Reti Informatiche

WireShark

www.wireshark.org

What is WireShark

Wireshark is a network packet analyzer. A network packet analyzer will try to capture network packets and tries to display that packet data as detailed as possible.

You could think of a network packet analyzer as a measuring device used to examine what's going on inside a network cable, just like a voltmeter is used by an electrician to examine what's going on inside an electric cable (but at a higher level, of course).

In the past, such tools were either very expensive, proprietary, or both. However, with the advent of Wireshark, all that has changed.

People use Wireshark for

- network administrators use it to troubleshoot
- network problems
- network security engineers use it to examine security problems
- developers use it to debug protocol implementations
- people use it to learn network protocol internals

Features

- Capture live packet data from a network interface.
- Display packets with very detailed protocol information.
- Open and Save packet data captured.
- Import and Export packet data from and to a lot of other capture programs.
- Filter packets on many criteria.
- Search for packets on many criteria.
- Colorize packet display based on filters.
- Create various statistics.

What WireShark is not

- Wireshark isn't an intrusion detection system.
- Wireshark will not manipulate things on the network.

A Brief History

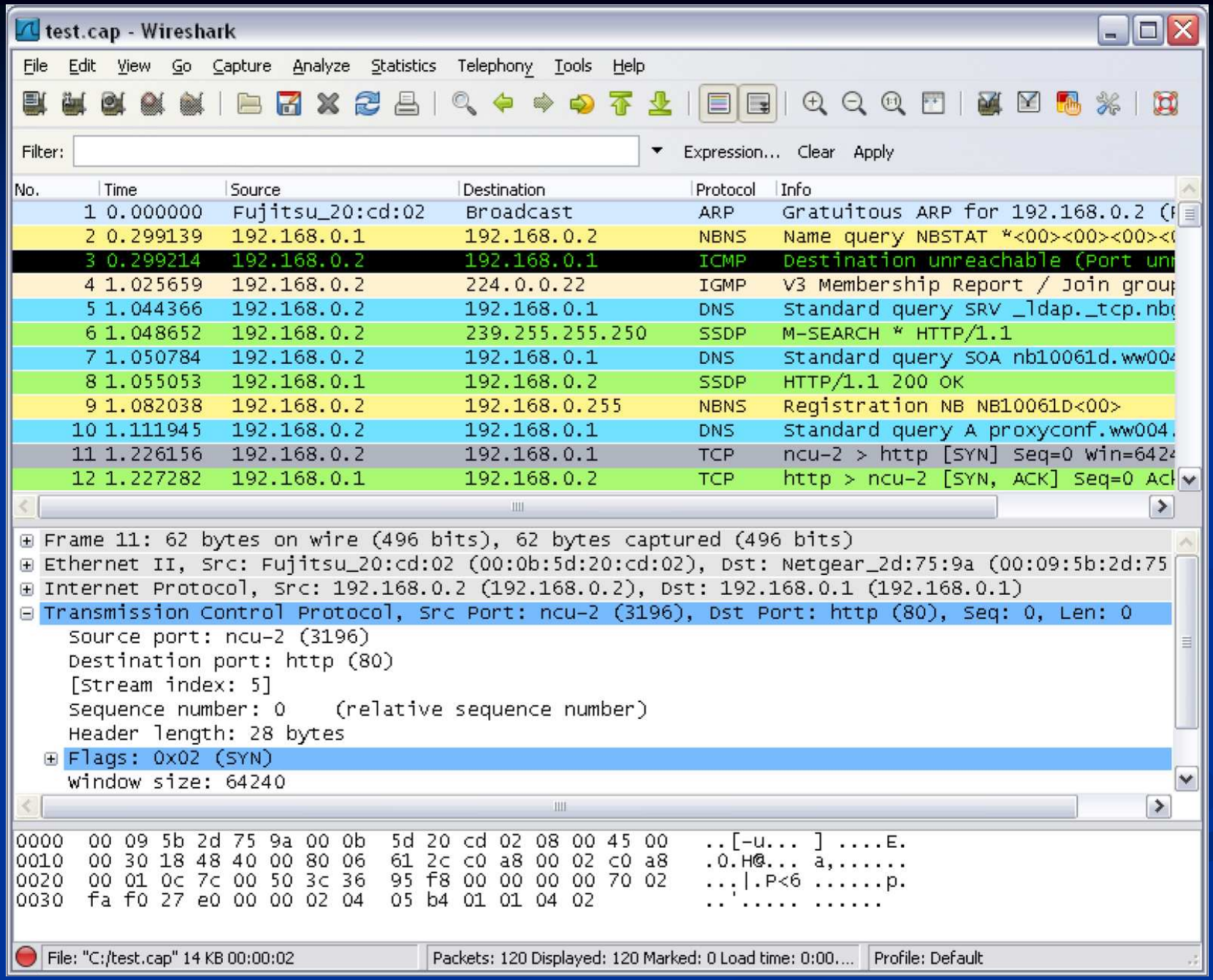
- In late 1997, Gerald Combs needed a tool for tracking down networking problems and wanted to learn more about networking, so he started writing Ethereal (the former name of the Wireshark project) as a way to solve both.
- In 2006 the project moved house and re-emerged under a new name: Wireshark.
- In 2008, after ten years of development, Wireshark finally arrived at version 1.0.

User Interface

- La send si blocca solo quando il buffer in trasmissione associato al socket è pieno.
- Se il socket è impostato come non bloccante ovviamente non ci sarà nessun blocco ma ritornerà un -1 settando la variabile di errore EAGAIN o EWOULDBLOCK a indicare che tale istruzione si sarebbe dovuta bloccare

Menu

- **File** This menu contains items to open and merge capture files, save / print / export capture files in whole or in part, and to quit from Wireshark.
- **Edit** This menu contains items to find a packet, time reference or mark one or more packets, handle configuration profiles, and set your preferences; (cut, copy, and paste are not presently implemented).
- **View** This menu controls the display of the captured data, including colorization of packets, zooming the font, showing a packet in a separate window, expanding and collapsing trees in packet details, ...
- **Go** This menu contains items to go to a specific packet.
- **Capture** This menu allows you to start and stop captures and to edit capture filters. In the packet detail, toggles the selected tree item.
- **Analyze** This menu contains items to manipulate display filters, enable or disable the dissection of protocols, configure user specified decodes and follow a TCP stream.
- **Statistics** This menu contains items to display various statistic windows, including a summary of the packets that have been captured, display protocol hierarchy statistics and much more.
- **Telephony** This menu contains items to display various telephony related statistic windows, including a media analysis, flow diagrams, display protocol hierarchy statistics and much more.
- **Tools** This menu contains various tools available in Wireshark, such as creating Firewall ACL Rules.
- **Help** This menu contains items to help the user.


test.cap - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter: Expression... Clear Apply
No. Time Source Destination Protocol Info
1 0.000000 Fujitsu_20:cd:02 Broadcast ARP Gratuitous ARP for 192.168.0.2
2 0.299139 192.168.0.1 192.168.0.2 NBNS Name query NBSTAT *<00><00><00>
3 0.299214 192.168.0.2 192.168.0.1 ICMP Destination unreachable (Port un
4 1.025659 192.168.0.2 224.0.0.22 IGMP V3 Membership Report / Join grou
5 1.044366 192.168.0.2 192.168.0.1 DNS Standard query SRV _ldap._tcp.nb
6 1.048652 192.168.0.2 239.255.255.250 SSDP M-SEARCH * HTTP/1.1
7 1.050784 192.168.0.2 192.168.0.1 DNS Standard query SOA nb10061d.ww00
8 1.055053 192.168.0.1 192.168.0.2 SSDP HTTP/1.1 200 OK
9 1.082038 192.168.0.2 192.168.0.255 NBNS Registration NB NB10061D<00>
10 1.111945 192.168.0.2 192.168.0.1 DNS Standard query A proxyconf.ww004.
11 1.226156 192.168.0.2 192.168.0.1 TCP ncu-2 > http [SYN] Seq=0 Win=642
12 1.227282 192.168.0.1 192.168.0.2 TCP http > ncu-2 [SYN, ACK] Seq=0 Ac
Frame 11: 62 bytes on wire (496 bits), 62 bytes captured (496 bits)
Ethernet II, Src: Fujitsu_20:cd:02 (00:0b:5d:20:cd:02), Dst: Netgear_2d:75:9a (00:09:5b:2d:75
Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.1 (192.168.0.1)
Transmission Control Protocol, Src Port: ncu-2 (3196), Dst Port: http (80), Seq: 0, Len: 0
Source port: ncu-2 (3196)
Destination port: http (80)
[Stream index: 5]
Sequence number: 0 (relative sequence number)
Header length: 28 bytes
Flags: 0x02 (SYN)
Window size: 64240
0000 00 09 5b 2d 75 9a 00 0b 5d 20 cd 02 08 00 45 00 ..[-u...] ....E.
0010 00 30 18 48 40 00 80 06 61 2c c0 a8 00 02 c0 a8 .0.H@... a,......
0020 00 01 0c 7c 00 50 3c 36 95 f8 00 00 00 00 70 02 ...|.P<6 ......p.
0030 fa f0 27 e0 00 00 02 04 05 b4 01 01 04 02 ..'..... ......
File: "C:/test.cap" 14 KB 00:00:02
Packets: 120 Displayed: 120 Marked: 0 Load time: 0:00....
Profile: Default

Live Capture

Figure 4.1. The "Capture Interfaces" dialog box on Microsoft Windows

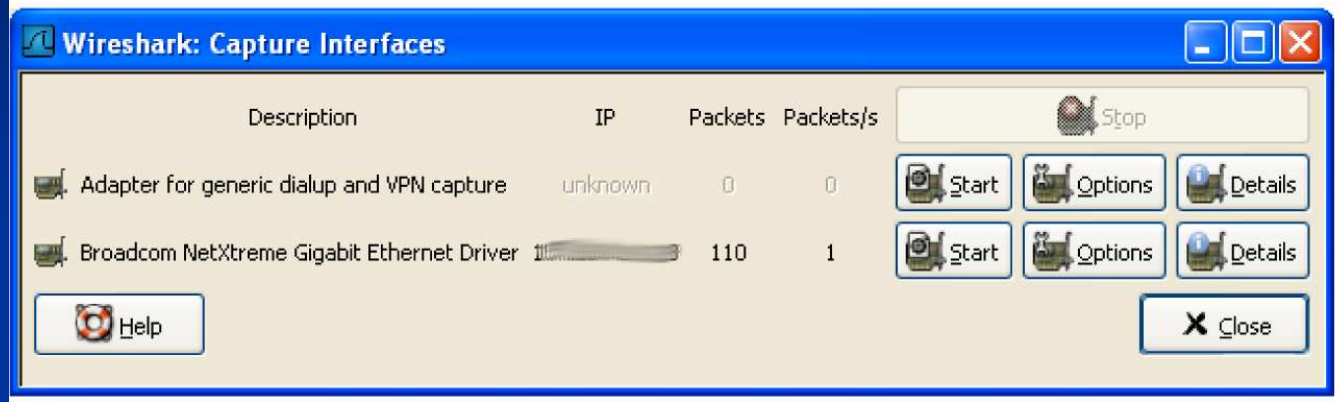


Figure 4.2. The "Capture Interfaces" dialog box on Unix/Linux

Filtering

- Wireshark uses the libpcap filter language for capture filters.
- This is explained in the tcpdump man page, which can be hard to understand
- [not] **primitive** [and|or [not] **primitive** ...]

Filtering (2)

- **[src|dst] host <host>** This primitive allows you to filter on a host IP address or name. You can optionally precede the primitive with the keyword **src|dst** to specify that you are only interested in source or destination addresses. If these are not present, packets where the specified address appears as either the source or the destination address will be selected.
- **ether [src|dst] host <ehost>** This primitive allows you to filter on Ethernet host addresses. You can optionally include the keyword **src|dst** between the keywords **ether** and **host** to specify that you are only interested in source or destination addresses. If these are not present, packets where the specified address appears in either the source or destination address will be selected.
- **gateway host <host>** This primitive allows you to filter on packets that used **host** as a gateway. That is, where the Ethernet source or destination was **host** but neither the source nor destination IP address was **host**.
- **[src|dst] net <net> [{mask <mask>}|{len <len>}]** This primitive allows you to filter on network numbers. You can optionally precede this primitive with the keyword **src| dst** to specify that you are only interested in a source or destination network. If neither of these are present, packets will be selected that have the specified network in either the source or destination address. In addition, you can specify either the netmask or the CIDR prefix for the network if they are different from your own.

Filtering (3)

- **[tcp|udp] [src|dst] port <port>** This primitive allows you to filter on TCP and UDP port numbers. You can optionally precede this primitive with the keywords **src|dst** and **tcp|udp** which allow you to specify that you are only interested in source or destination ports and TCP or UDP packets respectively. The keywords **tcp|udp** must appear before **src|dst**. If these are not specified, packets will be selected for both the TCP and UDP protocols and when the specified address appears in either the source or destination port field.
- **less|greater <length>** This primitive allows you to filter on packets whose length was less than or equal to the specified length, or greater than or equal to the specified length, respectively.
- **ip|ether proto <protocol>** This primitive allows you to filter on the specified protocol at either the Ethernet layer or the IP layer.
- **ether|ip broadcast|multicast** This primitive allows you to filter on either Ethernet or IP broadcasts or multicasts.
- **<expr> relop <expr>** This primitive allows you to create complex filter expressions that select bytes or ranges of bytes in packets. Please see the tcpdump man page at http://www.tcpdump.org/ tcpdump_man.html for more details.

WiFi

- Link Layer (radio) Packet Headers
- Monitor Mode
- Decrypt 802.11
  - Configure in pref pane
    - wpa-psk
    - wpa-pwd
    - Wep
  - http://wiki.wireshark.org/HowToDecrypt802.11
- http://wiki.wireshark.org/Wi-Fi?action=show&redirect=IEEE_802.11

Esercizi

1. Configurare wireshark in modo da poter decifrare la rete wifi-unipi
2. Catturare una serie di pacchetti che utilizzano il protocollo applicazione HTTP
3. Catturare una serie di pacchetti che utilizzano il protocollo applicazione FTP
4. Catturare una serie di pacchetti che utilizzano il protocollo applicazione SSH
5. Catturare una serie di pacchetti che utilizzano il protocollo di trasporto FTP
6. Catturare una serie di pacchetti che utilizzano il protocollo di trasporto UDP
7. Catturare una serie di pacchetti che utilizzano il protocollo ICMP

Esercizi

1. Tramite un hub connettetevi con dei vostri compagni
2. Per ogni hub ci dovrà essere un computer che genera traffico e uno o più computer che cercano di sniffare
3. Gli sniffer dovranno cercare di capire le intenzioni dei generatori di traffico