# Wireshark

If you need to do a quick tcpdump like capture from the command line in Windows, don't forget pktmon.

# Capture Filters

## Download and install

Silently install wireshark and npcap

```
# wireshark silent installer will not install npacp - tested
cd $env:TEMP
Invoke-WebRequest -URI https://1.na.dl.wireshark.org/win64/Wireshark-latest-x64.exe -Out
Wireshark-latest-x64.exe
Start-Process Wireshark-latest-x64.exe -Wait -ArgumentList
@("/D","/S","/desktopicon=no","/quicklaunchicon=no", "/EXTRACOMPONENTS=sshdump,udpdump")

get-process | Sort-Object -Property ProcessName | Where-Object {$_.ProcessName -Like
'Wireshark*'}



# npcap download and install
# only npcap oem supports silent installation
cd $env:TEMP
Invoke-WebRequest -URI https://npcap.com/dist/npcap-1.79.exe -Out npcap-1.79.exe
Start-Process npcap-1.79.exe -Wait -ArgumentList @("/force","/admin_only=yes")

get-process | Sort-Object -Property ProcessName | Where-Object {$_.ProcessName -Like 'npcap*'}
```

## MAC address OUI

Source

```
# haven't figured this capture filter out yet... display filter is easy...
```

# bootp and dhcp

Source

```
port 67 or port 68
```

# Name resolution protocols

## DNS

Cisco Discovery Protocol

```
udp port 53
```

## mDNS

multicast DNS

```
udp port 5353
```

## LLMNR

Link-local multicast name resolution

```
udp port 5355
```

## All together now

```
udp port 53 or udp port 5353 or udp port 5355
```

# Network discovery protocols

An easy way to view discovery protocol traffic from a laptop is by using Wireshark and the capture filters below for CDP, LLDP and MNDP. Use the appropriate capture filter for the type of device

you're trying to gather information about, or use all three of them in the same capture filter.

## CDP

Cisco Discovery Protocol

```
ether host 01:00:0c:cc:cc:cc and ether[16:4] = 0x0300000C and ether[20:2] == 0x2000
```

## LLDP

Link Layer Discovery Protocol

```
ether proto 0x88cc
```

## MNDP

Mikrotik Discovery Protocol

```
udp dst port 5678 and udp src port 5678
```

## CDP/LLDP/MNDP

All three of the above capture filters in one:

```
(ether host 01:00:0c:cc:cc:cc and ether[16:4] = 0x0300000C and ether[20:2] == 0x2000) or
(ether proto 0x88cc) or (udp dst port 5678 and udp src port 5678)
```

# Capturing on an interval in Linux

The command below will capture all traffic to/from 8.8.8.8. A new capture file will be created every 600 seconds (10 minutes).

```
dumpcap -b duration:600 -f "host 8.8.8.8" -w capture-google
```

# Mikrotik Packet Capture Streaming

To accept only TZSP traffic, Capture Filter like this can be used:

```
udp port 37008
```

Note that TZSP can be sent on any UDP port you set it to, so adjust the above capture as needed.

# Using tshark

## Interface List

This is typically needed when running tshark on Windows.

```
tshark -D
thsark -i <interface_id>
```

## Capture Filter

```
# capture only udp dns packets
tshark -f "udp port 53"
```

## Saving Packets

```
# save packets (doesn't display packets)
tsharp -f "udp port 37008" -w captured.pcap

# save and display packets
tsharp -f "udp port 37008" -w captured.pcap -P

# save and display packets with LOTS of detail
tsharp -f "udp port 37008" -w captured.pcap -P -O dns -V
```

## Automatic stop

Options are duration:[seconds], filesize:[KB], and files:[n].

```
tshark -a duration:60
tshark -a filesize:1000
```

## Ring Buffer Capture

```
tshark -b duration:3600 -b filesize:1000 -b files:24 -w ring_buffer.pcap
tshark -b duration:86400 -b filesize:1000 -b files:30 -w ring_buffer.pcap
```

## Practical examples

```
# TZSP stream capture on specific interface
tshark -f "udp port 37008" -i 5

# TZSP stream capture on alternate udp port, uses decode as feature
tshark -f "udp port 37091" -d udp.port==37091,tzsp
```

### DNS examples

```
# DNS queries
tshark -n -T fields -e ip.src -e ip.dst -e dns.qry.name -e dns.resp.name -f 'udp port 53'

# DNS query contains specific string
tshark -n -T fields -e dns.qry.name -f 'src port 53' -Y 'dns.qry.name contains "foo"'

# detailed DNS queries and responses
sudo tshark -nn -T fields -e frame.time -e ip.src -e ip.dst -e dns.count.queries -e
dns.count.answers -e dns.qry.name -e dns.qry.type -e dns.resp.name -e dns.resp.type -e
dns.resp.ttl -Y 'dns.flags.rcode==0 && dns.flags.response==1'
```

# Display Filters

### IP Address in Subnet

```
ip.address == 10.248.1.0/24
```

### Radius AVP User-Name contains string

The username could also be the hostname of the device authenticating.

```
radius.User_Name ~ "NIT"
```

# Wireless SSID via Radius AVP Called-Station-Id

```
radius.Called_Station_Id ~ "SSID"
```

-end

---

Revision #8
Created 2018-08-06 03:49:54 UTC by bluecrow76
Updated 2025-12-18 18:46:11 UTC by bluecrow76