



# Wireshark Plugins for Pentesters

**Nishant Sharma & Jeswin Mathai**

[www.PentesterAcademy.com](http://www.PentesterAcademy.com)



What is it?

# What is it?

- **Collection of Wireshark plugins to perform**
  - Macro analysis
  - Providing summary or overview
  - Dissecting unknown protocols
  - Detecting attacks/threats
- **Covered protocols**
  - WiFi
  - DNS
  - DHCP
  - HTTP, HTTPS
- **GitHub:** <http://www.github.com/pentesteracademy/patoolkit>



# Index

- [Introduction to Wireshark Plugins](#)
- [Writing Hello World Listener](#)
- [Writing Hello World Dissector](#)
- [Heuristic Dissector](#)
- [PA Toolkit Demo](#)
- [Q & A](#)





# Introduction to Wireshark Plugins

# Motivation

- Macro analysis
- Custom/Proprietary protocols
- Scaling detection logic (i.e. automating detection)
- Wireshark available everywhere, well maintained, free

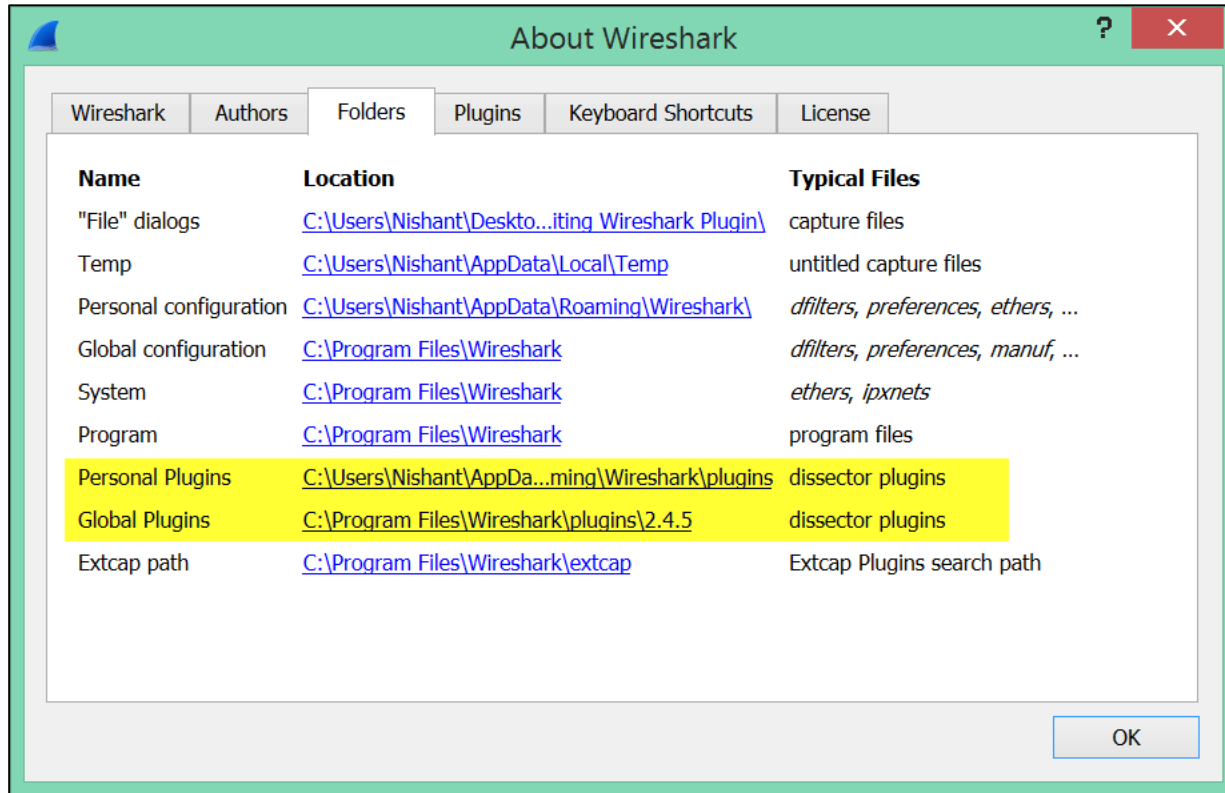
# Wireshark Plugins

- plugins for various purposes
- Plugins can be
  - Lua scripts
  - Compiled C/C++ code

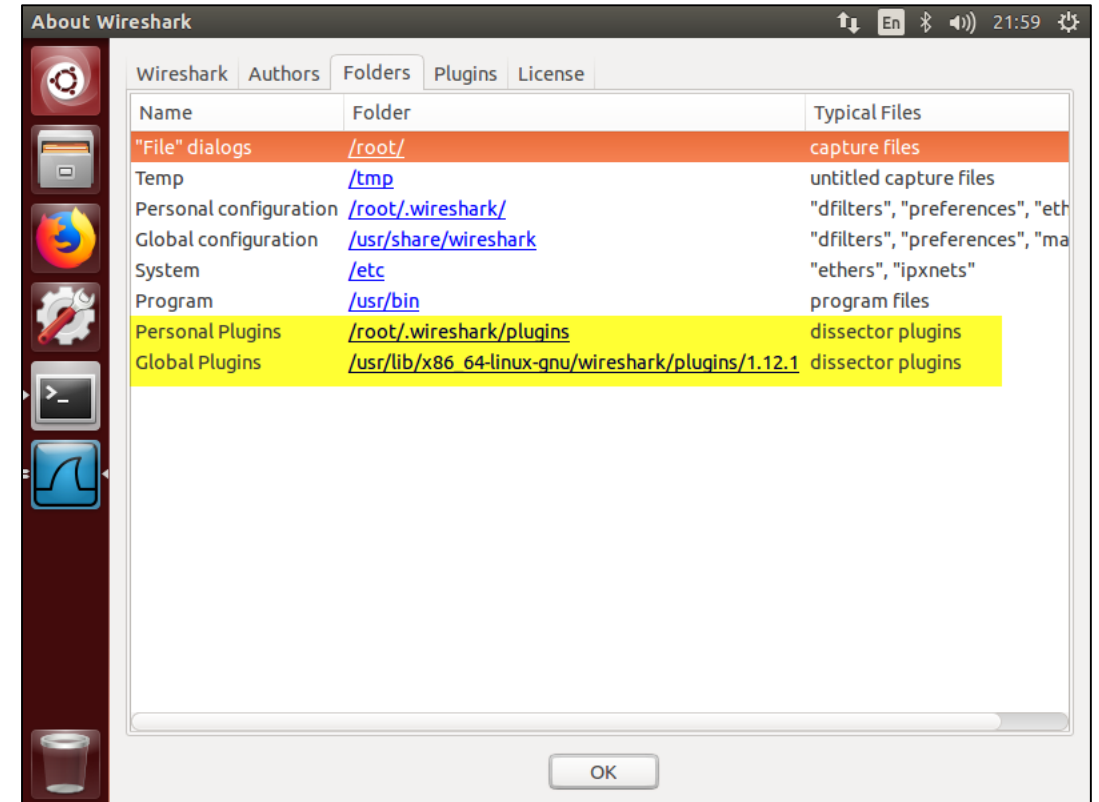
# Plugins locations

- Check Help > About Wireshark > Folders

## Windows

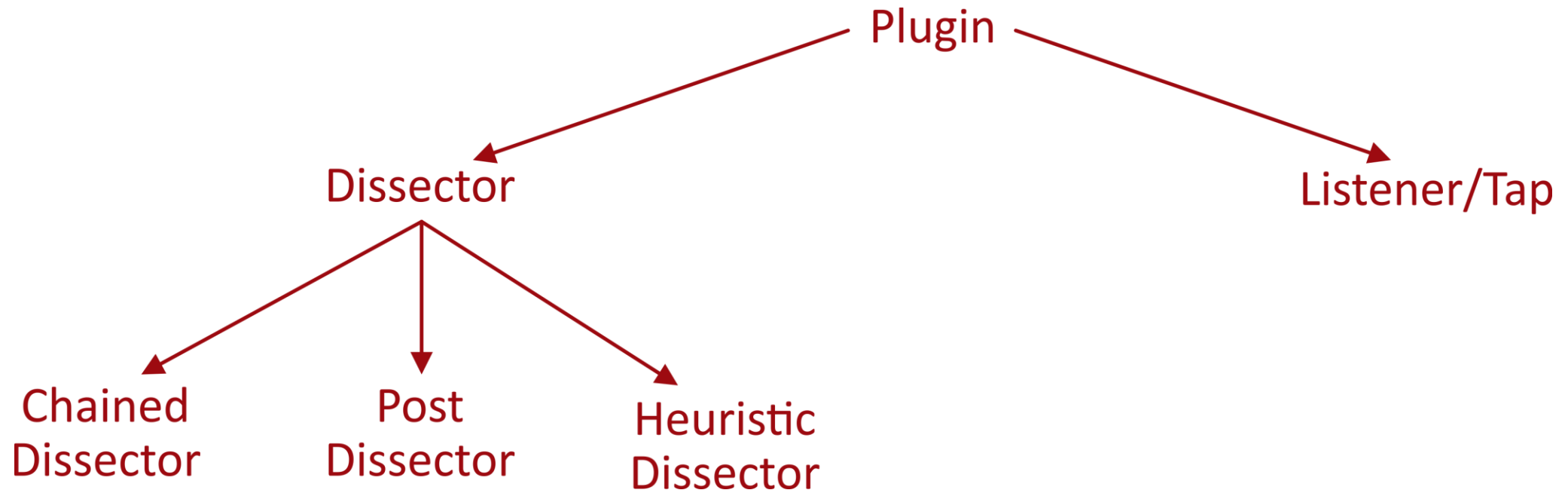


## Ubuntu





# Wireshark Plugins Types



# Dissector

- To interpret the payload data
- Decodes its part of the protocol and passes the payload to next

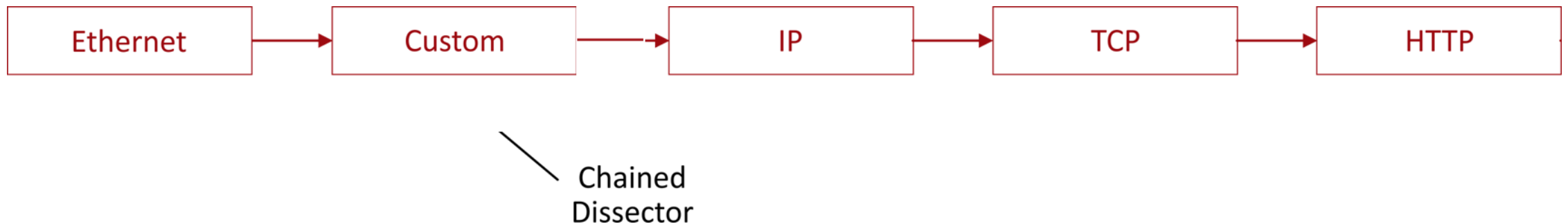
## Example Dissection Flow



# Chained Dissector

- Takes data from previous dissector, processes its part and pass the payload to next dissector

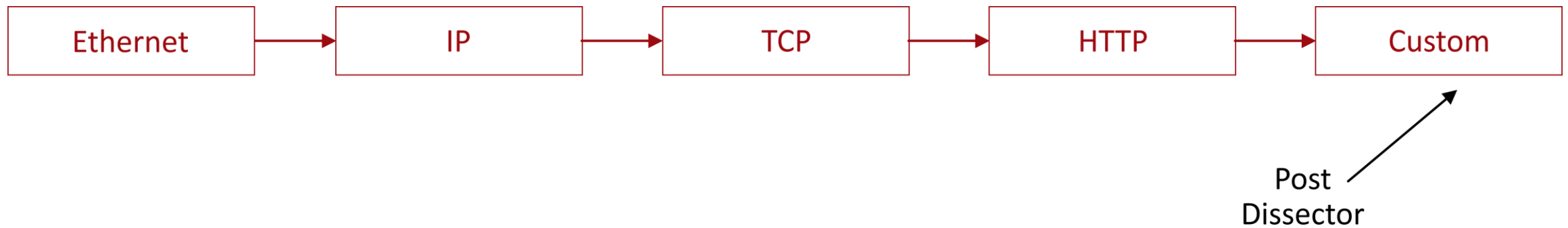
## Example Dissection Flow



# Post Dissector

- Last dissector in the dissector chain

## Example Dissection Flow





# Writing Hello World Listener

# Writing a Listener: Proposal

- A sample Hello World listener

## Why?

- Understanding various aspects
- Getting started with plugin development

# Writing a Hello World Listener

- **Lua File:** hello\_world\_listener.lua
- **Example PCAP:** Complete\_normal\_call.pcap

## Code Walkthrough

- Check if GUI is enabled. Done to make sure, it doesn't run and fail when tshark is used

```
-- If GUI is not enabled exit.  
if not gui_enabled() then return end
```

# Writing a Hello World Listener

- Main function to create pop up windows and showing content in it

```
-- Function to be called on selecting the option from Tools menu
local function dialog_menu()
    -- Create a new Window with "Hello World Title"
    local win = TextWindow.new("Hello World !");

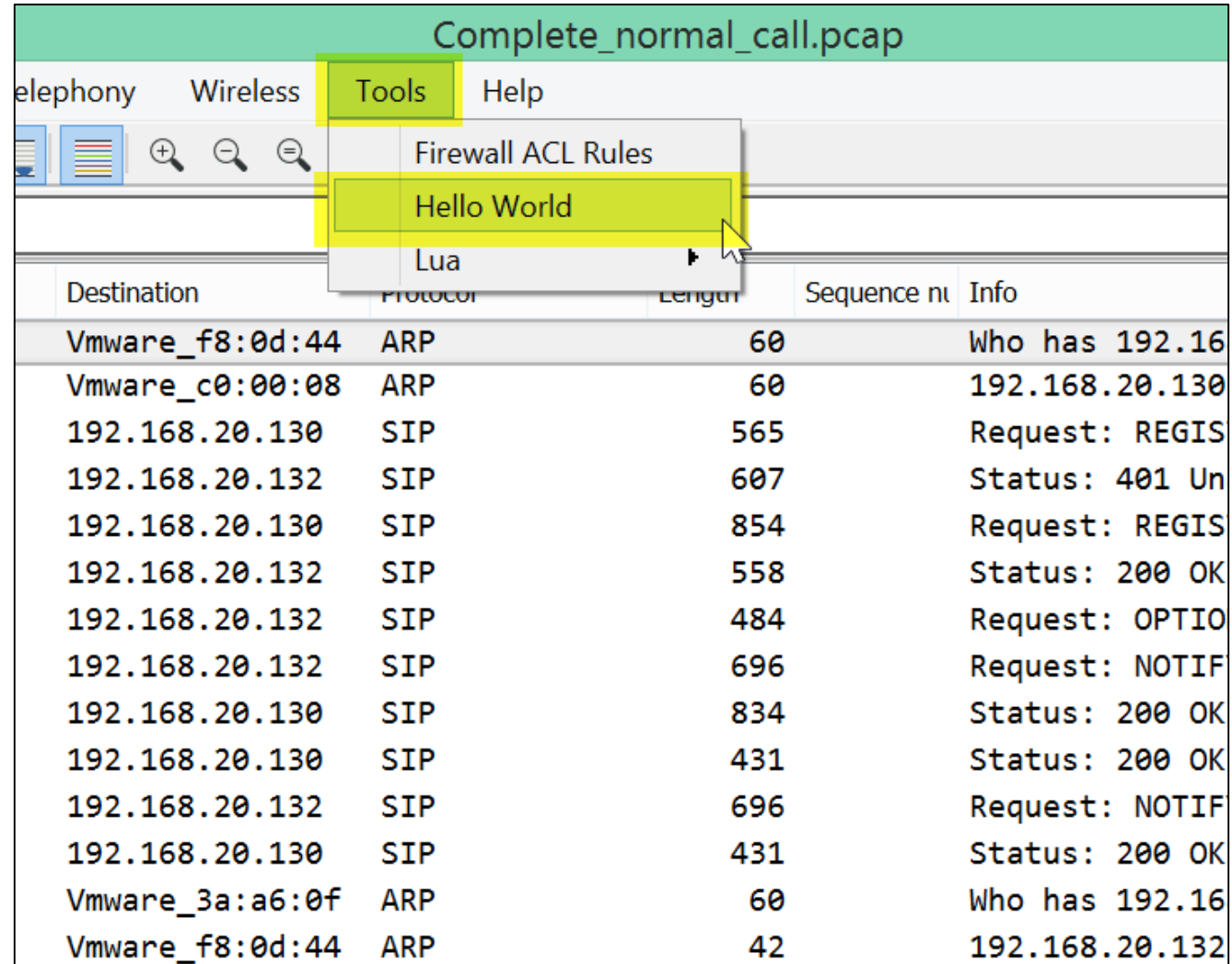
    -- Printing Hello World!!! in the Window
    win:set(" Hello World!!! ")
end
```

- Registering function with Tool menu

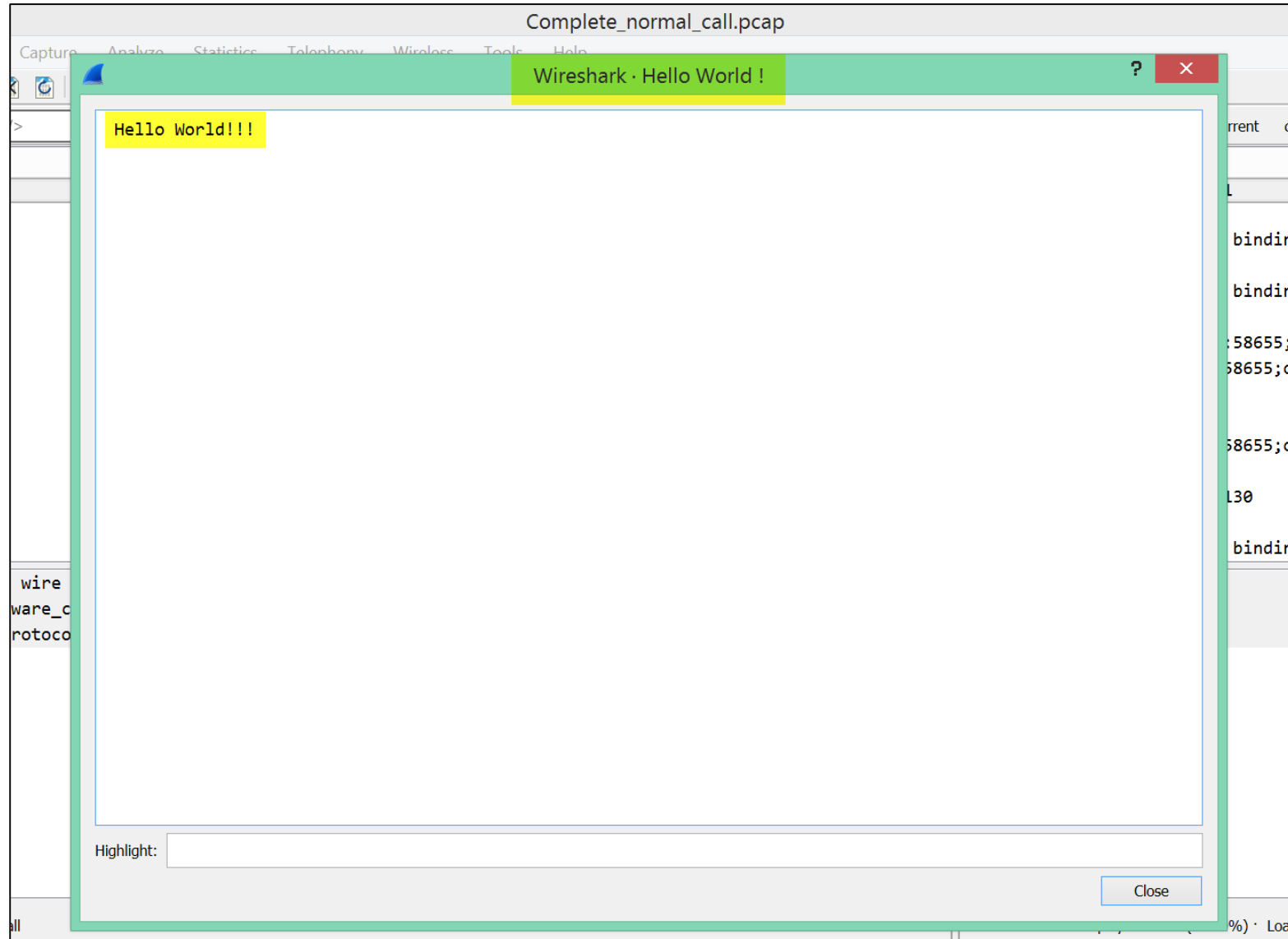
```
-- Register the function to Tools menu
register_menu("Hello World",dialog_menu, MENU_TOOLS_UNSORTED)
```



# Hello World Listener: Tools Menu Entry



# Hello World Listener: Output





# Writing Hello World Dissector

# Writing a Dissector: Proposal

- A Hello World dissector to add new protocol to protocol tree
- Dissector will add “Hello World” to all packets
- **Lua file:** hello\_world\_dissector.lua
- **Example PCAP:** Complete\_normal\_call.pcap

# Writing a Hello World Dissector

- Defining global variables and fields

```
-- Defining Proto Object
local world= Proto("hello_world","Hello World")

-- Defining fields for our protocol
local hello_world=ProtoField.string("hello_world.hello_world","Hello World")

-- Registering fields to our protocol
world.fields={
hello_world
}
```

# Writing a Hello World Dissector

- Dissector function, to add protocol tree and elements

```
function world.dissector(tvbuf,pktinfo,root)
    -- Packet length
    local pktlen=tvbuf:reported_length_remaining()

    -- Creating a tree for our protocol
    local tree=root:add(world,tvbuf:range(0,pktlen))

    -- Adding fields and corresponding value to the tree
    tree:add(hello_world,"Hello")

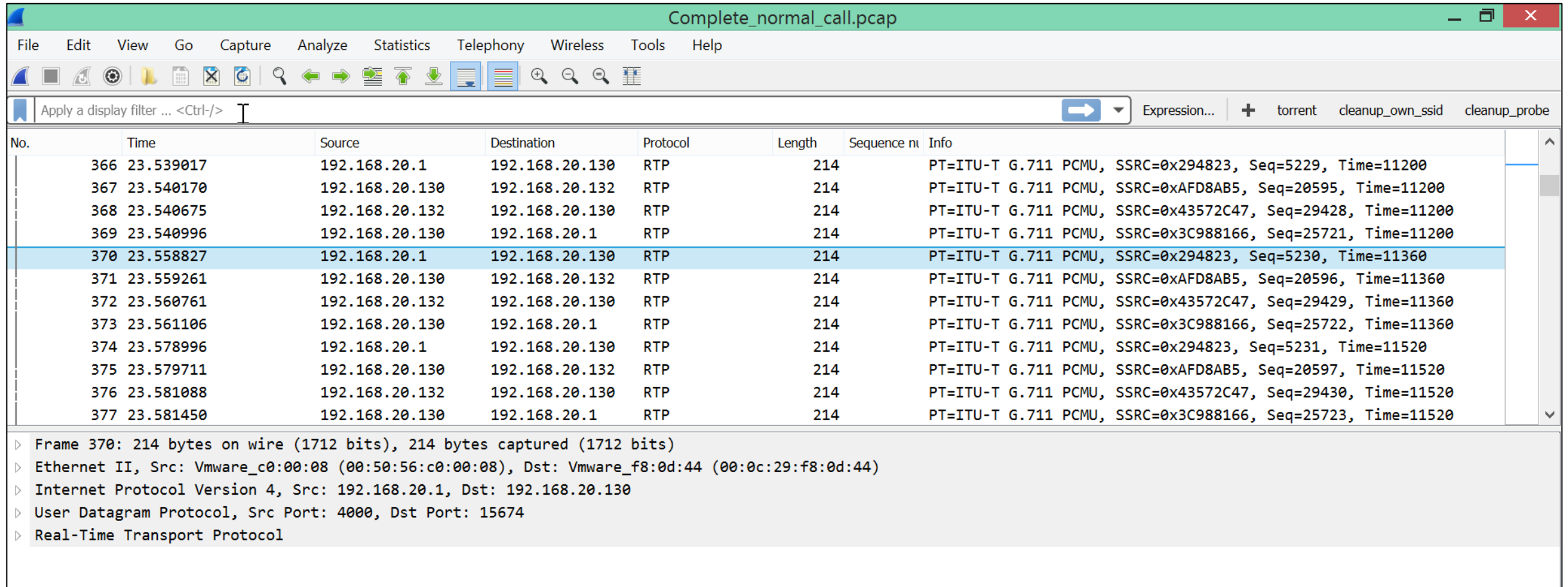
end
```

# Writing a Hello World Dissector

- Registering the dissector

```
-- Registering our protocol as post dissector  
register_postdissector(world)
```

# Hello World Dissector: Before



Complete\_normal\_call.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/> I Expression... + torrent cleanup\_own\_ssid cleanup\_probe

No.	Time	Source	Destination	Protocol	Length	Sequence no	Info
366	23.539017	192.168.20.1	192.168.20.130	RTP	214		PT=ITU-T G.711 PCMU, SSRC=0x294823, Seq=5229, Time=11200
367	23.540170	192.168.20.130	192.168.20.132	RTP	214		PT=ITU-T G.711 PCMU, SSRC=0xAFD8AB5, Seq=20595, Time=11200
368	23.540675	192.168.20.132	192.168.20.130	RTP	214		PT=ITU-T G.711 PCMU, SSRC=0x43572C47, Seq=29428, Time=11200
369	23.540996	192.168.20.130	192.168.20.1	RTP	214		PT=ITU-T G.711 PCMU, SSRC=0x3C988166, Seq=25721, Time=11200
370	23.558827	192.168.20.1	192.168.20.130	RTP	214		PT=ITU-T G.711 PCMU, SSRC=0x294823, Seq=5230, Time=11360
371	23.559261	192.168.20.130	192.168.20.132	RTP	214		PT=ITU-T G.711 PCMU, SSRC=0xAFD8AB5, Seq=20596, Time=11360
372	23.560761	192.168.20.132	192.168.20.130	RTP	214		PT=ITU-T G.711 PCMU, SSRC=0x43572C47, Seq=29429, Time=11360
373	23.561106	192.168.20.130	192.168.20.1	RTP	214		PT=ITU-T G.711 PCMU, SSRC=0x3C988166, Seq=25722, Time=11360
374	23.578996	192.168.20.1	192.168.20.130	RTP	214		PT=ITU-T G.711 PCMU, SSRC=0x294823, Seq=5231, Time=11520
375	23.579711	192.168.20.130	192.168.20.132	RTP	214		PT=ITU-T G.711 PCMU, SSRC=0xAFD8AB5, Seq=20597, Time=11520
376	23.581088	192.168.20.132	192.168.20.130	RTP	214		PT=ITU-T G.711 PCMU, SSRC=0x43572C47, Seq=29430, Time=11520
377	23.581450	192.168.20.130	192.168.20.1	RTP	214		PT=ITU-T G.711 PCMU, SSRC=0x3C988166, Seq=25723, Time=11520

Frame 370: 214 bytes on wire (1712 bits), 214 bytes captured (1712 bits)

Ethernet II, Src: Vmware\_c0:00:08 (00:50:56:c0:00:08), Dst: Vmware\_f8:0d:44 (00:0c:29:f8:0d:44)

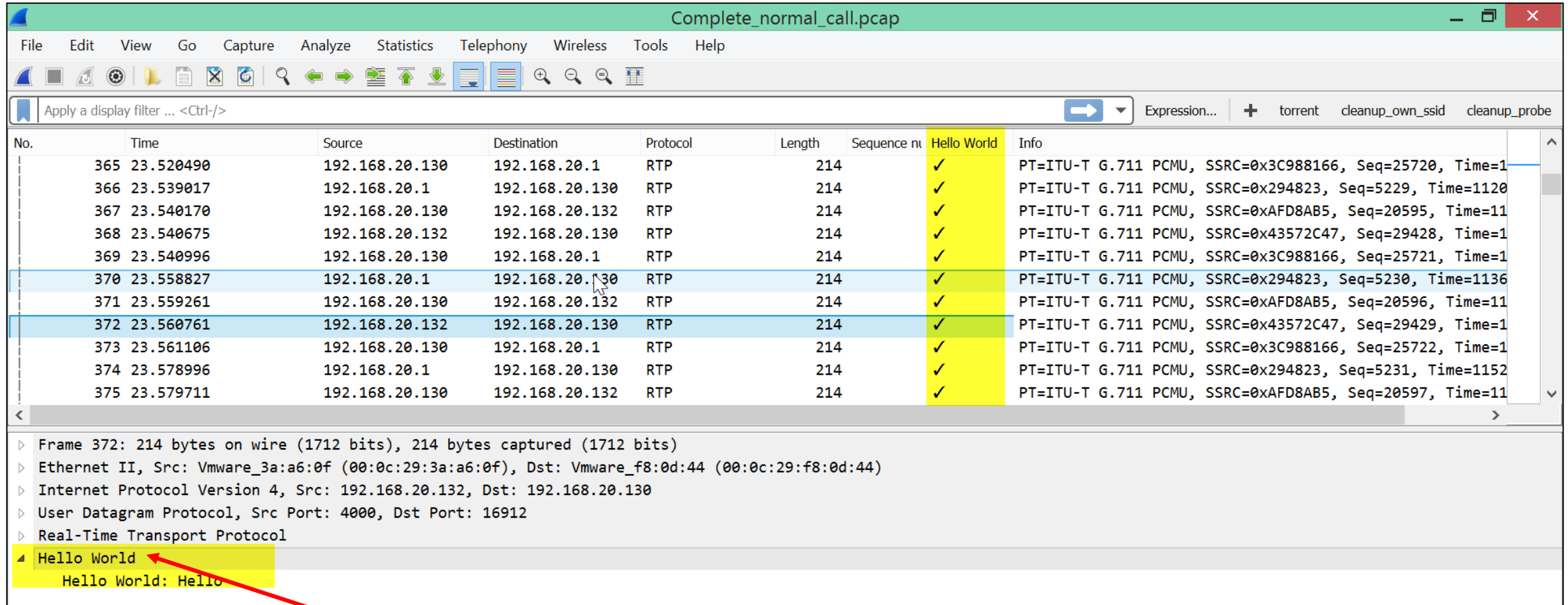
Internet Protocol Version 4, Src: 192.168.20.1, Dst: 192.168.20.130

User Datagram Protocol, Src Port: 4000, Dst Port: 15674

Real-Time Transport Protocol



# Hello World Dissector: After



The image shows a Wireshark interface with a packet capture named 'Complete\_normal\_call.pcap'. The packet list displays several RTP packets. Frame 372 is selected, and the packet details pane shows the following information:

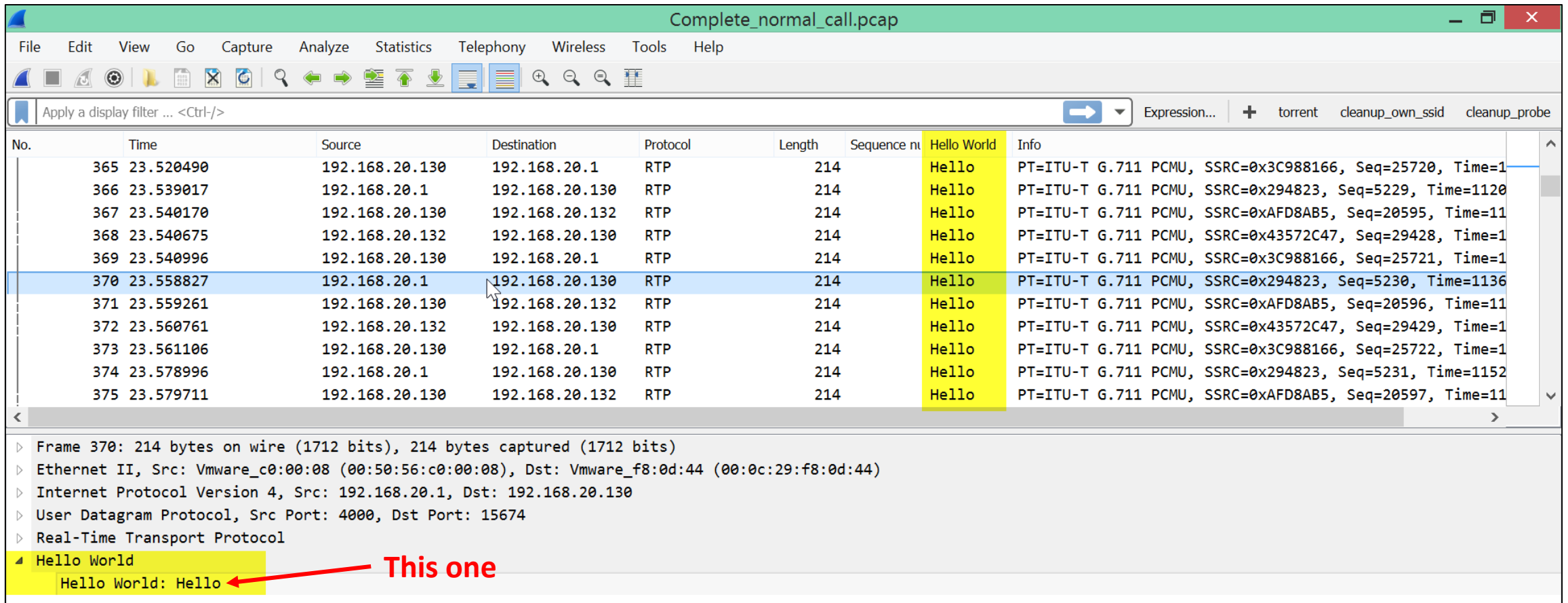
- Frame 372: 214 bytes on wire (1712 bits), 214 bytes captured (1712 bits)
- Ethernet II, Src: Vmware\_3a:a6:0f (00:0c:29:3a:a6:0f), Dst: Vmware\_f8:0d:44 (00:0c:29:f8:0d:44)
- Internet Protocol Version 4, Src: 192.168.20.132, Dst: 192.168.20.130
- User Datagram Protocol, Src Port: 4000, Dst Port: 16912
- Real-Time Transport Protocol
- Hello World
- Hello World: Hello

A red arrow points to the 'Hello World' message in the packet details pane.

This one

# Hello World Dissector: After

- Applying Hello World tree element as Column



The screenshot shows the Wireshark interface with the file 'Complete\_normal\_call.pcap' open. The packet list pane displays a series of RTP packets. The 'Hello World' column is visible, and the packet details pane shows the 'Hello World' tree element highlighted. A red arrow points to the 'Hello World: Hello' text in the packet details pane.

No.	Time	Source	Destination	Protocol	Length	Sequence no	Hello World	Info
365	23.520490	192.168.20.130	192.168.20.1	RTP	214		Hello	PT=ITU-T G.711 PCMU, SSRC=0x3C988166, Seq=25720, Time=1
366	23.539017	192.168.20.1	192.168.20.130	RTP	214		Hello	PT=ITU-T G.711 PCMU, SSRC=0x294823, Seq=5229, Time=1120
367	23.540170	192.168.20.130	192.168.20.132	RTP	214		Hello	PT=ITU-T G.711 PCMU, SSRC=0xAFD8AB5, Seq=20595, Time=11
368	23.540675	192.168.20.132	192.168.20.130	RTP	214		Hello	PT=ITU-T G.711 PCMU, SSRC=0x43572C47, Seq=29428, Time=1
369	23.540996	192.168.20.130	192.168.20.1	RTP	214		Hello	PT=ITU-T G.711 PCMU, SSRC=0x3C988166, Seq=25721, Time=1
370	23.558827	192.168.20.1	192.168.20.130	RTP	214		Hello	PT=ITU-T G.711 PCMU, SSRC=0x294823, Seq=5230, Time=1136
371	23.559261	192.168.20.130	192.168.20.132	RTP	214		Hello	PT=ITU-T G.711 PCMU, SSRC=0xAFD8AB5, Seq=20596, Time=11
372	23.560761	192.168.20.132	192.168.20.130	RTP	214		Hello	PT=ITU-T G.711 PCMU, SSRC=0x43572C47, Seq=29429, Time=1
373	23.561106	192.168.20.130	192.168.20.1	RTP	214		Hello	PT=ITU-T G.711 PCMU, SSRC=0x3C988166, Seq=25722, Time=1
374	23.578996	192.168.20.1	192.168.20.130	RTP	214		Hello	PT=ITU-T G.711 PCMU, SSRC=0x294823, Seq=5231, Time=1152
375	23.579711	192.168.20.130	192.168.20.132	RTP	214		Hello	PT=ITU-T G.711 PCMU, SSRC=0xAFD8AB5, Seq=20597, Time=11

Frame 370: 214 bytes on wire (1712 bits), 214 bytes captured (1712 bits)

- Ethernet II, Src: Vmware\_c0:00:08 (00:50:56:c0:00:08), Dst: Vmware\_f8:0d:44 (00:0c:29:f8:0d:44)
- Internet Protocol Version 4, Src: 192.168.20.1, Dst: 192.168.20.130
- User Datagram Protocol, Src Port: 4000, Dst Port: 15674
- Real-Time Transport Protocol
  - Hello World
    - Hello World: Hello



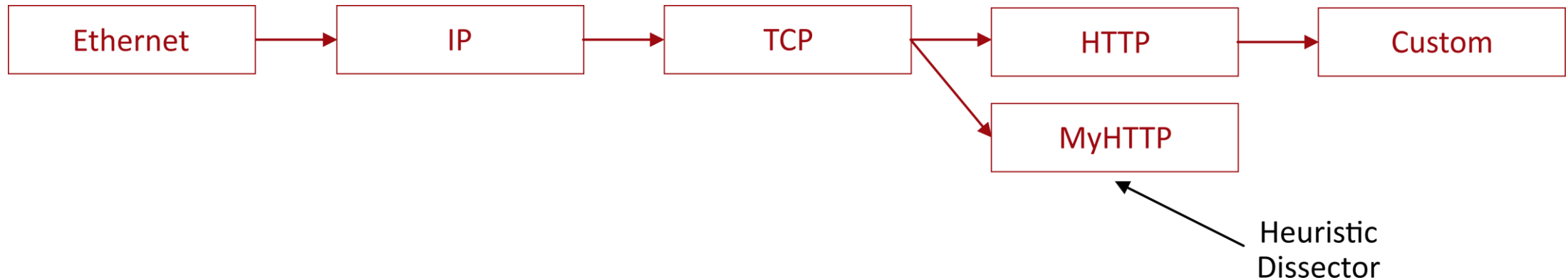
# Heuristic Dissector

# Heuristic Dissector

- Identifies the protocol on the basis of heuristics
- Heuristics can be
  - Average size or size range of the packets
  - Specific codes or strings in the header or the payload
- Useful when port based detection fails i.e. protocols operating on non standard ports (e.g. DNS server running on port 8089)

# Heuristic Dissector

## Example Dissection Flow



**Example:** DNS heuristic dissector

**Lua File:** dns\_dissector.lua

**File Source:**

<https://wiki.wireshark.org/Lua/Examples?action=AttachFile&do=get&target=dissector.lua>

**Note:** The heuristic dissector will only give result if no existing dissector is able to identify the packet

# Heuristic Dissector: DNS Server on Port 8089

DNS\_traffic\_server\_port\_8089.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

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

No.	Time	Source	Protocol	Destination	Length	Info
11	4.555392055	192.168.1.16	SSDP	239.255.255.250	216	M-SEARCH * HTTP/1.1
12	5.114518734	192.168.1.115	DB-LS...	192.168.1.255	188	Dropbox LAN sync Discovery Protocol
13	5.462842742	192.168.1.33	UDP	192.168.1.104	74	46191 → 8089 Len=32
14	5.463360963	192.168.1.104	UDP	192.168.1.33	90	8089 → 46191 Len=48

▶ Frame 13: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0

▶ Ethernet II, Src: HewlettP\_4b:06:c9 (48:0f:cf:4b:06:c9), Dst: CompalIn\_4b:c4:4d (f8:a9:63:4b:c4:4d)

▶ Internet Protocol Version 4, Src: 192.168.1.33, Dst: 192.168.1.104

▲ User Datagram Protocol, Src Port: 46191, Dst Port: 8089

- Source Port: 46191
- Destination Port: 8089
- Length: 40
- Checksum: 0xf239 [unverified]
- [Checksum Status: Unverified]
- [Stream index: 7]

▲ Data (32 bytes)

Data: 26da01000001000000000000377777706676f6f676c6503...

[Length: 32]

```
0000 f8 a9 63 4b c4 4d 48 0f cf 4b 06 c9 08 00 45 00 ..cK.MH. .K....E.
0010 00 3c cb 4f 00 00 40 11 2b 88 c0 a8 01 21 c0 a8 .<.O..@. +....!..
0020 01 68 b4 6f 1f 99 00 28 f2 39 26 da 01 00 00 01 .h.o...( .9&....
0030 00 00 00 00 00 00 03 77 77 77 06 67 6f 6f 67 6c .....w ww.googl
0040 65 03 63 6f 6d 00 00 01 00 01 e.com... ..
```

# Heuristic Dissector: Identifying DNS Traffic

DNS\_traffic\_server port\_8089.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

mydns

No.	Time	Source	Protocol	Destination	Length	Info
13	5.462842742	192.168.1.33	MYDNS	192.168.1.104	74	Query (9946) www.google.com
14	5.463360963	192.168.1.104	MYDNS	192.168.1.33	90	Response (9946) www.google.com
15	5.463688094	192.168.1.33	MYDNS	192.168.1.104	74	Query (40285) www.google.com
16	5.565232771	192.168.1.104	MYDNS	192.168.1.33	102	Response (40285) www.google.com

Frame 13: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0

Ethernet II, Src: HewlettP\_4b:06:c9 (48:0f:cf:4b:06:c9), Dst: CompalIn\_4b:c4:4d (f8:a9:63:4b:c4:4d)

Internet Protocol Version 4, Src: 192.168.1.33, Dst: 192.168.1.104

User Datagram Protocol, Src Port: 46191, Dst Port: 8089

Source Port: 46191

Destination Port: 8089

Length: 40

Checksum: 0xf239 [unverified]

[Checksum Status: Unverified]

[Stream index: 7]

[Heuristic dissector used]

MyDNS Protocol

Transaction ID: 9946

Flags: 0x0100

Number of Questions: 1

Number of Answer RRs: 0

Number of Authority RRs: 0

Number of Additional RRs: 0

Queries

# Heuristic Dissector for DNS

- Defining heuristic function

```
-- Note: this heuristic stuff is new in 1.11.3
local function heur_dissect_dns(tvbuf,pktinfo,root)
    dprint2("heur_dissect_dns called")
```

- Factor 1: Length should be greater than DNS header length

```
-- the length should be greater than DNS header length
if tvbuf:len() < DNS_HDR_LEN then
    dprint("heur_dissect_dns: tvb shorter than DNS_HDR_LEN of:",DNS_HDR_LEN)
    return false
end
```



# Heuristic Dissector for DNS

- Factor 2: Opcode has to be one of 0, 1, 2, 4, 5

```
-- the opcode has to be 0, 1, 2, 4 or 5
-- the opcode field starts at bit offset 17 (in C-indexing), for 4 bits in length
local check = tvbr:bitfield(17,4)
if check == 3 or check > 5 then
    dprint("heur_dissect_dns: invalid opcode:",check)
    return false
end
```

- Factor 3: Rcode has to be 0-10, 16-22

```
-- the rcode has to be 0-10, 16-22 (we're ignoring private use rcodes here)
-- the rcode field starts at bit offset 28 (in C-indexing), for 4 bits in length
check = tvbr:bitfield(28,4)
if check > 22 or (check > 10 and check < 16) then
    dprint("heur_dissect_dns: invalid rcode:",check)
    return false
end
```

# Heuristic Dissector for DNS

- Factor 4: Number of queries and answers should be reasonable

```
-- now let's verify the number of questions/answers are reasonable
check = tvbr:range(4,2):uint() -- num questions
if check > 100 then return false end
check = tvbr:range(6,2):uint() -- num answers
if check > 100 then return false end
check = tvbr:range(8,2):uint() -- num authority
if check > 100 then return false end
check = tvbr:range(10,2):uint() -- num additional
if check > 100 then return false end
```

- Register the function

```
-- now register that heuristic dissector into the udp heuristic list
dns:register_heuristic("udp",heur_dissect_dns)
```

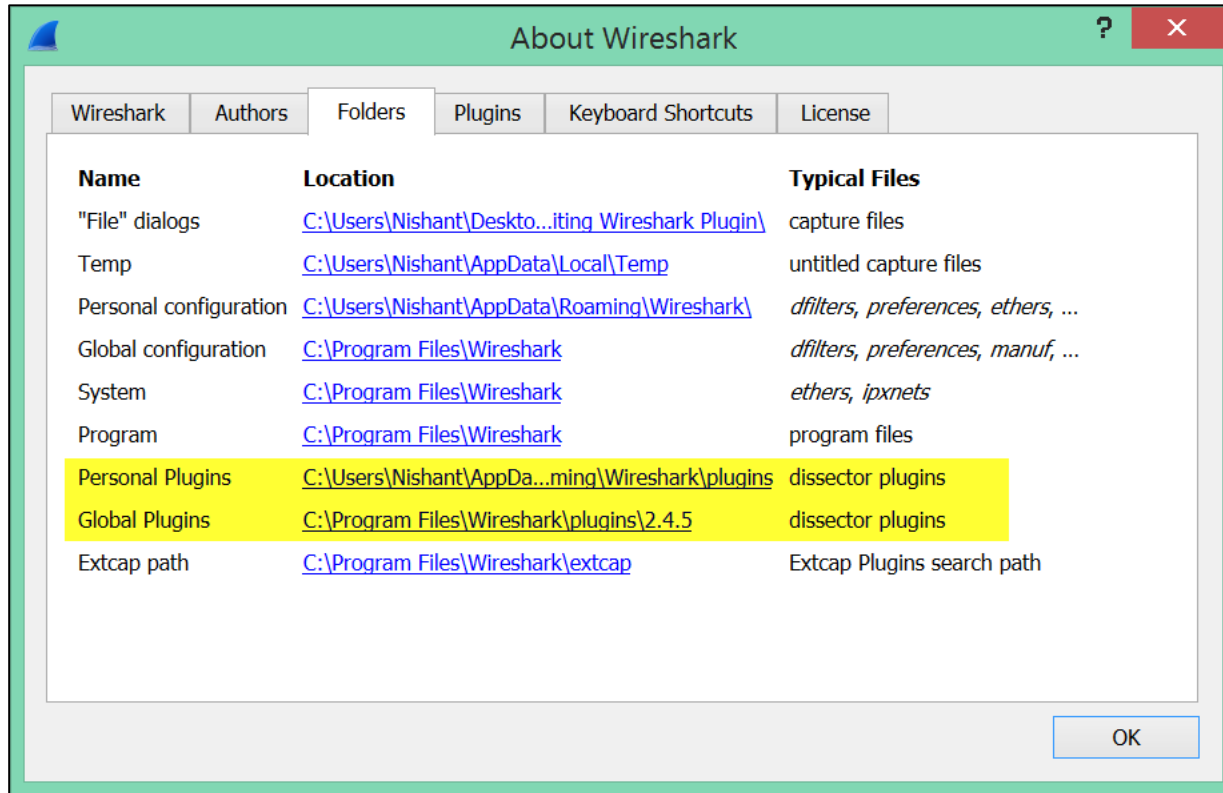


# PA Toolkit: WiFi

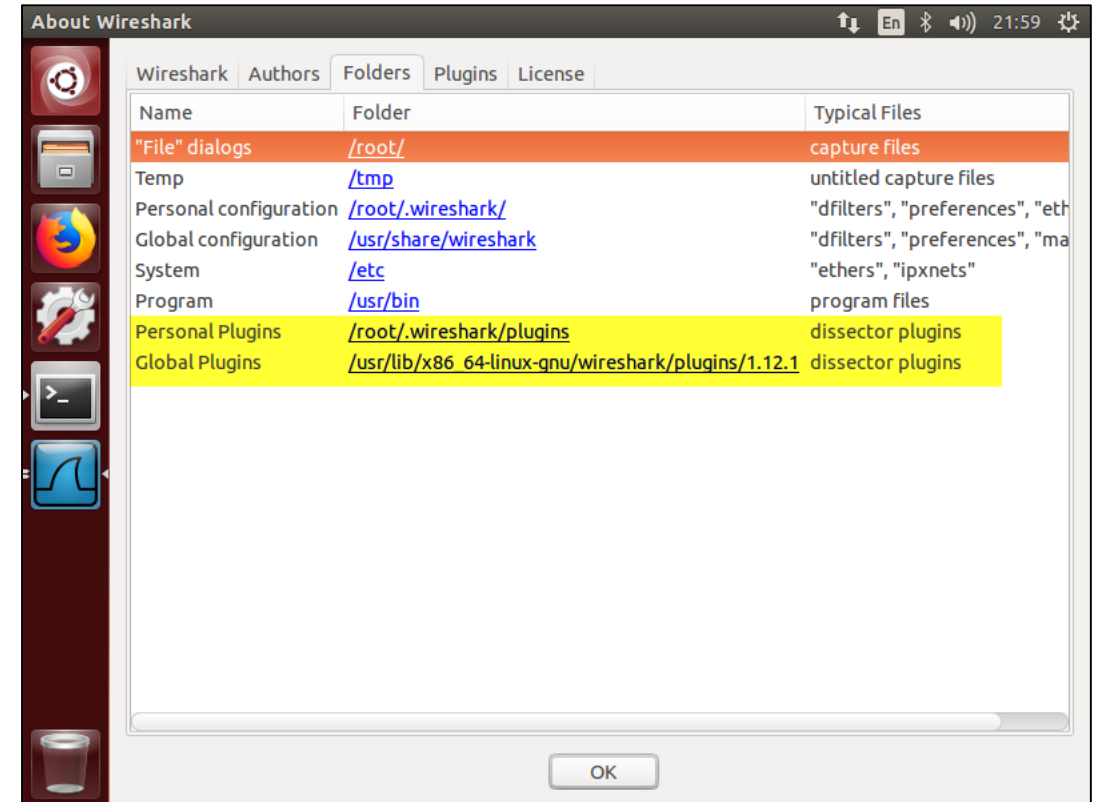
# Where to place plugins?

- Check Help > About Wireshark > Folders

## Windows



## Ubuntu

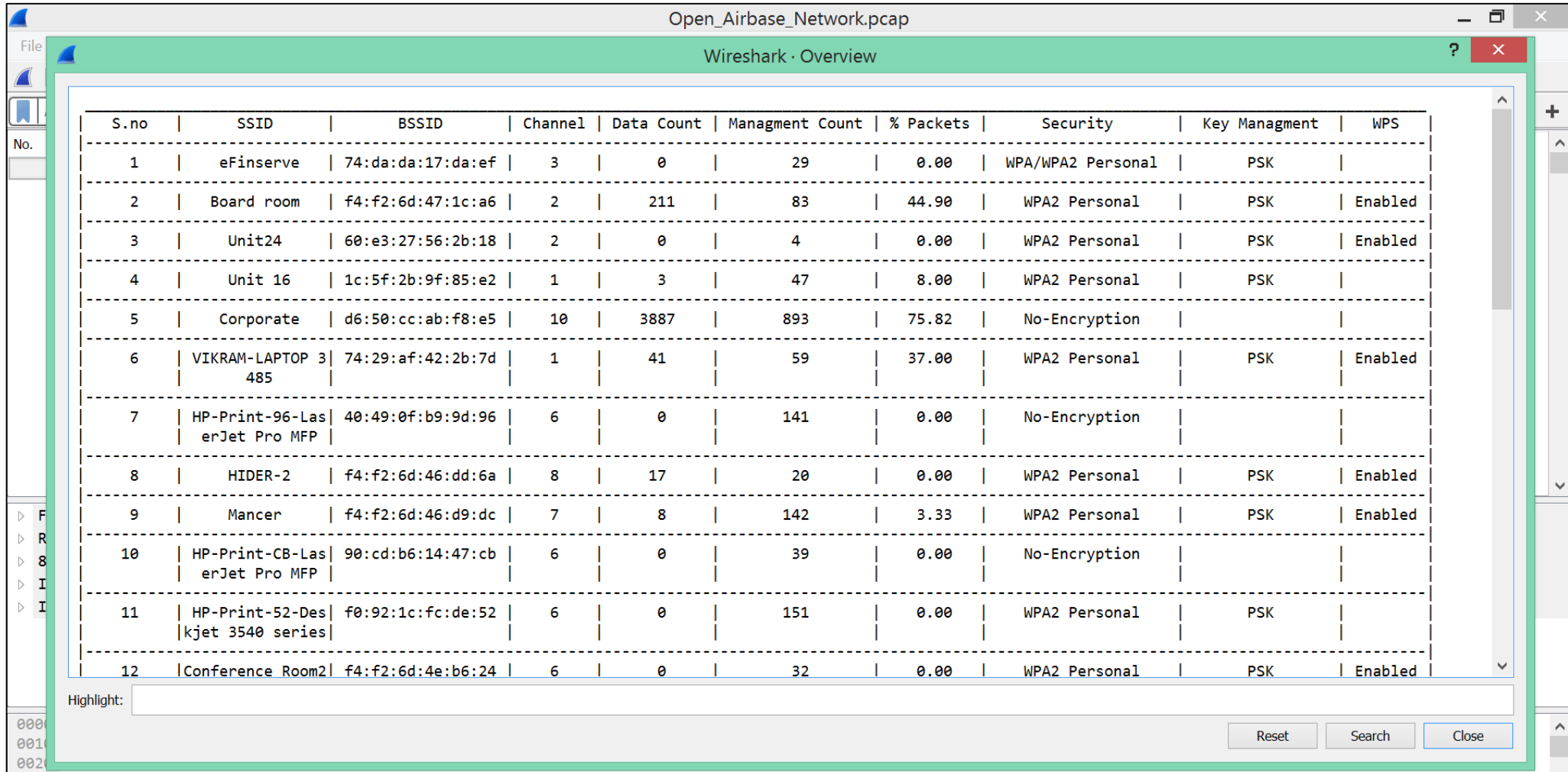


# WiFi Plugins

- Overview (Shows all present SSIDs with their details)
- Beacon Flood Detection (Show possible beacon flood attempts)
- Deauth Disassoc Detection (Shows stats related to deauth and disassoc packet)
- WPA Cracking Possibility (Shows AP-client pairs for which WPA handshake packets are available)

# Overview

- PCAP File: Airbase\_detection/Open\_Airbase\_Network.pcap

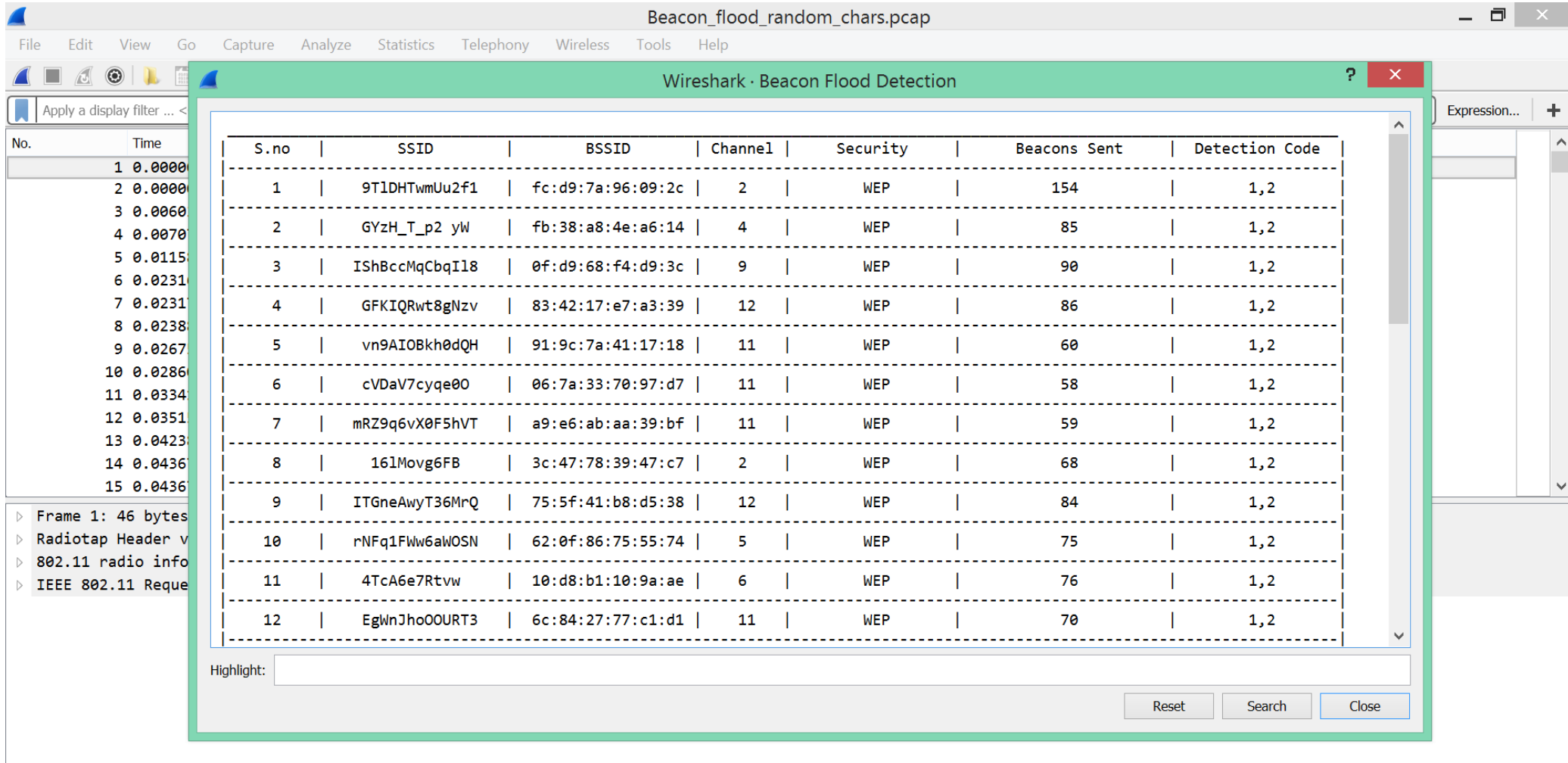


The image shows a screenshot of the Wireshark 'Overview' window. The window title is 'Open\_Airbase\_Network.pcap'. The main content is a table with 10 columns: S.no, SSID, BSSID, Channel, Data Count, Managment Count, % Packets, Security, Key Managment, and WPS. The table lists 12 detected networks. The interface includes a sidebar on the left with icons for File, No., and a list of network interfaces. At the bottom, there is a 'Highlight:' field and buttons for 'Reset', 'Search', and 'Close'.

S.no	SSID	BSSID	Channel	Data Count	Managment Count	% Packets	Security	Key Managment	WPS
1	eFinserve	74:da:da:17:da:ef	3	0	29	0.00	WPA/WPA2 Personal	PSK	
2	Board room	f4:f2:6d:47:1c:a6	2	211	83	44.90	WPA2 Personal	PSK	Enabled
3	Unit24	60:e3:27:56:2b:18	2	0	4	0.00	WPA2 Personal	PSK	Enabled
4	Unit 16	1c:5f:2b:9f:85:e2	1	3	47	8.00	WPA2 Personal	PSK	
5	Corporate	d6:50:cc:ab:f8:e5	10	3887	893	75.82	No-Encryption		
6	VIKRAM-LAPTOP 3485	74:29:af:42:2b:7d	1	41	59	37.00	WPA2 Personal	PSK	Enabled
7	HP-Print-96-LaserJet Pro MFP	40:49:0f:b9:9d:96	6	0	141	0.00	No-Encryption		
8	HIDER-2	f4:f2:6d:46:dd:6a	8	17	20	0.00	WPA2 Personal	PSK	Enabled
9	Mancer	f4:f2:6d:46:d9:dc	7	8	142	3.33	WPA2 Personal	PSK	Enabled
10	HP-Print-CB-LaserJet Pro MFP	90:cd:b6:14:47:cb	6	0	39	0.00	No-Encryption		
11	HP-Print-52-Deskjet 3540 series	f0:92:1c:fc:de:52	6	0	151	0.00	WPA2 Personal	PSK	
12	Conference Room2	f4:f2:6d:4e:b6:24	6	0	32	0.00	WPA2 Personal	PSK	Enabled

# Beacon Flood Detection

- **PCAP File:** Beacon\_flood\_random.pcap

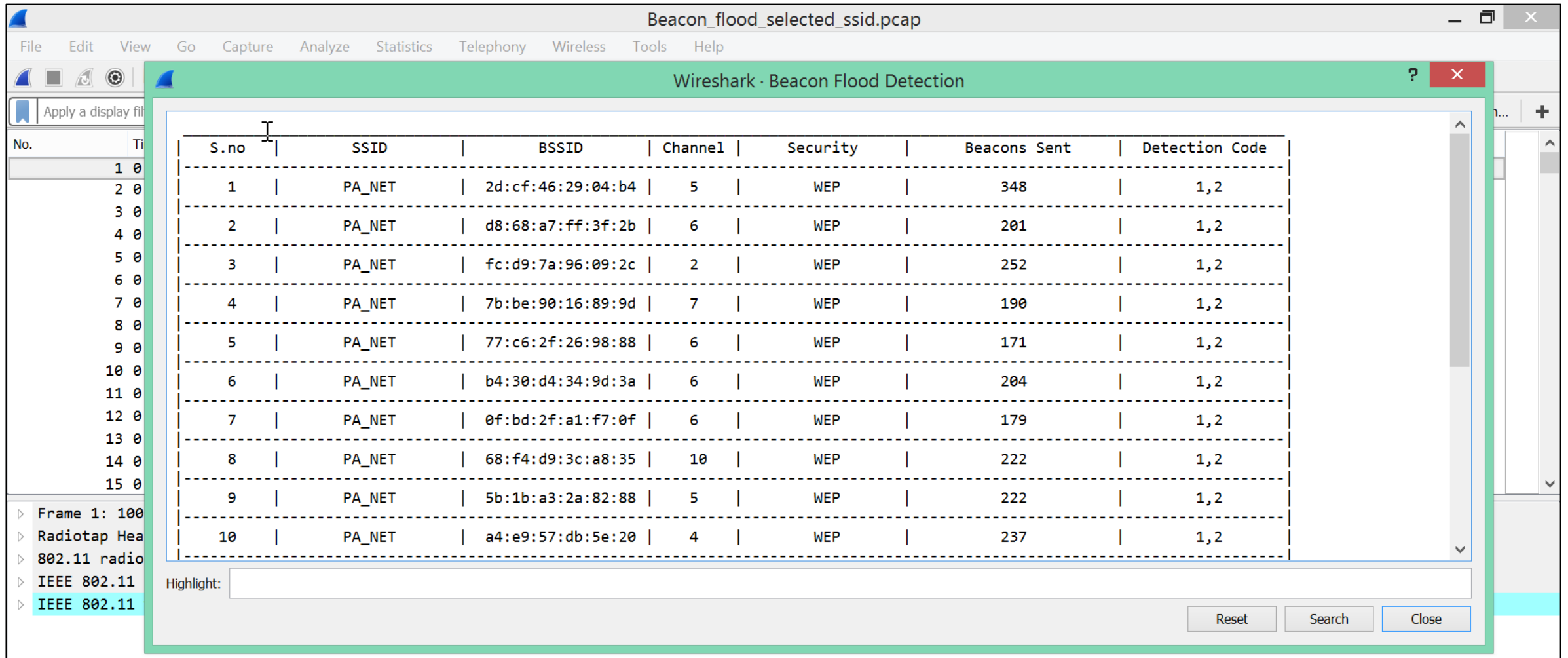


The image shows a Wireshark window titled "Beacon\_flood\_random\_chars.pcap" with a menu bar (File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, Help) and a toolbar. A "Wireshark · Beacon Flood Detection" dialog box is open, displaying a table of detected beacon floods. The table has columns: S.no, SSID, BSSID, Channel, Security, Beacons Sent, and Detection Code. The table contains 12 rows of data. Below the table is a "Highlight:" text field and buttons for "Reset", "Search", and "Close".

S.no	SSID	BSSID	Channel	Security	Beacons Sent	Detection Code
1	9T1DHTwmUu2f1	fc:d9:7a:96:09:2c	2	WEP	154	1,2
2	GYzH_T_p2 yW	fb:38:a8:4e:a6:14	4	WEP	85	1,2
3	IShBccMqCbqIl8	0f:d9:68:f4:d9:3c	9	WEP	90	1,2
4	GFKIQRwt8gNzv	83:42:17:e7:a3:39	12	WEP	86	1,2
5	vn9AIOBkh0dQH	91:9c:7a:41:17:18	11	WEP	60	1,2
6	cVDAv7cyqe00	06:7a:33:70:97:d7	11	WEP	58	1,2
7	mRZ9q6vX0F5hVT	a9:e6:ab:aa:39:bf	11	WEP	59	1,2
8	16IMovg6FB	3c:47:78:39:47:c7	2	WEP	68	1,2
9	ITGneAwyt36MrQ	75:5f:41:b8:d5:38	12	WEP	84	1,2
10	rNFq1FWw6aWOSN	62:0f:86:75:55:74	5	WEP	75	1,2
11	4TcA6e7Rtvw	10:d8:b1:10:9a:ae	6	WEP	76	1,2
12	EgWnJho0OURT3	6c:84:27:77:c1:d1	11	WEP	70	1,2

# Beacon Flood Detection

- **PCAP File:** Beacon\_flood\_selected\_ssid.pcap



The image shows a Wireshark window titled "Beacon\_flood\_selected\_ssid.pcap" with a menu bar (File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, Help) and a toolbar. A "Wireshark · Beacon Flood Detection" dialog box is open, displaying a table of detected beacon floods. The table has columns for S.no, SSID, BSSID, Channel, Security, Beacons Sent, and Detection Code. The data is as follows:

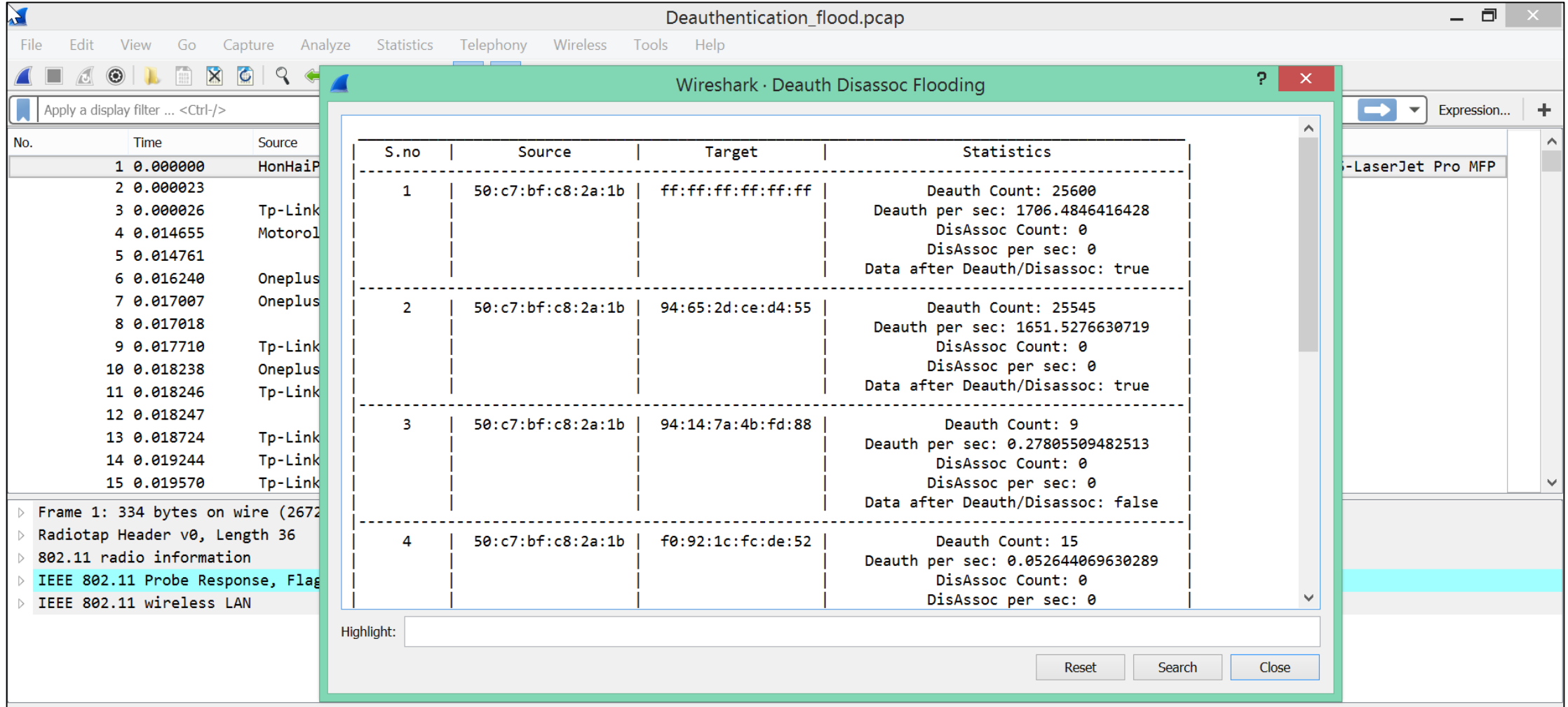
S.no	SSID	BSSID	Channel	Security	Beacons Sent	Detection Code
1	PA_NET	2d:cf:46:29:04:b4	5	WEP	348	1,2
2	PA_NET	d8:68:a7:ff:3f:2b	6	WEP	201	1,2
3	PA_NET	fc:d9:7a:96:09:2c	2	WEP	252	1,2
4	PA_NET	7b:be:90:16:89:9d	7	WEP	190	1,2
5	PA_NET	77:c6:2f:26:98:88	6	WEP	171	1,2
6	PA_NET	b4:30:d4:34:9d:3a	6	WEP	204	1,2
7	PA_NET	0f:bd:2f:a1:f7:0f	6	WEP	179	1,2
8	PA_NET	68:f4:d9:3c:a8:35	10	WEP	222	1,2
9	PA_NET	5b:1b:a3:2a:82:88	5	WEP	222	1,2
10	PA_NET	a4:e9:57:db:5e:20	4	WEP	237	1,2

Below the table is a "Highlight:" text field. At the bottom right of the dialog are "Reset", "Search", and "Close" buttons. The background shows the Wireshark interface with a packet list on the left and a packet details pane on the right.



# Deauth Disassoc Flooding

- **PCAP File:** Deauthentication\_flood.pcap



The screenshot shows the Wireshark interface with a packet capture of a Deauthentication Disassoc Flooding attack. A summary window titled "Wireshark · Deauth Disassoc Flooding" is open, displaying a table of attack statistics.

S.no	Source	Target	Statistics
1	50:c7:bf:c8:2a:1b	ff:ff:ff:ff:ff:ff	Deauth Count: 25600 Deauth per sec: 1706.4846416428 DisAssoc Count: 0 DisAssoc per sec: 0 Data after Deauth/Disassoc: true
2	50:c7:bf:c8:2a:1b	94:65:2d:ce:d4:55	Deauth Count: 25545 Deauth per sec: 1651.5276630719 DisAssoc Count: 0 DisAssoc per sec: 0 Data after Deauth/Disassoc: true
3	50:c7:bf:c8:2a:1b	94:14:7a:4b:fd:88	Deauth Count: 9 Deauth per sec: 0.27805509482513 DisAssoc Count: 0 DisAssoc per sec: 0 Data after Deauth/Disassoc: false
4	50:c7:bf:c8:2a:1b	f0:92:1c:fc:de:52	Deauth Count: 15 Deauth per sec: 0.052644069630289 DisAssoc Count: 0 DisAssoc per sec: 0

The background shows the main Wireshark interface with a packet list on the left and a packet details pane on the right. The packet list shows a capture of 15 frames, with the first frame selected. The packet details pane shows the structure of the selected frame, including the IEEE 802.11 wireless LAN header and the IEEE 802.11 Probe Response frame.

# WPA Cracking Possibility

- **PCAP File:** Bruteforcing\_WPA2\_PSK.pcap

The image shows the Wireshark network protocol analyzer interface. The main window displays a PCAP file named 'Bruteforcing\_WPA2\_PSK.pcap'. The packet list on the left shows 15 packets, with the first packet selected. A dialog box titled 'Wireshark · Possible Handshake Cracking' is overlaid on the main window. This dialog box contains a table with the following data:

S.no	Access Point	Station	Available Handshakes For Breaking
1	30:b5:c2:11:de:2a	e8:de:27:16:70:c9	1,2,3,4
2	30:b5:c2:11:de:2a	5c:51:88:31:a0:3b	1,2,3,4

Below the table, there is a 'Highlight:' text field and three buttons: 'Reset', 'Search', and 'Close'.



# Live Demo: PA Toolkit



Q & A



Thanks!