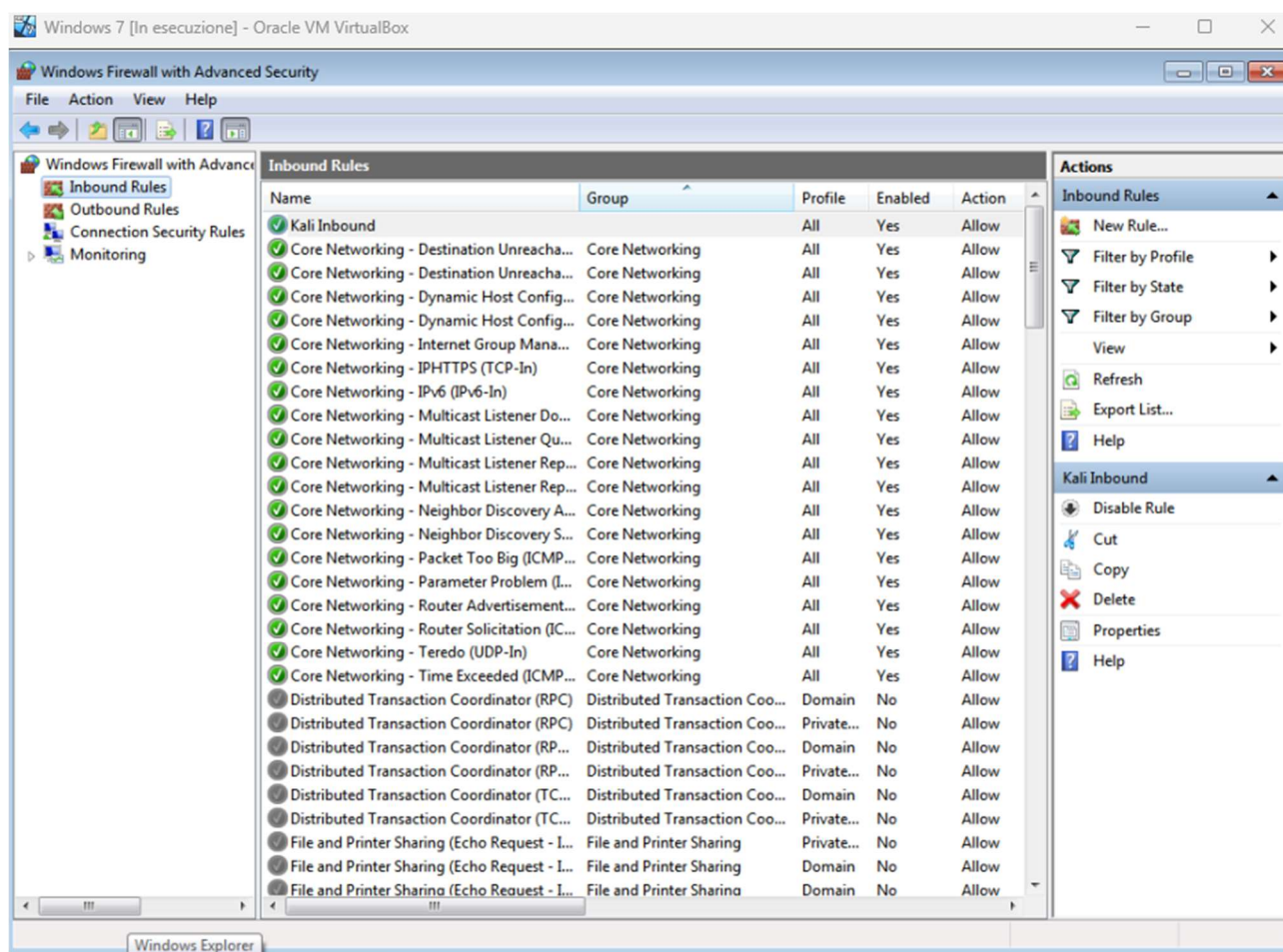


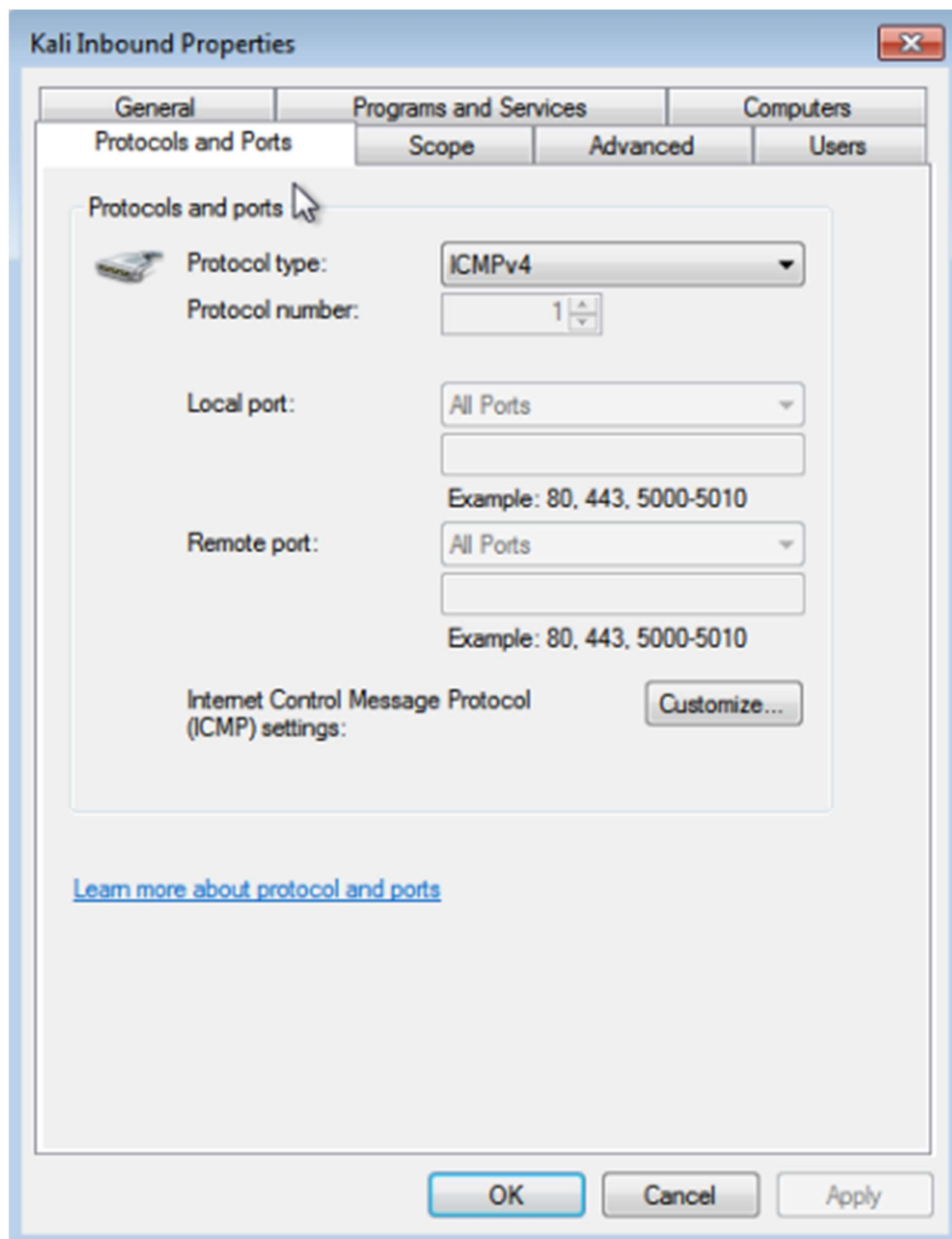
UTILIZZO DI FIREWALL WINDOWS E WIRESHARK

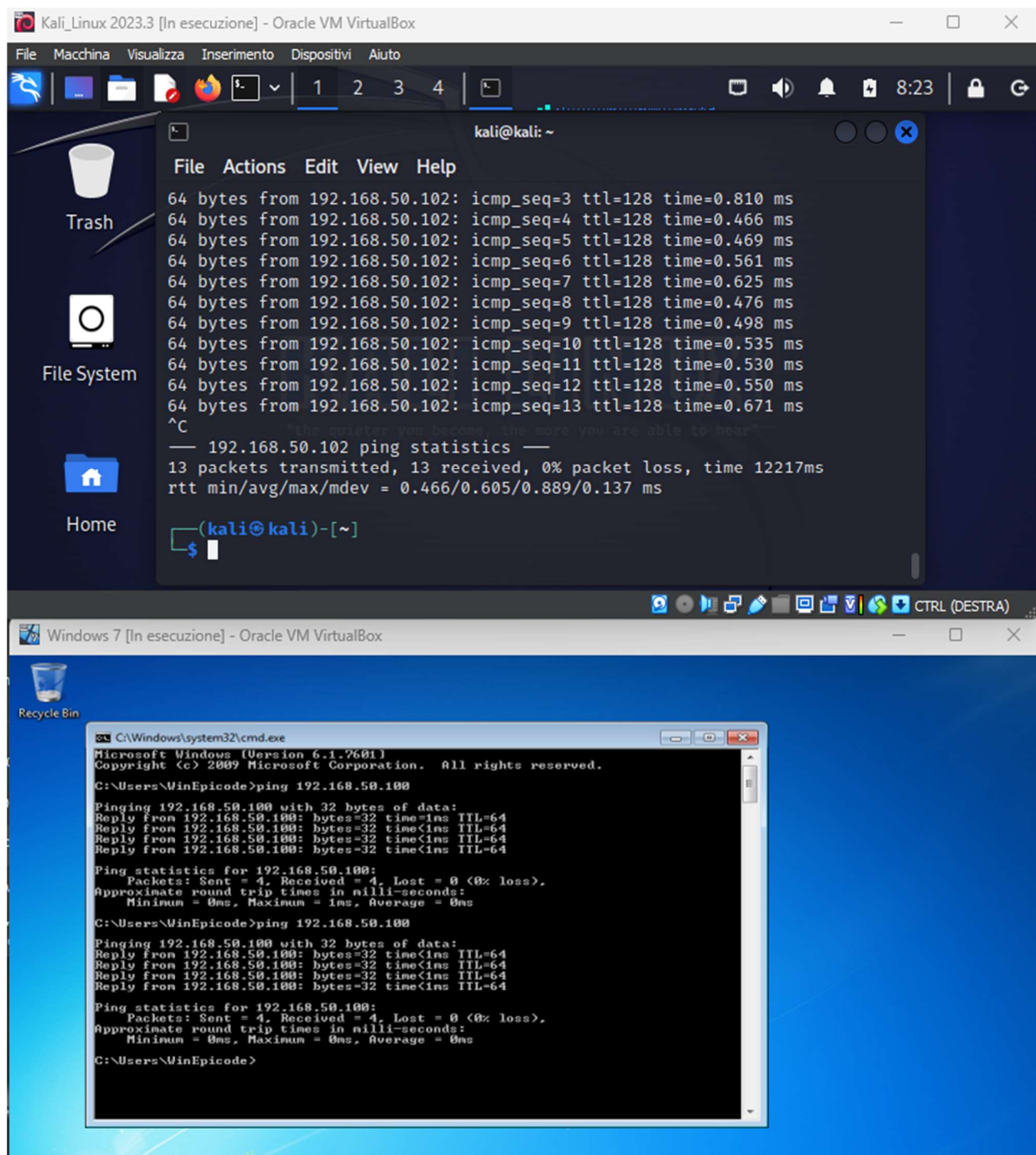
CONFIGURARE POLICY PER PERMETTERE IL PING DA FRA MACCHINE WINDOWS 7 E KALI LINUX.

Entro nella macchina virtuale di Windows 7 e accedo al Firewall di sistema per configurare una regola che permette la comunicazione TCP fra Windows e Kali Linux



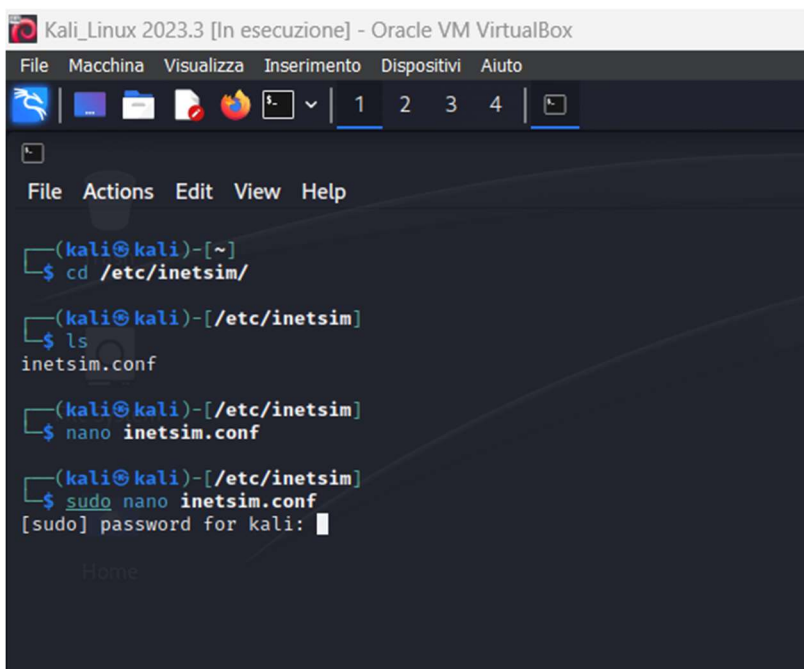
Ho creato una regola sia in ingresso che in uscita dalla macchina Windows7. Dopo di che sono entrato nelle Proprietà della regola e ho impostato il protocollo ICMPv4 (premetto che sulla pagina di creazione questa voce non era presente fra le opzioni e ho dovuto fare così). Successivamente ho lanciato il comando PING su entrambe le macchine per verificarne la corretta configurazione.





UTILIZZO DELLA UTILITY DELL'INETSIM PER L'EMULAZIONE DI SERVIZI INTERNET

Avvio la macchina virtuale Kali-Linux e command prompt per accedere alla modifica del file inetsim.conf



```
Kali_Linux 2023.3 [In esecuzione] - Oracle VM VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
1 2 3 4

File Actions Edit View Help

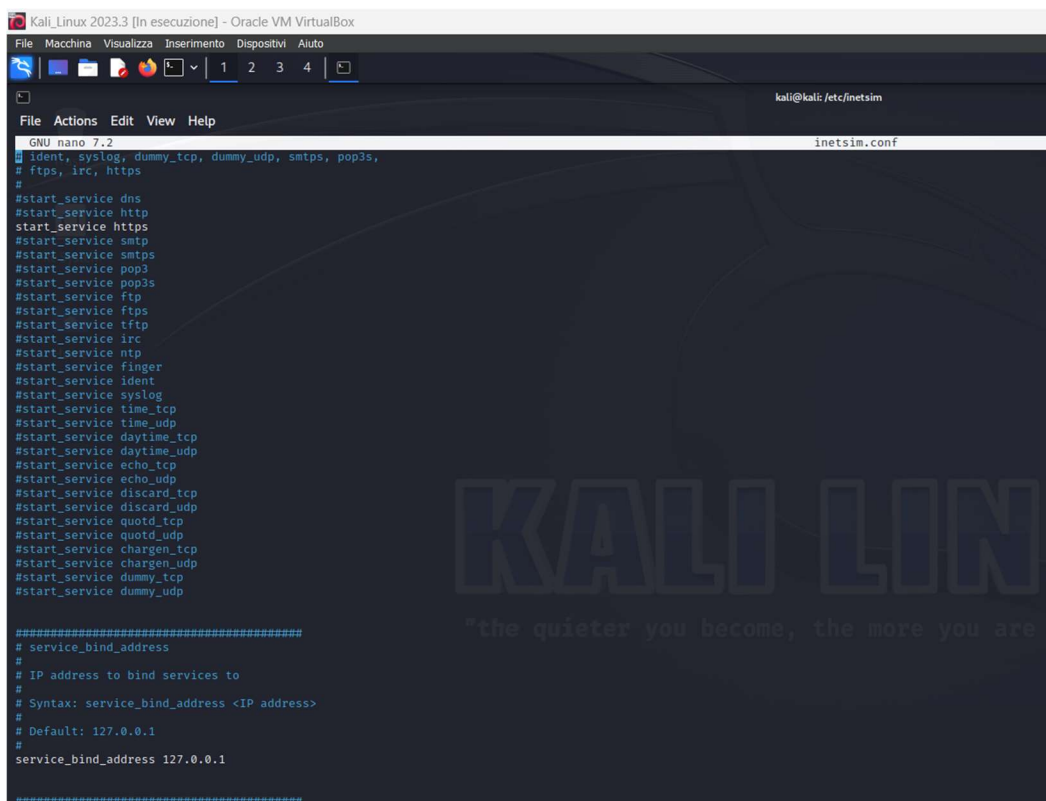
(kali@kali)-[~]
$ cd /etc/inetsim/

(kali@kali)-[/etc/inetsim]
$ ls
inetsim.conf

(kali@kali)-[/etc/inetsim]
$ nano inetsim.conf

(kali@kali)-[/etc/inetsim]
$ sudo nano inetsim.conf
[sudo] password for kali: 
```

Per praticità, mostro la schermata con la modifica dei servizi offerti solo su http e https. Forzo la porta su quella di localhost=127.0.0.1



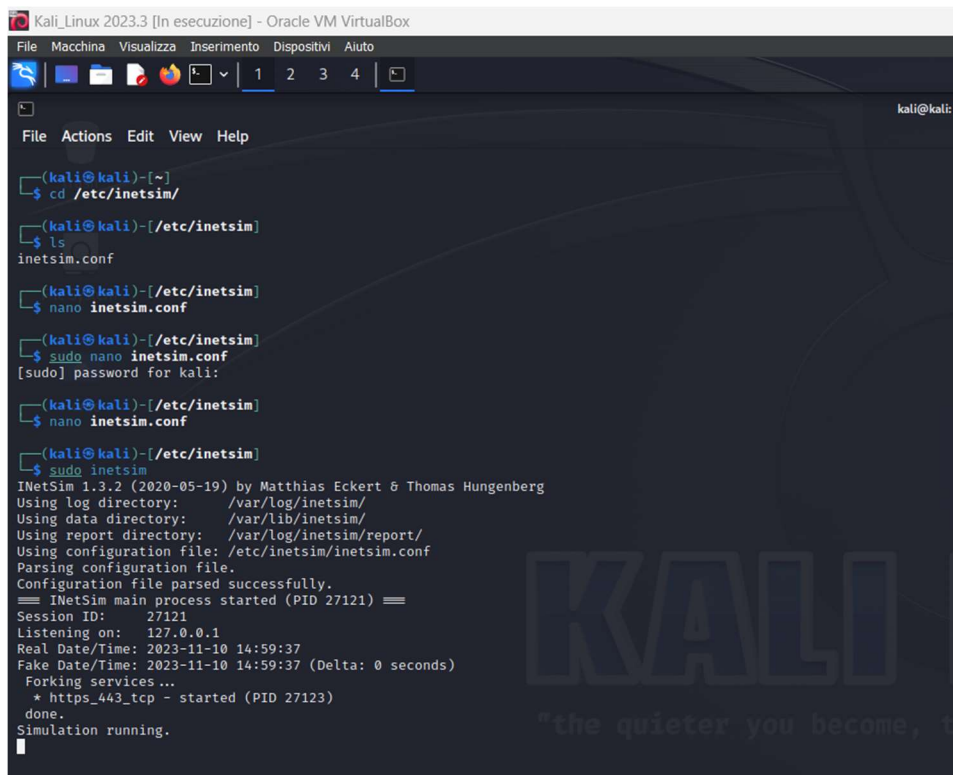
```
Kali_Linux 2023.3 [In esecuzione] - Oracle VM VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
1 2 3 4

File Actions Edit View Help

GNU nano 7.2 inetsim.conf
# ident, syslog, dummy_tcp, dummy_udp, smtps, pop3s,
# ftps, irc, https
#
#start_service dns
#start_service http
start_service https
#start_service smtp
#start_service smtps
#start_service pop3
#start_service pop3s
#start_service ftp
#start_service ftps
#start_service tftp
#start_service irc
#start_service ntp
#start_service finger
#start_service ident
#start_service syslog
#start_service time_tcp
#start_service time_udp
#start_service daytime_tcp
#start_service daytime_udp
#start_service echo_tcp
#start_service echo_udp
#start_service discard_tcp
#start_service discard_udp
#start_service quotd_tcp
#start_service quotd_udp
#start_service chargen_tcp
#start_service chargen_udp
#start_service dummy_tcp
#start_service dummy_udp

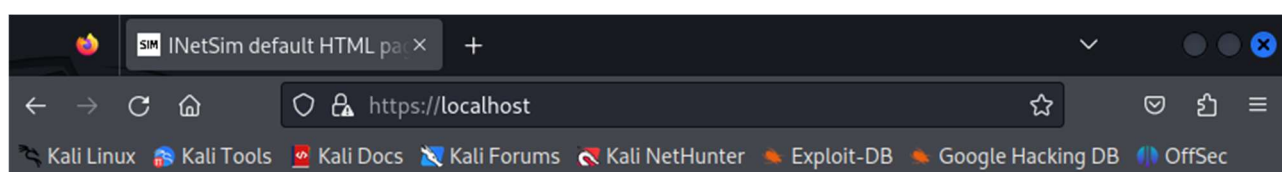
#####
# service_bind_address
#
# IP address to bind services to
#
# Syntax: service_bind_address <IP address>
#
# Default: 127.0.0.1
#
service_bind_address 127.0.0.1
#####
```

Una volta salvato ed effettuato la modifica (ovviamente il file aperto con il comando `sudo`, altrimenti non sarebbe possibile salvarlo) avvio il servizio InetSim tramite comando **`sudo inetsim`**.



```
Kali_Linux 2023.3 [In esecuzione] - Oracle VM VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
1 2 3 4
File Actions Edit View Help
(kali@kali)~$ cd /etc/inetsim/
(kali@kali)~/etc/inetsim$ ls
inetsim.conf
(kali@kali)~/etc/inetsim$ nano inetsim.conf
(kali@kali)~/etc/inetsim$ sudo nano inetsim.conf
[sudo] password for kali:
(kali@kali)~/etc/inetsim$ nano inetsim.conf
(kali@kali)~/etc/inetsim$ sudo inetsim
INetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
Using log directory: /var/log/inetsim/
Using data directory: /var/lib/inetsim/
Using report directory: /var/log/inetsim/report/
Using configuration file: /etc/inetsim/inetsim.conf
Parsing configuration file.
Configuration file parsed successfully.
== INetSim main process started (PID 27121) ==
Session ID: 27121
Listening on: 127.0.0.1
Real Date/Time: 2023-11-10 14:59:37
Fake Date/Time: 2023-11-10 14:59:37 (Delta: 0 seconds)
Forking services...
* https_443_tcp - started (PID 27123)
done.
Simulation running.
```

Per verificare che la sia effettivamente avviato, apro una pagina web puntando la ricerca su <https://localhost>.

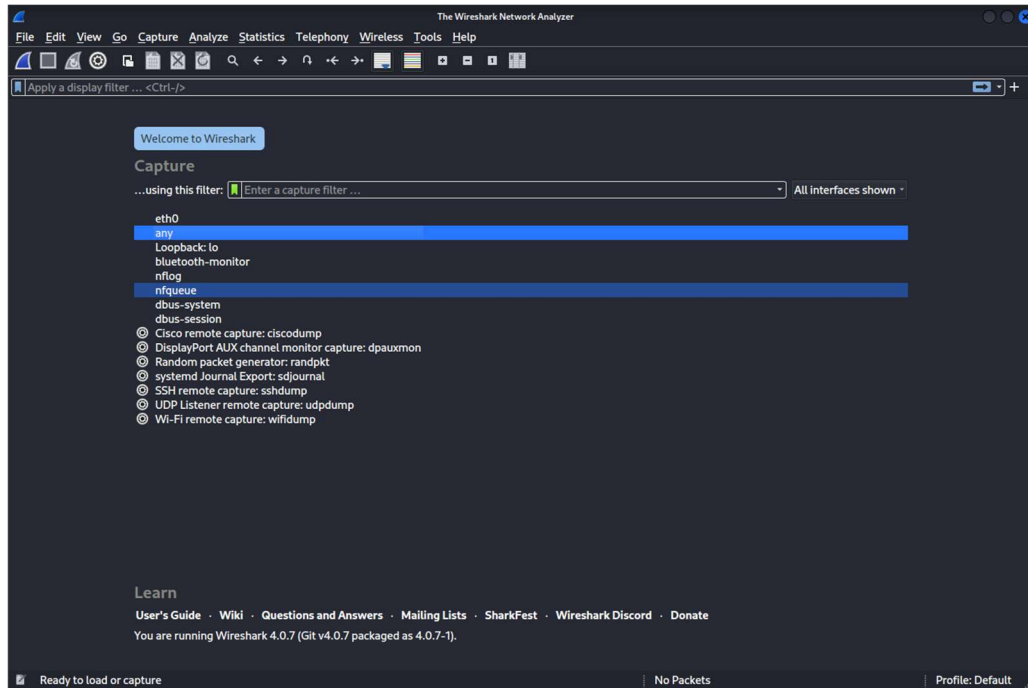


This is the default HTML page for INetSim HTTP server fake mode.

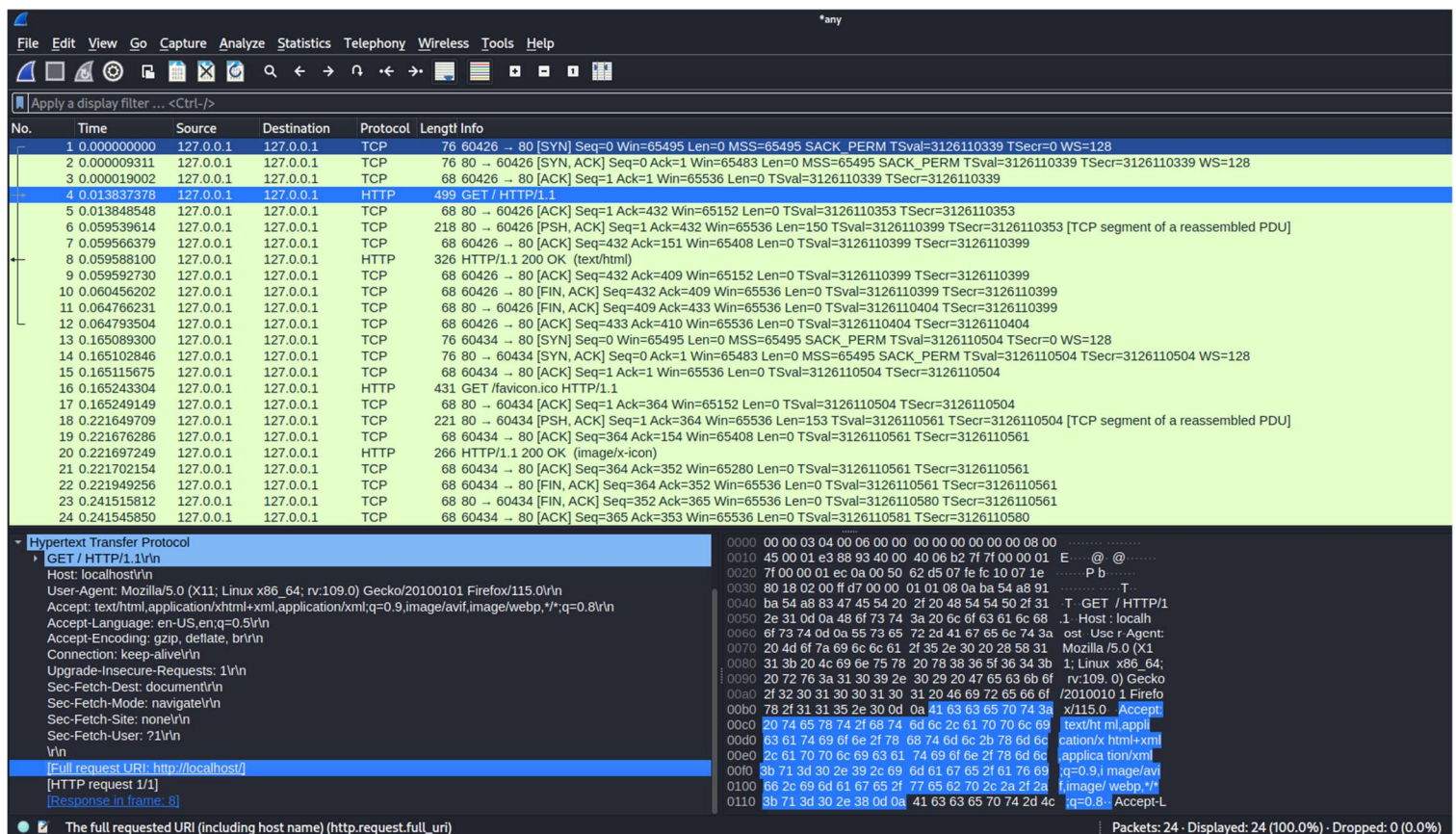
This file is an HTML document.

CATTURA DI PACCHETTI CON WIRESHARK

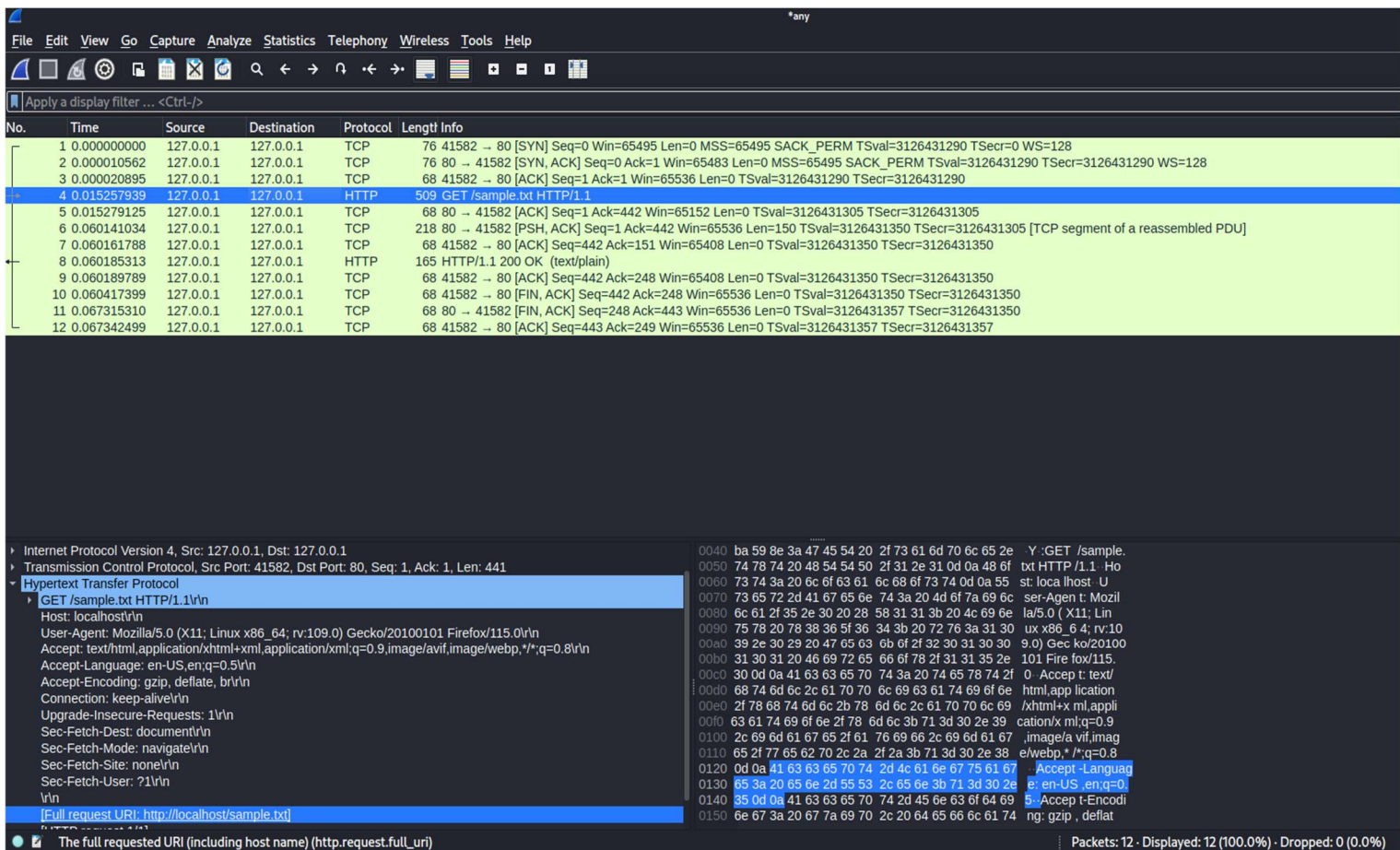
Lasciando configurato InetSim come in precedenza, ho avviato una pagina di ricerca in Kali-linux e avvia Wireshark in ascolto su tutte le porte di rete configurate.



Quello che riporto qui sotto è la cattura dei pacchetti ricercando sul browser <http://localhost> ed evidenziando il pacchetto GET.



In questa schermata dopo ho effettuato una nuova ricerca su un file presente in localhost che è sample.txt sempre sotto protocollo http. In pratica ho ricercato questo: <http://localhost/sample.txt>



The screenshot shows a Wireshark capture of network traffic. The packet list on the left shows 12 packets. Packet 4 is selected, which is an HTTP GET request to `http://localhost/sample.txt`. The packet details pane on the right shows the structure of the request, including the Host, User-Agent, Accept, Accept-Encoding, Connection, Upgrade-Insecure-Requests, Sec-Fetch-Dest, Sec-Fetch-Mode, Sec-Fetch-Site, Sec-Fetch-User, and the full request URI.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	76	41582 → 80 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=3126431290 TSecr=0 WS=128
2	0.000010562	127.0.0.1	127.0.0.1	TCP	76	80 → 41582 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM TSval=3126431290 TSecr=3126431290 WS=128
3	0.000020895	127.0.0.1	127.0.0.1	TCP	68	41582 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=3126431290 TSecr=3126431290
4	0.015257939	127.0.0.1	127.0.0.1	HTTP	509	GET /sample.txt HTTP/1.1
5	0.015279125	127.0.0.1	127.0.0.1	TCP	68	80 → 41582 [ACK] Seq=1 Ack=442 Win=65152 Len=0 TSval=3126431305 TSecr=3126431305
6	0.060141034	127.0.0.1	127.0.0.1	TCP	218	80 → 41582 [PSH, ACK] Seq=1 Ack=442 Win=65536 Len=150 TSval=3126431350 TSecr=3126431305 [TCP segment of a reassembled PDU]
7	0.060161788	127.0.0.1	127.0.0.1	TCP	68	41582 → 80 [ACK] Seq=442 Ack=151 Win=65408 Len=0 TSval=3126431350 TSecr=3126431350
8	0.060185313	127.0.0.1	127.0.0.1	HTTP	165	HTTP/1.1 200 OK (text/plain)
9	0.060189789	127.0.0.1	127.0.0.1	TCP	68	41582 → 80 [ACK] Seq=442 Ack=248 Win=65408 Len=0 TSval=3126431350 TSecr=3126431350
10	0.060417399	127.0.0.1	127.0.0.1	TCP	68	41582 → 80 [FIN, ACK] Seq=442 Ack=248 Win=65536 Len=0 TSval=3126431350 TSecr=3126431350
11	0.067315310	127.0.0.1	127.0.0.1	TCP	68	80 → 41582 [FIN, ACK] Seq=248 Ack=443 Win=65536 Len=0 TSval=3126431357 TSecr=3126431350
12	0.067342499	127.0.0.1	127.0.0.1	TCP	68	41582 → 80 [ACK] Seq=443 Ack=249 Win=65536 Len=0 TSval=3126431357 TSecr=3126431357

Packet 4 Details:

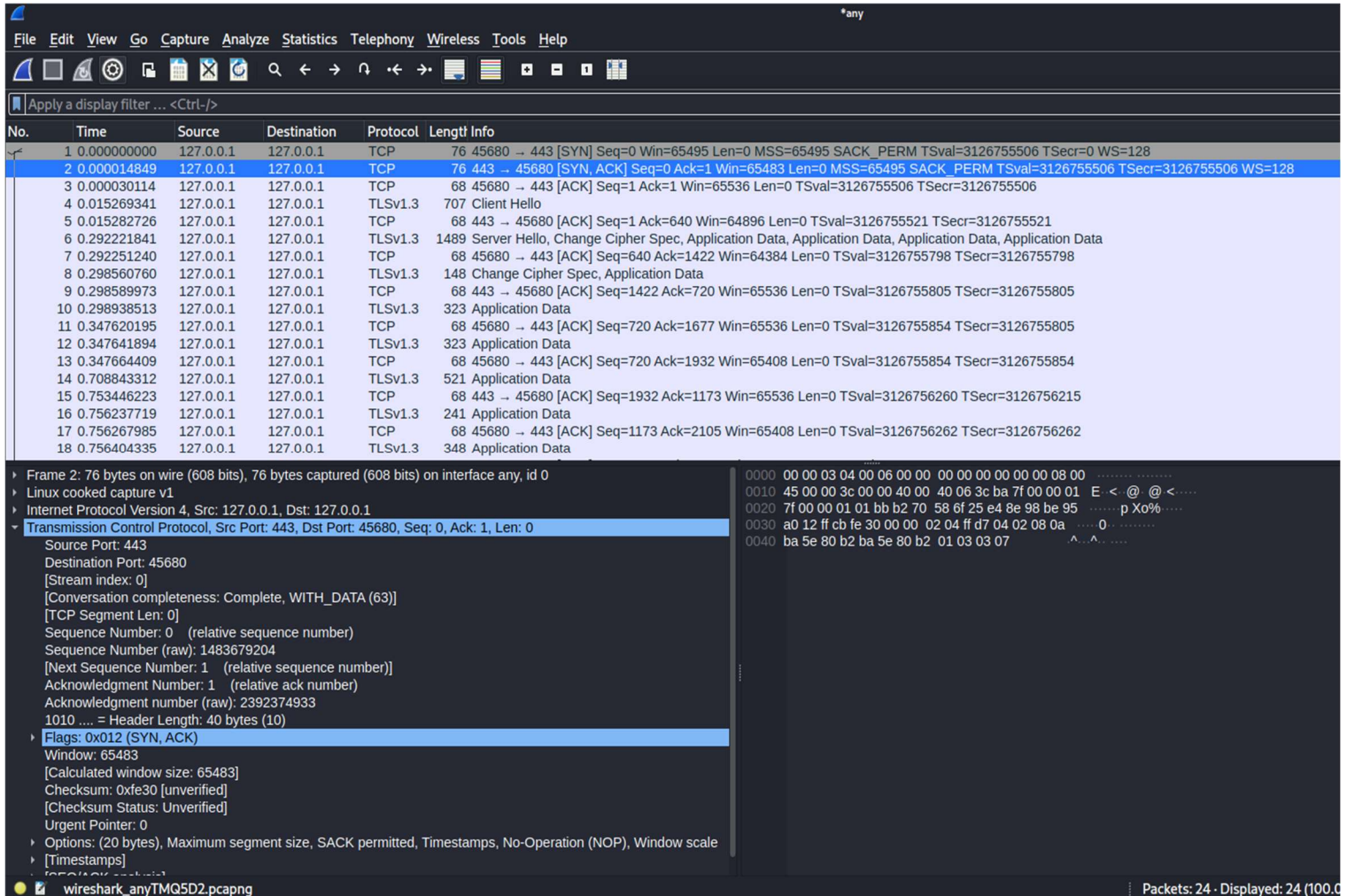
- Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
- Transmission Control Protocol, Src Port: 41582, Dst Port: 80, Seq: 1, Ack: 1, Len: 441
- Hypertext Transfer Protocol
 - GET /sample.txt HTTP/1.1
 - Host: localhost
 - User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
 - Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
 - Accept-Language: en-US,en;q=0.5
 - Accept-Encoding: gzip, deflate, br
 - Connection: keep-alive
 - Upgrade-Insecure-Requests: 1
 - Sec-Fetch-Dest: document
 - Sec-Fetch-Mode: navigate
 - Sec-Fetch-Site: none
 - Sec-Fetch-User: ?1
- Full request URI: <http://localhost/sample.txt>

The full requested URI (including host name) (`http.request.full_uri`)

Packets: 12 - Displayed: 12 (100.0%) - Dropped: 0 (0.0%)

Quando si utilizza protocollo HTTP i pacchetti sono scritti in chiaro e quindi posso vederne il contenuto che il client ha richiesto al server.

Dopo di questa ho fatto la cattura dei pacchetti con protocollo HTTPS.
 Riparto ricercando sul browser questa volta con: <https://localhost>



Frame 2: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface any, id 0

Linux cooked capture v1

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

Transmission Control Protocol, Src Port: 443, Dst Port: 45680, Seq: 0, Ack: 1, Len: 0

Source Port: 443
 Destination Port: 45680
 [Stream index: 0]
 [Conversation completeness: Complete, WITH_DATA (63)]
 [TCP Segment Len: 0]
 Sequence Number: 0 (relative sequence number)
 Sequence Number (raw): 1483679204
 [Next Sequence Number: 1 (relative sequence number)]
 Acknowledgment Number: 1 (relative ack number)
 Acknowledgment number (raw): 2392374933
 1010 = Header Length: 40 bytes (10)

Flags: 0x012 (SYN, ACK)
 Window: 65483
 [Calculated window size: 65483]
 Checksum: 0xfe30 [unverified]
 [Checksum Status: Unverified]
 Urgent Pointer: 0

Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
 [Timestamps]

wireshark_anyTMQ5D2.pcapng

Packets: 24 · Displayed: 24 (100%)

Ovviamente il discorso qui cambia in quanto si usa il protocollo HTTPS che cifra il contenuto del pacchetto in viaggio e di fatto non mi è possibile verificarne il contenuto che sta transitando. Di seguito provo a ricercare <https://localhost/sample.txt>

Wireshark interface showing a packet capture on interface any, id 0. The capture is filtered by 'any'.

Packet List:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	76	49668 → 443 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=3127647582 TSecr=0 WS=128
2	0.000012843	127.0.0.1	127.0.0.1	TCP	76	443 → 49668 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM TSval=3127647582 TSecr=3127647582 WS=128
3	0.000027479	127.0.0.1	127.0.0.1	TCP	68	49668 → 443 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=3127647582 TSecr=3127647582
4	0.003216464	127.0.0.1	127.0.0.1	TLSv1.3	707	Client Hello
5	0.003225107	127.0.0.1	127.0.0.1	TCP	68	443 → 49668 [ACK] Seq=1 Ack=640 Win=64896 Len=0 TSval=3127647586 TSecr=3127647586
6	0.245446566	127.0.0.1	127.0.0.1	TLSv1.3	1489	Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data, Application Data
7	0.245464573	127.0.0.1	127.0.0.1	TCP	68	49668 → 443 [ACK] Seq=640 Ack=1422 Win=64384 Len=0 TSval=3127647828 TSecr=3127647828
8	0.249919216	127.0.0.1	127.0.0.1	TLSv1.3	148	Change Cipher Spec, Application Data
9	0.249930657	127.0.0.1	127.0.0.1	TCP	68	443 → 49668 [ACK] Seq=1422 Ack=720 Win=65536 Len=0 TSval=3127647832 TSecr=3127647832
10	0.250311917	127.0.0.1	127.0.0.1	TLSv1.3	323	Application Data
11	0.298591573	127.0.0.1	127.0.0.1	TCP	68	49668 → 443 [ACK] Seq=720 Ack=1677 Win=65536 Len=0 TSval=3127647881 TSecr=3127647833
12	0.298620084	127.0.0.1	127.0.0.1	TLSv1.3	323	Application Data
13	0.298624480	127.0.0.1	127.0.0.1	TCP	68	49668 → 443 [ACK] Seq=720 Ack=1932 Win=65408 Len=0 TSval=3127647881 TSecr=3127647881
14	0.627256711	127.0.0.1	127.0.0.1	TLSv1.3	531	Application Data
15	0.670451803	127.0.0.1	127.0.0.1	TLSv1.3	241	Application Data
16	0.670473993	127.0.0.1	127.0.0.1	TCP	68	49668 → 443 [ACK] Seq=1183 Ack=2105 Win=65408 Len=0 TSval=3127648253 TSecr=3127648253
17	0.671318278	127.0.0.1	127.0.0.1	TLSv1.3	187	Application Data
18	0.671333229	127.0.0.1	127.0.0.1	TCP	68	49668 → 443 [ACK] Seq=1183 Ack=2224 Win=65408 Len=0 TSval=3127648254 TSecr=3127648254
19	0.671537376	127.0.0.1	127.0.0.1	TLSv1.3	92	Application Data
20	0.671552331	127.0.0.1	127.0.0.1	TCP	68	49668 → 443 [FIN, ACK] Seq=1207 Ack=2224 Win=65536 Len=0 TSval=3127648254 TSecr=3127648254
21	0.678364292	127.0.0.1	127.0.0.1	TLSv1.3	92	Application Data
22	0.678388164	127.0.0.1	127.0.0.1	TCP	56	49668 → 443 [RST] Seq=1208 Win=0 Len=0

Packet 22 Details:

- Frame 3: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface any, id 0
- Linux cooked capture v1
- Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
- Transmission Control Protocol, Src Port: 49668, Dst Port: 443, Seq: 1, Ack: 1, Len: 0
 - Source Port: 49668
 - Destination Port: 443
 - [Stream index: 0]
 - [Conversation completeness: Complete, WITH_DATA (63)]
 - [TCP Segment Len: 0]
 - Sequence Number: 1 (relative sequence number)
 - Sequence Number (raw): 1846364713
 - [Next Sequence Number: 1 (relative sequence number)]
 - Acknowledgment Number: 1 (relative ack number)
 - Acknowledgment number (raw): 3934297400
 - 1000 = Header Length: 32 bytes (8)
 - Flags: 0x010 (ACK)
 - Window: 512
 - [Calculated window size: 65536]
 - [Window size scaling factor: 128]

Packet 22 Hex Data:

```
0000 00 00 03 04 00 06 00 00 00 00 00 00 00 08 00 .....
0010 45 00 00 34 99 00 40 00 40 06 a3 c1 7f 00 00 01 E 4 @ @.....
0020 7f 00 00 01 c2 04 01 bb 6e 0d 4a 29 ea 80 9d 38 ..... n J) 8
0030 80 10 02 00 fe 28 00 00 01 01 08 0a ba 6c 1d 5e ..... I ^
0040 ba 6c 1d 5e | ^
```

Wireshark_anyBI3PD2.pcapng

Packets: 22 · Displayed: 22 (100.0%) · Dropped: 0 (0.0%)