

Duct tape and baling wire: Extending Wireshark with Lua



Chuck Craft

Your Company Here

set_plugin_info(presenter_info)

```
local presenter_info =  
{  
    version = "SF22US Kansas City",  
    author = "Chuck Craft",  
    description = "Wireshark contributor",  
    repository = "https://www.linkedin.com/in/cpu4coffee"  
}
```

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

Pre-req / Foundation Info

- <https://sharkfestus.wireshark.org/sf15>
 - wslua - who/what/why/... (Why not Python)
 - Walk through of test/lua/dissector.lua



Down the Rabbit Hole

- Finding an obscure or elegant solution in Wireshark is exciting but at a certain point the original question / problem needs to be solved.
- Pick a threshold of when to move on:
 - Time invested
 - Solution checklist of steps to try
 - Wireshark source code

A solid blue circle is positioned to the left of the section header.

#1 - ARP Target IP Address

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

“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.”

#1 - ARP Target IP Address

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

∨ 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

● #1 (cont)

- !!! Sample capture: 220703_arp-storm.pcapng
<https://wiki.wireshark.org/SampleCaptures>
- Add column for original field: `arp.dst.proto_ipv4`
- Resolve names enabled: `View->Name Resolution`
- What's in the Conversations/Endpoints?

● #1 (cont)

- Is there a better field to use?
 - Display Filter Reference
 - <https://www.wireshark.org/docs/dfref/a/arp.html>
 - `tshark -G fields | grep -i resolved | grep -i arp`
- Is there a Preference setting to be tweaked?
- Code spelunking (even deeper in the rabbit hole)
 - See the original Ask question

● Make the plugin already

- Have now spent more time looking for a solution than it will take to write/test the Lua plugin. (xkcd 627)
- Six easy steps to write a plugin (Templatized):
 - Step 1 - document as you go
 - Step 2 - create a protocol to attach new fields to
 - Step 3 - add field(s) to Step 2 protocol
 - Step 4 - Field extractor to copy packet field data
 - Step 5 - create the postdissector function that will run on each frame/packet
 - Step 6 - register the new protocol as a postdissector

Step 1 - Document

```
-- arp_host.lua
-- https://ask.wireshark.org/question/22016/resolved-or-mapped-arp-target-ip-address/
-- Sample capture: https://wiki.wireshark.org/SampleCaptures#arp-rarp - arp-storm.pcap

-- Step 1 - document as you go. See header above and set_plugin_info().
local arp_host_info =
{
    version = "1.0.0",
    author = "Good Coder",
    description = "Arp IP Target - resolved",
    repository = "Floppy in top drawer"
}

set_plugin_info(arp_host_info)
```

Step 2 - new Protocol

```
-- Step 2 - create a protocol to attach new fields to  
local arp_host_p = Proto.new("arp_host", "Arp IP Target - resolved")
```

- Analyze -> Reload Lua Plugins
- Analyze -> Enabled Protocols...

Step 3 - add results field

```
-- Step 3 - add some field(s) to Step 2 protocol
local pf = {
    target_host = ProtoField.string("arp_host.target", "ARP target (resolved)")
}
```

```
arp_host_p.fields = pf
```

- Analyze -> Reload Lua Plugins
- View -> Internals -> Supported Protocols

Step 4 - get original field

```
-- Step 4 - create a Field extractor to copy packet field data.  
local arp_target_f = Field.new("arp.dst.proto_ipv4")
```



Step 5 - “miracle occurs”

-- Step 5 - create the postdissector function that will run on each frame/packet

```
function arp_host_p.dissector(tvb,pinfo,tree)
    local subtree = nil

    -- copy existing field(s) into table for processing
    finfo = { arp_target_f() }

    if (#finfo > 0) then
        if not subtree then
            subtree = tree:add(arp_host_p)
        end
        for k, v in pairs(finfo) do
            -- process data and add results to the tree
            subtree:add(pf.target_host, v.display)
        end
    end
end
end
```

Step 6 - register dissector

```
-- Step 6 - register the new protocol as a postdissector  
register_postdissector(arp_host_p)
```

- Analyze -> Reload Lua Plugins
- Display Filter:
 arp_host.target matches "[a-z]"

● wsluarm

- Wireshark's Lua API documented in WSDG
https://www.wireshark.org/docs/wsdg_html/
- Generated by scripts from epan/wslua_* source
See doc/README.wslua
- wslua Index - available on Wiki
<https://wiki.wireshark.org/lua#wireshark-s-lua-api>



EASYPOST.lua

- A template file for a simple post dissector
<https://wiki.wireshark.org/lua#examples>

EASYPOST.lua - fields

```
-- Step 3 - add some field(s) to Step 2 protocol
local pf = {
    results = ProtoField.string("easypost.results", "EASYPOST results")
}

easypost_p.fields = pf

-- Step 4 - create a Field extractor to copy packet field data.
easypost_results_f = Field.new("frame.protocols")
```



EASYPOST.lua - results

```
finfo = { easypost_results_f() }

if (#finfo > 0) then
    if not subtree then
        subtree = tree:add(easypost_p)
    end
    for k, v in pairs(finfo) do
        -- process data and add results to the tree
        local field_data = string.format("%s", v):upper()
        subtree:add(pf.results, field_data)
    end
end
```

EASYPOST.lua - results

```
Frame Number: 1
Frame Length: 60 bytes (480 bits)
Capture Length: 60 bytes (480 bits)
[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: LexmarkP_83:76:2c (00:04:00:83:76:2c), Dst: Broadcast (ff:f
> Address Resolution Protocol (reverse request)
v Important EASYPOST Protocol
  EASYPOST results: ETH:ETHERTYPE:ARP ←
```

The Ultimate PCAP v20210130.pcapng

ws_expert.lua

- <https://gitlab.com/wireshark/wireshark/-/issues/15990>
“... create a display filter to display frames with more than one expert info:

```
count(_ws.expert.message)>4
```

Would be nice to be able to add a column `count(_ws.expert.message)` to sort on and have available when analyzing.”

ws_expert.lua - fields

```
-- Step 3 - add some field(s) to Step 2 protocol
local pf = {
    ws_count = ProtoField.uint8("ws_expert.count", "message count"),
    ws_string = ProtoField.string("ws_expert.string", "message string")
}

ws_expert_p.fields = pf

-- Step 4 - create a Field extractor to copy packet field data.
local ws_expert_message_f = Field.new("_ws.expert.message")
```



ws_expert.lua - results

```
finfo = { ws_expert_message_f() }

if (#finfo > 0) then
  if not subtree then
    subtree = tree:add(ws_expert_p)
  end
  subtree:add(pf.ws_count, #finfo)
  for k, v in pairs(finno) do
    -- process data and add results to the tree
    local field_data = string.format("%s", v):upper()
    subtree:add(pf.ws_string, field_data)
  end
end
end
```



ws_expert.lua - results

#sf22us

The Ultimate PCAP v20210130.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ws_expert.count >2

No.	Time	Source	Destination	Protocol	Length	Info
14...	331010751.174...	quadnine	193.24.227.230	DNS	428	[TCP Previous segment not capture
15...	331010869.245...	dns.quad9.net	2001:470:765b::b1...	TCP	78	[TCP Previous segment not capture
15...	331010871.254...	dns.quad9.net	2001:470:765b::b1...	DNS	468	[TCP Previous segment not capture
17...	332407574.036...	84.146.135.221	217.0.21.65	SIP	844	Status: 200 OK (BYE)
19...	332670851.558...	84.146.135.221	217.0.21.65	SIP	823	Status: 200 OK (BYE)
21...	332690522.370...	84.146.135.221	217.0.21.65	SIP	845	Status: 200 OK (BYE)
22	335431461.269	whois.nine.net	2001:470:765b::b1	WHOIS	166	[TCP Previous segment not capture

<

- > Frame 17456: 844 bytes on wire (6752 bits), 844 bytes captured (6752 bits) on interface unknown, id 41
- > Ethernet II, Src: AVMAudio_7e:33:a0 (c8:0e:14:7e:33:a0), Dst: JuniperN_50:d2:1a (3c:61:04:50:d2:1a)
- > 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 7
- > PPP-over-Ethernet Session
- > Point-to-Point Protocol
- > Internet Protocol Version 4, Src: 84.146.135.221 (84.146.135.221), Dst: 217.0.21.65 (217.0.21.65)
- > User Datagram Protocol, Src Port: 5060, Dst Port: 5060
- > Session Initiation Protocol (200)

ws_expert.message count

- message count: 3
- message string: UNRECOGNISED SIP HEADER (X-RTP-STAT)
- message string: UNRECOGNISED SIP HEADER (X-RTP-STAT-ADD)
- message string: UNRECOGNISED SIP HEADER (X-SIP-STAT)

message count (ws_expert.count)

How to display slice as a filter in column?

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