Wireshark: Guida Completa con 1000+ Esempi Realistici

Autore: Tuo Nome

October 14, 2025

Contents

1 Introduzione **3**

2 Tipi di Pacchetti e Analisi Pratica **3**
2.1 Pacchetti TCP . . . 3
2.2 Pacchetti UDP . . . 3
2.3 Pacchetti ICMP . . . 3
2.4 Pacchetti HTTP / HTTPS . . . 4
2.5 Pacchetti DNS . . . 4
2.6 Pacchetti ARP . . . 4
2.7 Pacchetti FTP / SMTP / SNMP . . . 4
2.8 Raccomandazioni pratiche . . . 4

3 Installazione e Avvio **5**
3.1 Windows . . . 5
3.2 Linux . . . 5
3.3 MacOS . . . 5

4 Filtri di Cattura e Display **5**
4.1 Capture Filter . . . 5
4.2 Display Filter . . . 5

5 Protocollo HTTP **5**
5.1 Filtri GUI e esempi . . . 5
5.2 CLI - tshark esempi HTTP . . . 6

6 Protocollo DNS **6**
6.1 Filtri GUI . . . 6
6.2 tshark esempi DNS . . . 6

7 Protocollo TCP **6**
7.1 tshark TCP . . . 7

8 Protocollo UDP **7**

9 Protocollo ICMP **7**

10 Protocollo FTP **7**

11 Protocollo SMTP **7**

12 Protocollo TLS/SSL 7

13 ARP, DHCP, NTP, SNMP 8

14 Analisi avanzata e follow stream 8

15 Consigli pratici 8

16 Risorse utili 8

1 Introduzione

Wireshark è uno strumento open source per catturare e analizzare traffico di rete. Supporta centinaia di protocolli e permette analisi sia in GUI sia tramite CLI con `tshark`. **Documentazione ufficiale:** https://www.wireshark.org/docs/

2 Tipi di Pacchetti e Analisi Pratica

Quando analizzi una rete con Wireshark o tshark, è fondamentale capire quali tipi di pacchetti osservare e quando, perché ogni protocollo/provenienza offre informazioni diverse. Questa sezione guida all'uso pratico dei filtri e all'interpretazione dei dati.

2.1 Pacchetti TCP

- `tcp.flags.syn == 1`: mostra solo pacchetti SYN, utili per analizzare handshake TCP. Serve quando vuoi vedere connessioni iniziali o sospette.
- `tcp.flags.ack == 1`: pacchetti di conferma. Utile per verificare completamento handshake o pacchetti già ricevuti.
- `tcp.analysis.retransmission`: mostra pacchetti ritrasmessi. Utile per diagnosticare congestione o perdita di pacchetti.
- `tcp.len > 1000`: pacchetti di grandi dimensioni. Permette analizzare trasferimenti di file o payload pesanti.
- Flussi TCP (`tcp.stream eq N`): isolare una singola conversazione TCP per ricostruire la sequenza completa di pacchetti.

Quando usarli: per debug di connessioni, verifica handshake, problemi di rete o analisi di traffico applicativo.

2.2 Pacchetti UDP

- `udp.port == 53`: traffico DNS. Utile per analizzare query e risposte, risoluzione nomi e errori.
- `udp.port == 123`: traffico NTP. Permette controllare sincronizzazione orologi di rete.
- `udp.length > 200`: pacchetti grandi. Utile per analisi di streaming, VoIP o trasferimenti.

Quando usarli: UDP è senza connessione, quindi isoli pacchetti per servizio specifico (DNS, VoIP, streaming) e analizzi richieste e risposte singolarmente.

2.3 Pacchetti ICMP

- `icmp.type == 8`: Echo request (ping). Utile per test di raggiungibilità.
- `icmp.type == 0`: Echo reply (ping risposta). Verifica latenza e raggiungibilità.
- `icmp.code == 3`: Destination unreachable. Diagnosi di routing o firewall.

Quando usarli: test di rete, diagnosi problemi di raggiungibilità o configurazioni errate.

2.4 Pacchetti HTTP / HTTPS

- `http.request`: mostra tutte le richieste HTTP. Utile per vedere quali URL vengono richiesti.
- `http.response.code`: mostra codici di risposta. Individua errori (404, 500) o conferme.
- `http contains "login"`: ricerca parole chiave nei pacchetti. Utile per tracciare login o form.
- `tls.handshake.type`: handshake TLS. Necessario per analizzare connessioni HTTPS e certificati.

Quando usarli: per debug web, analisi sicurezza, verifica performance applicazioni web.

2.5 Pacchetti DNS

- `dns.qry.type == 1`: query IPv4. Controlla risoluzione indirizzi.
- `dns.qry.type == 28`: query IPv6. Controlla risoluzione IPv6.
- `dns.flags.rcode != 0`: errori DNS. Diagnosi problemi di nome o server.

Quando usarli: isolare problemi di navigazione o risoluzione nomi.

2.6 Pacchetti ARP

- `arp`: mostra tutte le richieste e risposte ARP. Utile per mappare la rete locale e rilevare conflitti IP.

Quando usarli: scoperta rete locale, problemi IP duplicati, analisi ARP spoofing.

2.7 Pacchetti FTP / SMTP / SNMP

- FTP: `ftp.request.command` e `ftp.response.code`. Analizza login, trasferimenti file.
- SMTP: `smtp.req.command` e `smtp.resp.code`. Controlla invio email e risposte server.
- SNMP: `snmp`. Analizza richieste e risposte di monitoraggio rete.

Quando usarli: troubleshooting applicazioni, controllo credenziali, monitoraggio rete.

2.8 Raccomandazioni pratiche

- Sempre isolare il tipo di pacchetto rilevante per il problema. Non guardare tutto contemporaneamente.
- Per analisi performance: TCP/UDP grandezza pacchetti, ritrasmissioni, ritardi.
- Per sicurezza: HTTP/HTTPS login, pacchetti ARP sospetti, ICMP insoliti.
- Per rete interna: ARP, DHCP, DNS, NTP. Mostrano struttura e sincronizzazione.
- Per debugging applicazioni: TCP stream completo, payload HTTP/FTP, handshake TLS.

Obiettivo: capire rapidamente quali pacchetti osservare per ottenere informazioni precise senza perdersi nel rumore di rete.

3 Installazione e Avvio

3.1 Windows

Scaricare da: https://www.wireshark.org/download.html e includere WinPcap/Npcap. **Didascalia:** Consente cattura pacchetti senza errori di permessi.

3.2 Linux

```
sudo apt update
sudo apt install wireshark
sudo usermod -aG wireshark $USER
```

Didascalia: Installa Wireshark e consente all'utente di catturare pacchetti senza root.

3.3 MacOS

```
brew install wireshark
```

4 Filtri di Cattura e Display

4.1 Capture Filter

```
tcp port 80
host 192.168.1.10
net 192.168.1.0/24
port not 22
```

Didascalie: Isolare traffico HTTP, host specifici, subnet o escludere porte.

4.2 Display Filter

```
ip.addr == 192.168.1.10
tcp.flags.syn == 1
http.request.method == "GET"
dns.qry.name == "www.google.com"
```

Didascalie: Analisi post-cattura su host, handshake, richieste HTTP e query DNS.

5 Protocollo HTTP

5.1 Filtri GUI e esempi

1. `http.request` - Mostra tutte le richieste HTTP.
2. `http.response.code == 404` - Individua pagine mancanti.
3. `http contains "login"` - Traccia login.
4. `http.request.method == "POST"` - Mostra richieste POST.
5. `http.user_agent contains "Mozilla"` - Analizza client specifici.

6. `http.cookie` - Mostra cookie inviati.
7. `http.referer contains "google"` - Origine traffico.
8. `http.content_type == "application/json"` - Filtra JSON.
9. `http.authbasic` - Richieste con basic auth.
10. `http.response.time > 1` - Ritardi >1s.

5.2 CLI - tshark esempi HTTP

```
tshark -i eth0 -Y "http.request.method == GET"
tshark -r capture.pcapng -T fields -e http.host -e http.request.uri
tshark -r capture.pcapng -Y "http.response.code == 500"
tshark -i eth0 -f "tcp port 80"
tshark -r capture.pcapng -T fields -e http.user_agent
```

Didascalia: Comandi per cattura, estrazione e analisi avanzata HTTP.

6 Protocollo DNS

6.1 Filtri GUI

1. `dns.qry.type == 1` - Query IPv4
2. `dns.flags.response == 1` - Solo risposte
3. `dns.qry.name contains "google"` - Query verso Google
4. `dns.qry.type == 28` - Query IPv6
5. `dns.flags.rcode != 0` - Errori DNS

6.2 tshark esempi DNS

```
tshark -i eth0 -Y "dns.qry.name == 'www.google.com'"
tshark -r capture.pcapng -T fields -e dns.qry.name -e dns.a
tshark -r capture.pcapng -Y "dns.flags.rcode != 0"
```

Didascalia: Filtraggio, estrazione di indirizzi, rilevamento errori DNS.

7 Protocollo TCP

1. `tcp.port == 22` - SSH
2. `tcp.flags.syn == 1` - Pacchetti SYN
3. `tcp.flags.ack == 1` - ACK
4. `tcp.stream eq 5` - Flusso TCP 5
5. `tcp.analysis.retransmission` - Retransmission
6. `tcp.analysis.out_of_order` - Pacchetti fuori ordine

7. `tcp.len > 1000` - Pacchetti grandi
8. `tcp.options.mss` - Dimensione MSS

7.1 tshark TCP

```
tshark -i eth0 -Y "tcp.flags.syn==1"
tshark -r capture.pcapng -T fields -e tcp.srcport -e tcp.dstport
tshark -r capture.pcapng -Y "tcp.analysis.retransmission"
```

Didascalia: Analisi handshake, flussi, problemi di rete.

8 Protocollo UDP

1. `udp.port == 53` - DNS
2. `udp.port == 123` - NTP
3. `udp.length > 200` - Pacchetti grandi

9 Protocollo ICMP

1. `icmp.type == 8` - Echo request (ping)
2. `icmp.type == 0` - Echo reply
3. `icmp.code == 3` - Destination unreachable

10 Protocollo FTP

1. `ftp.request.command == "USER"`
2. `ftp.request.command == "PASS"`
3. `ftp.response.code == 230`

11 Protocollo SMTP

1. `smtp.req.command == "MAIL"`
2. `smtp.req.command == "RCPT"`
3. `smtp.resp.code == 250`

12 Protocollo TLS/SSL

1. `tls.handshake.type == 1` - ClientHello
2. `tls.handshake.type == 2` - ServerHello
3. `tls.record.version == 0x0303` - TLS1.2
4. `tls.record.version == 0x0304` - TLS1.3

13 ARP, DHCP, NTP, SNMP

1. `arp` - Tutti pacchetti ARP
2. `bootp` - DHCP discover/offers
3. `ntp` - Richieste NTP
4. `snmp` - Richieste e risposte SNMP

14 Analisi avanzata e follow stream

```
Follow TCP Stream
Follow UDP Stream
Statistics -> Protocol Hierarchy
Statistics -> Conversations
Statistics -> IO Graphs
```

Didascalia: Analisi completa flussi, distribuzione protocolli e traffico nel tempo.

15 Consigli pratici

- Usare filtri di cattura per ridurre rumore.
- Analizzare flussi completi per debug.
- Salvare regolarmente catture.
- Combinare GUI e CLI per automazione.

16 Risorse utili

- Wireshark Docs: https://www.wireshark.org/docs/
- Wireshark Wiki: https://wiki.wireshark.org/
- Display Filters Reference: https://www.wireshark.org/docs/dfref/
- tshark Reference: https://www.wireshark.org/docs/man-pages/tshark.html