 00 00 ...
0020 00 80 06 00 00 00 01 10 00 00 05 00 08 00 07 ...
0030 45 72 72 6f 72 20 52 65 67 69 73 74 65 72 00 ...
0040 01 16 00 1a 01 1a 00 1c 12 1c 13 1c 32 1c 33 ...
0050 00 60 00 62 00 00 00 00 00 00 00 00 00 00 00 ...
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
00a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
00b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
00c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
00d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
00e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
00f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
0100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
```

DRAFT

ecatdata.lua

‘I want to add a column with displaying 4 bytes form ethercat data: ecat.data[0:4]

For some reason filter "ecat.data[0:4]" is not work.

I found that filter "ecat.data[0:4] & 0xff" is works, but only if I use it as a normal filter.

I can't set this filter as cloumn.’

<https://ask.wireshark.org/question/27207/how-to-display-slice-as-a-filter-in-column/> ^{DRAFT}



ecatdata.lua

Source field:

```
ecatdata_f = Field.new("ecat.data")
```

Destination field:

```
payload = ProtoField.string("ecatdata.slice", "ecat.data slice")
```

Transformation:

```
local slicelen = 4
if (v.len < slicelen) then
    slicelen = v.len
end
local field_data = string.format("%s", v.range(0,slicelen))
tree:add(pf.payload, field_data)
```

WSLUARM: fieldinfo.range

“The TvbRange covering the bytes of this field in a Tvb
or nil if there is none.”

DRAFT

Download: [code](#) in [Ask question](#) [answer](#)/[comments](#)

filtcols

The screenshot shows a Wireshark interface with a packet capture of SMB2 traffic. The filter bar is set to 'filtcols.info contains file87.txt'. The packet list shows several SMB2 packets, with packet 178 selected. The packet details pane shows the structure of the SMB2 Create Request packet, including Ethernet II, Internet Protocol Version 4, Transmission Control Protocol, NetBIOS Session Service, and SMB2 (Server Message Block Protocol version 2). The Info column shows 'Create Request File: file87.txt'. The packet bytes pane shows the raw hex and ASCII data of the packet.

No.	Time	Source	Destination	Protocol	Length	Info
178	15.077438	10.3.1.2	10.3.1.1	SMB2	214	Create Request File: file87.txt
179	15.082181	10.3.1.1	10.3.1.2	SMB2	222	Create Response File: file87.txt
181	15.084746	10.3.1.2	10.3.1.1	SMB2	175	GetInfo Request FILE_INFO/SMB2_FILE_ALL_INFO File: file87.txt
183	15.092158	10.3.1.2	10.3.1.1	SMB2	158	Close Request File: file87.txt
185	15.099177	10.3.1.2	10.3.1.1	SMB2	254	Create Request File: file87.txt
186	15.103444	10.3.1.1	10.3.1.2	SMB2	222	Create Response File: file87.txt
187	15.106755	10.3.1.2	10.3.1.1	SMB2	175	GetInfo Request FILE_INFO/SMB2_FILE_INTERNAL_INFO File: file87.txt
189	15.114983	10.3.1.2	10.3.1.1	SMB2	183	Read Request Len:8192 Off:0 File: file87.txt
196	15.122438	10.3.1.2	10.3.1.1	SMB2	158	Close Request File: file87.txt
198	15.131615	10.3.1.2	10.3.1.1	SMB2	254	Create Request File: file87.txt
199	15.135157	10.3.1.1	10.3.1.2	SMB2	222	Create Response File: file87.txt
200	15.138475	10.3.1.2	10.3.1.1	SMB2	175	GetInfo Request FILE_INFO/SMB2_FILE_INTERNAL_INFO File: file87.txt
202	15.147432	10.3.1.2	10.3.1.1	SMB2	183	Read Request Len:4096 Off:0 File: file87.txt
207	15.153658	10.3.1.2	10.3.1.1	SMB2	1382	Write Request Len:4096 Off:0 File: file87.txt
211	15.161156	10.3.1.2	10.3.1.1	SMB2	158	Close Request File: file87.txt

```

> Frame 178: 214 bytes on wire (1712 bits), 214 bytes captured (1712 bits)
> Ethernet II, Src: a6:96:79:5e:31:ba (a6:96:79:5e:31:ba), Dst: 6e:a8:a8:d3:75:ec (6e:a8:a8:d3:75:ec)
> Internet Protocol Version 4, Src: 10.3.1.2, Dst: 10.3.1.1
> Transmission Control Protocol, Src Port: 56746, Dst Port: 445, Seq: 13117, Ack: 42588, Len:
> NetBIOS Session Service
> SMB2 (Server Message Block Protocol version 2)
  Protocol column: SMB2
  Info column: Create Request File: file87.txt
  
```

DRAFT

<https://wiki.wireshark.org/Lua/Examples/filtcols/>

filtcols

Reading from columns is “wonky”.

Cache values for later use after dissector pass 2.

Why does column check fail and return “(%s)” after first couple dissection passes?

https://gitlab.com/wireshark/wireshark/-/blob/master/epan/wslua/wslua_column.c#L122

<https://gitlab.com/wireshark/wireshark/-/blob/master/epan/wslua/wslua.h#L359>

DRAFT



filtcols

```
-- variables to persist across all packets
local pkt_data = {} -- indexed per packet

pkt_data.protocol = {}
pkt_data.info = {}
```

```
-- let's do it!
function filtcols_p.dissector(tvb,pinfo,tree)
    local cols_protocol = tostring(pinfo.cols.protocol)

    if cols_protocol ~= "(protocol)" then
        pkt_data.protocol[pinfo.number] = cols_protocol
    end

    local pkt_proto = pkt_data.protocol[pinfo.number]

    if pkt_proto ~= nil then
        tree:add(col_protocol_field, pkt_proto)
    end
end
```

DRAFT



Dissecting an unsupported protocol

- MYSTERY_PKT.lua
- Malfored_MyRoom.pcapng



MYSTERY_PKT.lua

```
4500 0034 8bfd 4000 8006 1068 c0a8 6e83
c0a8 6e8a 081a 01f6 41d2 eac6 e115 3ace
5018 fcc6 0032 0000 00d1 0000 0006 0103
0001 0001
```

Practical Packet Analysis, 3rd Edition - Chris Sanders
(<https://nostarch.com/packetanalysis3>)
Navigating a Mystery Packet 330

DRAFT



MYSTERY_PKT.lua

Add Ethernet header

No
Offsets

```
45 00 00 34 8b fd 40 00 80 06 10 68 c0 a8 6e 83
c0 a8 6e 8a 08 1a 01 f6 41 d2 ea c6 e1 15 3a ce
50 18 fc c6 00 32 00 00 00 d1 00 00 00 06 01 03
00 01 00 01
```

Add spaces between bytes

Import from Hex Dump...

Offsets: None

Encapsulation Type: Ethernet

Ethernet header - Ethertype (hex): 0800

DRAFT



MYSTERY_PKT.lua

```
45 00 00 34 8b fd 40 00 80 06 10 68 c0 a8 6e 83
```

```
c0 a8 6e 8a - IP header (20 bytes)
```

```
08 1a 01 f6 41 d2 ea c6 e1 15 3a ce 50 18 fc c6
```

```
00 32 00 00 - TCP header (20 bytes)
```

```
00 d1 00 00 00 06 01 03 00 01 00 01 - TCP payload DRAFT  
(12 bytes)
```




MYSTERY_PKT.lua

Source field:

```
mystery_payload_f = Field.new("tcp.payload")
```

Destination fields:

```
local pf = {  
  payload = ProtoField.string("mystery.payload", "Mystery Packet data"),  
  tid = ProtoField.uint16("mystery.tid", "Mystery - Transaction Identifier", base.HEX),  
  pid = ProtoField.uint16("mystery.pid", "Mystery - Protocol Identifier", base.HEX),  
  length = ProtoField.uint16("mystery.length", "Mystery - Length", base.HEX),  
  uid = ProtoField.uint8("mystery.uid", "Mystery - Unit Identifier", base.HEX),  
  fcode = ProtoField.uint8("mystery.fcode", "Mystery - Function Code", base.HEX),  
  variable = ProtoField.uint32("mystery.fcode", "Mystery - Remainder", base.HEX)
```

```
}
```

DRAFT



MYSTERY_PKT.lua

Transformation:

```
local field_data = string.format("%s", v):upper()
subtree:add(pf.payload, v.range, field_data)
subtree:add(pf.tid, v.range(0,2))
subtree:add(pf.pid, v.range(2,2))
subtree:add(pf.length, v.range(4,2))
subtree:add(pf.uid, v.range(6,1))
subtree:add(pf.fcode, v.range(7,1))
subtree:add(pf.variable, v.range(8,4))
```

WSLUARM: fieldinfo.range

“The TvbRange covering the bytes of this field in a Tvb or nil if there is none.”

DRAFT,



Malformed_MyRoom.pcapng

No.	Time	Source	Destination	Protocol	Length	Payload	Info
1	0.000000	local_ip	remote_ip	UDP	77	000000000003df590000001ddb9ad826f0000013000fef00a7c4f1234a4e8a201a0e	60652 → 13400 Len=35[Malformed Packet]
2	0.030947	remote_ip	local_ip	UDP	61	51000000130010ef00a7c4f1234a4e8a201a0e	13400 → 60652 Len=19[Malformed Packet]
3	6.004145	local_ip	remote_ip	UDP	77	000000000003df590000001ddb9ad826f0000013000f0b5da74bf1234a4e8a203283	60652 → 13400 Len=35[Malformed Packet]
4	6.035878	remote_ip	local_ip	UDP	61	510000001300100b5da74bf1234a4e8a203283	13400 → 60652 Len=19[Malformed Packet]
5	12.006304	local_ip	remote_ip	UDP	77	000000000003df590000001ddb9ad826f0000013000f2f043bb8f1234a4e8a202b15	60652 → 13400 Len=35[Malformed Packet]
6	12.035948	remote_ip	local_ip	UDP	61	510000001300102f043bb8f1234a4e8a202b15	13400 → 60652 Len=19[Malformed Packet]
7	18.004144	local_ip	remote_ip	UDP	77	000000000003df590000001ddb9ad826f0000013000fd713b92ff1234a4e8a2043a0	60652 → 13400 Len=35[Malformed Packet]
8	18.032389	remote_ip	local_ip	UDP	61	51000000130010d713b92ff1234a4e8a2043a0	13400 → 60652 Len=19[Malformed Packet]
9	24.013709	local_ip	remote_ip	UDP	77	000000000003df590000001ddb9ad826f0000013000f6e833640f1234a4e8a20743a	60652 → 13400 Len=35[Malformed Packet]
10	24.042595	remote_ip	local_ip	UDP	61	510000001300106e833640f1234a4e8a20743a	13400 → 60652 Len=19[Malformed Packet]

<https://discord.com/channels/889214182837321788/1007722817736945795/1082900474707050556> ^{DRAFT}

(Wireshark Discord - pcap-help - 03/07/2023 11:39 PM)



Malformed_MyRoom.pcapng

Chuckc 03/08/2023 10:24 AM

Looks like a six second heartbeat, echoing back the message sent by the client.

1	0.000000	00000000003df590000001ddb9ad826f0000013000f	ef00a7c4f1234a4e8a201a0e
2	0.030947	5100000130010	ef00a7c4f1234a4e8a201a0e
3	6.004145	00000000003df590000001ddb9ad826f0000013000f	0b5da74bf1234a4e8a203283
4	6.035878	5100000130010	0b5da74bf1234a4e8a203283
5	12.006304	00000000003df590000001ddb9ad826f0000013000f	2f043bb8f1234a4e8a202b15
6	12.035948	5100000130010	2f043bb8f1234a4e8a202b15
7	18.004144	00000000003df590000001ddb9ad826f0000013000f	d713b92ff1234a4e8a2043a0
8	18.032389	5100000130010	d713b92ff1234a4e8a2043a0
9	24.013709	00000000003df590000001ddb9ad826f0000013000f	6e833640f1234a4e8a20743a
10	24.042595	5100000130010	6e833640f1234a4e8a20743a

<https://discord.com/channels/889214182837321788/1007722817736945795/1083062717692260382> ^{DRAFT}

(Wireshark Discord - pcap-help - — 03/08/2023 10:24 AM

● Tap/Listener vs Dissector

```
dissector:call tvb, pinfo, tree)
```

A dissector can update the tree but has no corresponding `draw()` to update the GUI.

```
tap.packet(pinfo, tvb, tapinfo)
```

A listener can't update the tree but has a `listener.draw()` that will be called once every few seconds to redraw the GUI objects.

● Relate data across multiple packets

- `tls_conversations.lua`



tls_conversations.lua

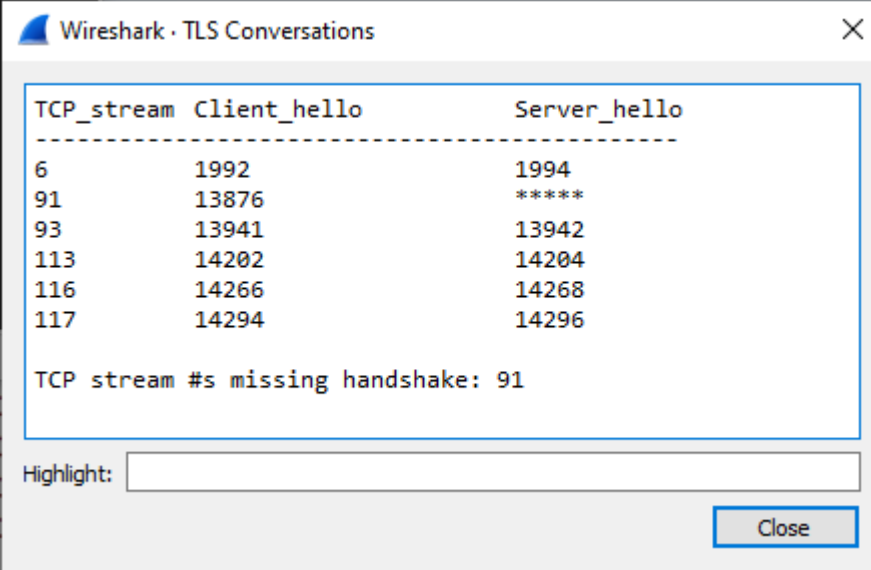
‘I see I can filter "tls.handshake.type == 1" for Client Hello and "tls.handshake.type == 2" for server hello.

I have server side capture and I want to filter all the TCP stream which has "Client Hello" but no "Server Hello" response back.’

<https://ask.wireshark.org/question/26618/filter-tls-with-no-server-hello/>

DRAFT

tls_conversations.lua



Wireshark · TLS Conversations

TCP_stream	Client_hello	Server_hello
6	1992	1994
91	13876	*****
93	13941	13942
113	14202	14204
116	14266	14268
117	14294	14296

TCP stream #s missing handshake: 91

Highlight:

Close

DRAFT

The Ultimate PCAP v20221220.pcapng (<https://weberblog.net/the-ultimate-pcap/>)
Screenshot of export trimmed to first 20000 packets



tls_conversations.lua

Source fields:

```
local tls_handshake_type_f = Field.new("tls.handshake.type")  
local tcp_stream_f = Field.new("tcp.stream")
```

Download: <https://wiki.wireshark.org/Contrib> -> Post-Dissectors

DRAFT

● Custom statistics

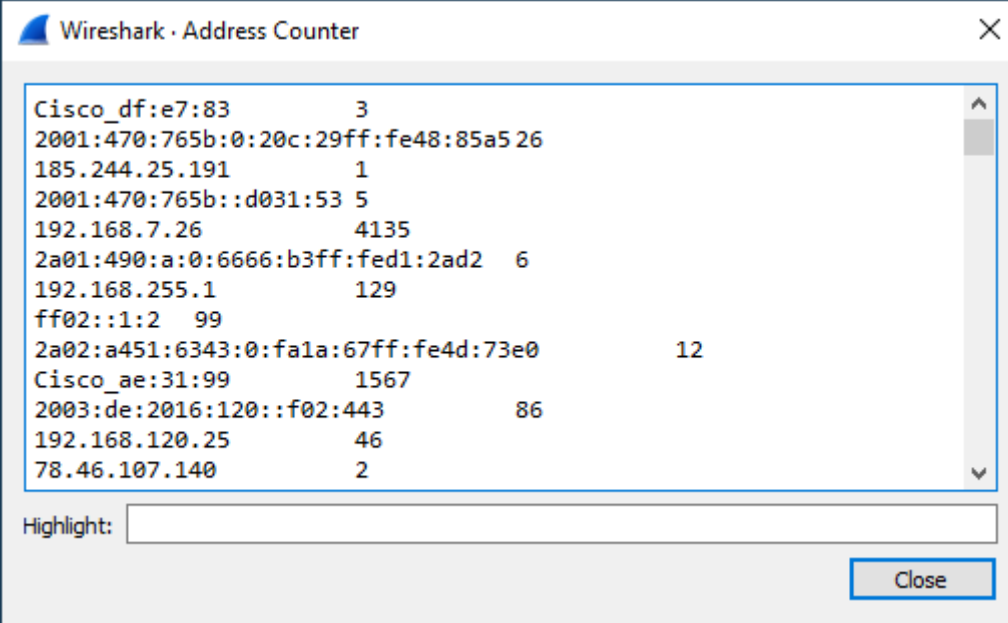
- WSDG - Address Counter
- WSDG - Address Counter (Sorted)
- QA Cafe (CloudShark) -
“How to write a Wireshark tap plugin in Lua”



address_counter.lua

- WSDG -
https://www.wireshark.org/docs/wsdg_html/#wslua_tap_example
- "This program will register a menu that will open a window with a count of occurrences of every address in the capture"

address_counter.lua



Wireshark · Address Counter

Cisco_df:e7:83	3	
2001:470:765b:0:20c:29ff:fe48:85a5 26		
185.244.25.191	1	
2001:470:765b::d031:53	5	
192.168.7.26	4135	
2a01:490:a:0:6666:b3ff:fed1:2ad2	6	
192.168.255.1	129	
ff02::1:2	99	
2a02:a451:6343:0:fa1a:67ff:fe4d:73e0		12
Cisco_ae:31:99	1567	
2003:de:2016:120::f02:443		86
192.168.120.25	46	
78.46.107.140	2	

Highlight:

Close

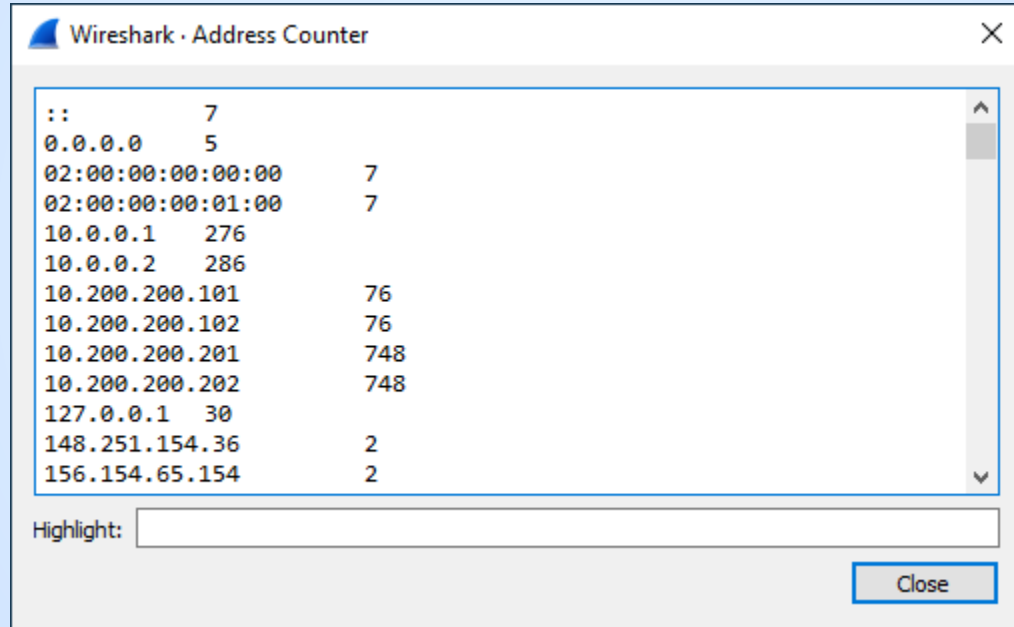
DRAFT

The Ultimate PCAP v20221220.pcapng (<https://weberblog.net/the-ultimate-pcap/>)
Screenshot of export trimmed to first 20000 packets

● address_counter_sorted.lua

- WSDG -
https://www.wireshark.org/docs/wsdg_html/#wslua_tap_example
- "This program will register a menu that will open a window with a count of occurrences of every address in the capture"

address_counter_sorted.lua



The screenshot shows the 'Wireshark - Address Counter' window. It contains a list of IP addresses and their corresponding counts. The list is sorted by count in descending order. Below the list is a 'Highlight:' field and a 'Close' button.

IP Address	Count
::	7
0.0.0.0	5
02:00:00:00:00:00	7
02:00:00:00:01:00	7
10.0.0.1	276
10.0.0.2	286
10.200.200.101	76
10.200.200.102	76
10.200.200.201	748
10.200.200.202	748
127.0.0.1	30
148.251.154.36	2
156.154.65.154	2

DRAFT

The Ultimate PCAP v20221220.pcapng (<https://weberblog.net/the-ultimate-pcap/>)
Screenshot of export trimmed to first 20000 packets

address_counter_sorted.lua

Print sorted output:

```
function tap.draw(t)
    tw:clear()
    table.sort(ips)
    for ip,num in pairsByKeys(ips) do
        tw:append(ip .. "\t" .. num .. "\n");
    end
end
```

Programming in Lua: (<https://www.lua.org/pil/19.3.html>)
19.3 – Sort

DRAFT

```
function pairsByKeys (t, f)
```



wifi-networks.lua

- <https://github.com/cloudshark/WiFi-Networks-Plugin>
- <https://www.qacafe.com/resources/how-to-write-a-wireshark-tap-plugin-in-lua/>
- Gui'fy ???



wifi-networks.lua

#sf23us

```
Command Prompt
C:\ tshark -q -r wifi-networks.pcapng
BSSID SSID Security Vendor Hidden Signal Noise SNR Channel
62:ab:eb:6c:fb:c0 "CloudShark" WPA2-Personal 62:ab:eb:6c:fb:c0 false -34 -74 40 11
4e:9d:08:53:d1:12 "ACME Corp" WPA2-Personal 4e:9d:08:53:d1:12 false -49 -74 25 11
de:dd:d7:51:ba:95 "The Neighbors" WPA2/3-Personal de:dd:d7:51:ba:95 false -41 -73 32 11
de:4e:cb:fe:62:d4 "ISEEYOURPACKETS" Open de:4e:cb:fe:62:d4 false -39 -76 37 11
3e:71:ce:4f:32:68 "Ye Olde Coffee Shop" WPA3-Personal 3e:71:ce:4f:32:68 false -41 -74 33 11
C:\
```

DRAFT

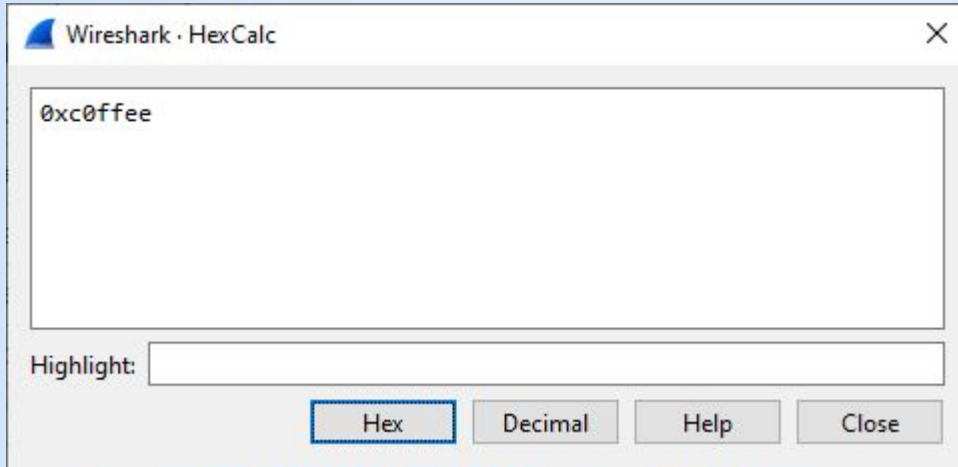
wifi-networks.pcapng - <https://www.cloudshark.org/captures/6d72d13108b3>

● Add menu items/utilities

- hexcalc.lua



hexcalc.lua



Download: <https://wiki.wireshark.org/Contrib> -> Other

<https://gitlab.com/wireshark/wireshark/-/issues/18386>
funnel/lua: closing child window disconnects buttons of parent

DRAFT

● hexcalc.lua

Document - `set_plugin_info(hexcalc_info)`

In GUI? - `if not gui_enabled() then return end`

Main function - open window, read text, output value.

```
local function hexcalc()
```

```
    local win = TextWindow.new("HexCalc")
```

```
    win:set_editable(true)
```

Add buttons -

```
        win:add_button("Help", function()
```

DRAFT

Create the menu entry -

```
    register_menu("Hex Calculator", hexcalc, MENU_TOOLS_UNSORTED)
```