

# Analisi del Traffico di Rete

L. Massari, G. Nebbione  
giuseppe.nebbione01@universitadipavia.it

University of Pavia

Anno Accademico 2017/2018



# Outline

- 1 Introduzione all'analisi dei pacchetti
- 2 Principi di Packet Capturing
- 3 Wireshark
- 4 Pillole di Networking su Linux
- 5 Riferimenti

# Outline

- 1 Introduzione all'analisi dei pacchetti
- 2 Principi di Packet Capturing
- 3 Wireshark
- 4 Pillole di Networking su Linux
- 5 Riferimenti

# Analisi dei Pacchetti

- Cos'è ?
- Perché ?
  - Analisi del traffico
  - Rilevamento di problemi legati alle reti
  - Riportare statistiche sul traffico
  - Debugging
  - Reverse Engineering di protocolli proprietari

## Analisi dei Pacchetti: Alcune Definizioni

- Packet Analyzer (or Sniffer)
- Packet Capture (or Sniffing)
- Network Interface Card (or NIC)
- Promiscuous Mode & Monitor Mode

# Packet Analyzer

Esempi comuni di Packet Analyzer sono:

- Wireshark (Trattato in questo lab)
- Tshark (versione command line di wireshark)
- tcpdump (altro command line packet sniffer)

# Outline

- 1 Introduzione all'analisi dei pacchetti
- 2 Principi di Packet Capturing**
- 3 Wireshark
- 4 Pillole di Networking su Linux
- 5 Riferimenti

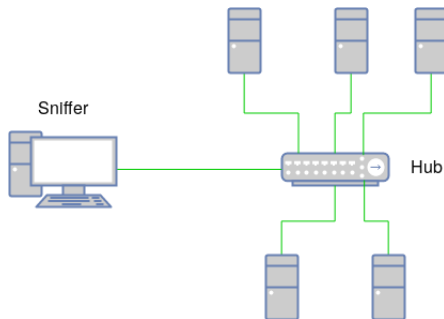
# Posizione del Packet Sniffer

- Sniffing con Hub
- Sniffing con Switch
- Sniffing con Router



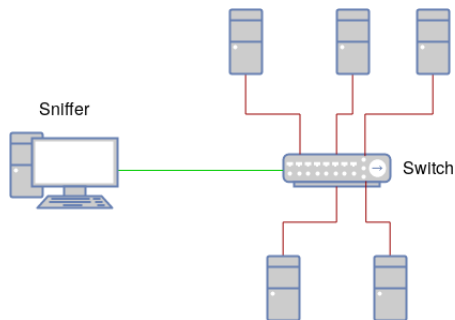
## Sniffing con Hub

- Intera visibilità della rete, il traffico viene mandato a tutti
- Rari da trovare oggi, in genere gli hub non sono performanti



## Sniffing con Switch

- Dispositivo di interconnessione più comune
- Visibilità limitata



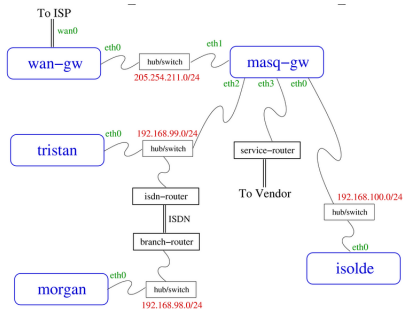
## Sniffing con Switch

Esistono diverse tecniche per eseguire packet capturing in ambienti con switch:

- Port Mirroring (se disponibile)
- Hubbing Out (se il port mirroring non è disponibile)
- TAP (simile ad Hubbing Out ma ottimizzato per l'analisi di rete, più costoso)
- ARP Cache Poisoning (se tutto il resto non è possibile)

## Sniffing con Router

- Con reti grandi, è comodo creare una "network map"
- La rete va analizzata segmento per segmento, per avere una visione dettagliata/completa



# Outline

- 1 Introduzione all'analisi dei pacchetti
- 2 Principi di Packet Capturing
- 3 Wireshark**
- 4 Pillole di Networking su Linux
- 5 Riferimenti

# Wireshark: Main Window

The screenshot displays the Wireshark main window with the following sections:

- Packet List:** A table showing captured packets. The selected packet (No. 1737) is highlighted in blue.
 

No.	Time	Source	Destination	Protocol	Length	Info
5548	42.458055643	193.204.34.20	10.87.174.185	SSH	90	Server: Encrypted packet (len=24)
5544	42.282441440	10.87.174.185	193.204.34.20	SSH	122	Client: Encrypted packet (len=56)
4596	32.190703350	193.204.34.20	10.87.174.185	SSH	90	Server: Encrypted packet (len=24)
4595	32.185189925	10.87.174.185	193.204.34.20	SSH	122	Client: Encrypted packet (len=56)
3251	22.185439787	193.204.34.20	10.87.174.185	SSH	90	Server: Encrypted packet (len=24)
3250	22.183939025	10.87.174.185	193.204.34.20	SSH	122	Client: Encrypted packet (len=56)
1737	17.183497025	193.204.34.20	10.87.174.185	SSH	90	Server: Encrypted packet (len=24)
1736	12.186650812	10.87.174.185	193.204.34.20	SSH	122	Client: Encrypted packet (len=56)
435	2.178162244	193.204.34.20	10.87.174.185	SSH	90	Server: Encrypted packet (len=24)
434	2.167187621	10.87.174.185	193.204.34.20	SSH	122	Client: Encrypted packet (len=56)
- Packet Details:** Shows the structure of the selected packet (No. 1737).
  - Frame 1737: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface 0
  - Ethernet II, Src: ZebraTec\_86:01:30 (40:83:de:86:01:30), Dst: Azurewav\_a8:a8:46 (1c:4b:d6:a8:a8:46)
  - Internet Protocol Version 4, Src: 193.204.34.20, Dst: 10.87.174.185
  - Transmission Control Protocol, Src Port: 22, Dst Port: 39468, Seq: 25, Ack: 113, Len: 24
    - Destination Port: 39468
    - [Stream index: 3]
    - [TCP Segment Len: 24]
    - Sequence number: 25 (relative sequence number)
    - [Next sequence number: 49 (relative sequence number)]
    - Acknowledgment number: 113 (relative ack number)
    - 1860 ... = Header Length: 32 bytes (8)
    - Flags: 0x018 (PSH, ACK)
    - Window size value: 262
    - [Calculated window size: 262]
    - [Window size scaling factor: -1 (unknown)]
    - Checksum: 0x56fd [unverified]
    - [Checksum status: Unverified]
- Packet Bytes:** Shows the raw hexadecimal and ASCII data of the selected packet.
 

```

0000 1c 4b d6 a8 a8 46 02 06 06 01 30 00 00 45 10  ..K..FB...@...E.
0010 00 4c 09 48 48 00 3e 06 06 0b c3 cc 22 14 0a 57  ..L.00>..k...M
0020 ae b0 09 16 9a 2c 05 5e 4b b5 89 80 81 a5 89 18  ....^..K.....
0030 01 00 5e f0 00 01 01 00 0e 07 be 0a 04 01 ae  ..^.....@.....
0040 16 ec 7d 01 34 12 26 81 79 bb 9d 2b ae 67 8e 47  ..].4.&.D..+..G.G
0050 bb fe 4d 3d 4b f2 e9 c4 9e 4e  ..M*...J
      
```

## Wireshark: Tipi di Filtri

- **Capture** Filters, cattura solo determinati pacchetti, utilizza notazione BPF (Berkeley Packet Filter)
- **Display** Filters, mostra solo determinati pacchetti dalla lista dei pacchetti catturati

### Attenzione

Capture filter  $\neq$  Display Filter

# Wireshark Capture Filters: Notazione BPF

- Filtro creato con BPF = espressione
- Una Espressione è composta da una o più primitive legate da operatori
- Ogni primitiva è composta da uno o più qualifier che possono essere di tipo:
  - Type
  - Dir
  - Proto

## Esempio

```
dst host 192.168.0.10 && tcp port 80
```



# Wireshark Display Filters

- Molto più utilizzati in Wireshark rispetto ai capture filter
- Notazione semplice e flessibile

## Esempio

```
tcp.dstport == 112 || tcp.dstport == 113
```

# Outline

- 1 Introduzione all'analisi dei pacchetti
- 2 Principi di Packet Capturing
- 3 Wireshark
- 4 Pillole di Networking su Linux**
- 5 Riferimenti

## Qual'è il mio indirizzo IP?

- ifconfig
- ip a (il comando ip sta sostituendo ifconfig col tempo)

```
eth0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 48:5b:39:4f:e9:41 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 32

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 10458 bytes 11467841 (10.9 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 10458 bytes 11467841 (10.9 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.87.174.185 netmask 255.255.0.0 broadcast 10.87.255.255
    inet6 fe80::1e4b:d6ff:fea8:a846 prefixlen 64 scopeid 0x20<link>
    ether 1c:4b:d6:a8:a8:46 txqueuelen 1000 (Ethernet)
    RX packets 4193834 bytes 834299105 (795.6 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 187973 bytes 25318815 (24.1 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

# Tabella di Routing

- `route -n` (generalmente sono necessari i diritti di root)
- `ip route show`

```
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0          10.87.254.254  0.0.0.0        UG    600    0      0 wlan0
10.87.0.0        0.0.0.0        255.255.0.0    U    600    0      0 wlan0
```

## Quali porte sono aperte?

Molto spesso siamo interessati a capire quali porte sono aperte e quale processo le tiene aperte

- netstat -ntlp
- ss -ntlp

```
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp    0      0 127.0.0.1:6379          0.0.0.0:*                LISTEN     745/redis-server 12
tcp    0      0 0.0.0.0:111            0.0.0.0:*                LISTEN     535/rpcbind
tcp    0      0 0.0.0.0:39857          0.0.0.0:*                LISTEN     -
tcp    0      0 127.0.0.1:5939         0.0.0.0:*                LISTEN     1179/teamviewerd
tcp    0      0 127.0.0.1:34165        0.0.0.0:*                LISTEN     27226/python
tcp    0      0 127.0.0.1:631          0.0.0.0:*                LISTEN     565/cupsd
tcp    0      0 127.0.0.1:5432         0.0.0.0:*                LISTEN     820/postgres
tcp    0      0 0.0.0.0:52189          0.0.0.0:*                LISTEN     708/rpc.mountd
tcp    0      0 0.0.0.0:2049           0.0.0.0:*                LISTEN     -
tcp    0      0 0.0.0.0:39875          0.0.0.0:*                LISTEN     708/rpc.mountd
tcp    0      0 0.0.0.0:46147          0.0.0.0:*                LISTEN     708/rpc.mountd
```

# Netcat

Netcat è il coltellino svizzero dei network-administrator, è utilizzato in svariati ambiti ed è presente sulla maggior parte dei sistemi \*NIX  
Alcune applicazioni sono:

- Network Troubleshooting
- Banner Grabbing (Che servizio è attivo sulla porta x ?)
- Trasferire file
- Redirect di comandi su macchine remote
- Testare software di rete

# Netcat

## Basi di netcat

- Aprire un server TCP sulla porta 3333
  - `nc -l 3333`
- Connettersi ad un server TCP sulla porta 3333
  - `nc 127.0.0.1 3333`
- Aprire un server UDP sulla porta 4444
  - `nc -u -l 4444`
- Connettersi ad un server UDP sulla porta 4444
  - `nc -u 127.0.0.1 4444`

# Outline

- 1 Introduzione all'analisi dei pacchetti
- 2 Principi di Packet Capturing
- 3 Wireshark
- 4 Pillole di Networking su Linux
- 5 Riferimenti**



# Riferimenti



Practical Packet Analysis 2Ed, Sanders



Mastering Linux Network Administration, Jay LaCroix



Netcat Power Tools, Jan Kanclirz