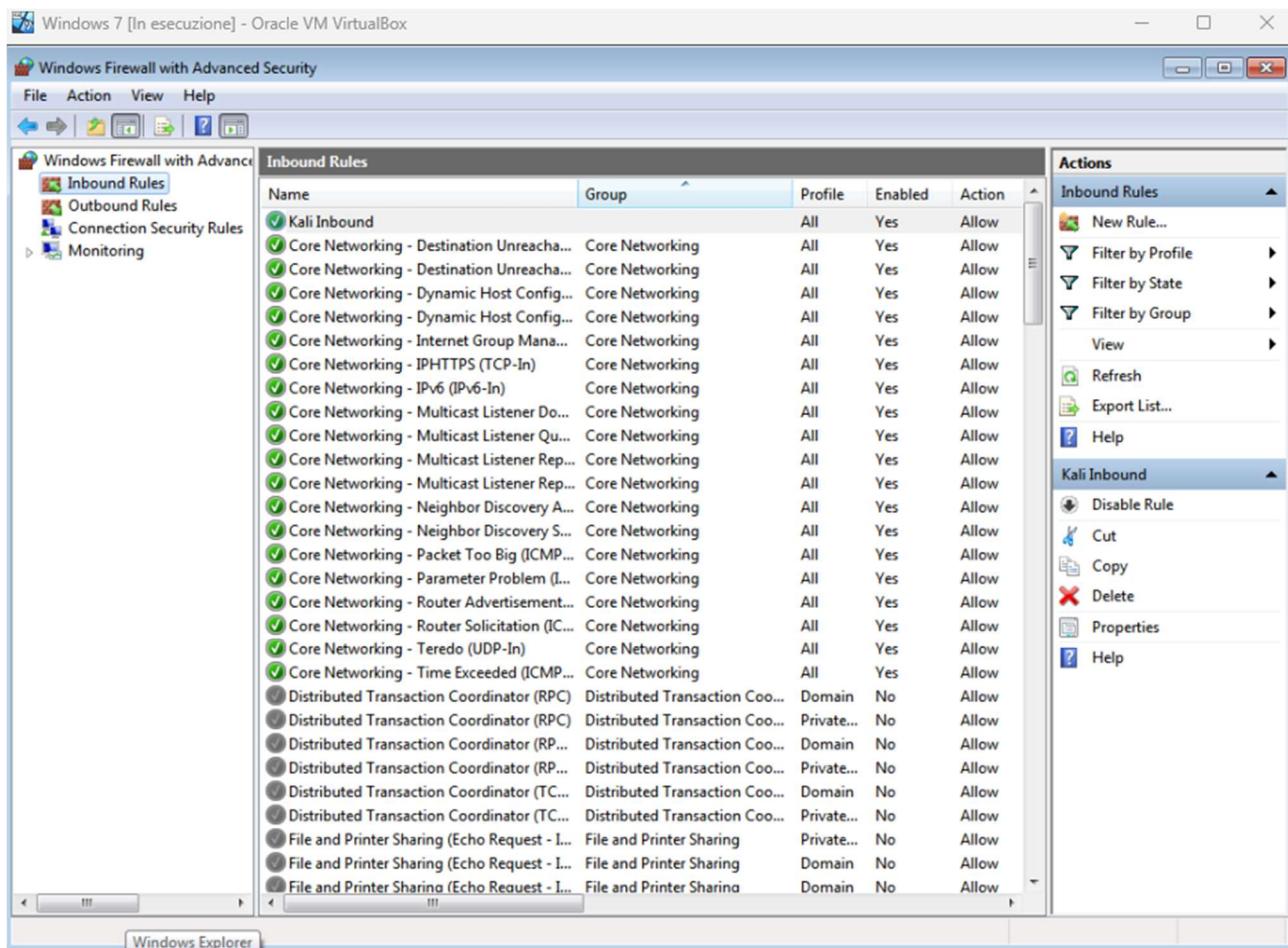


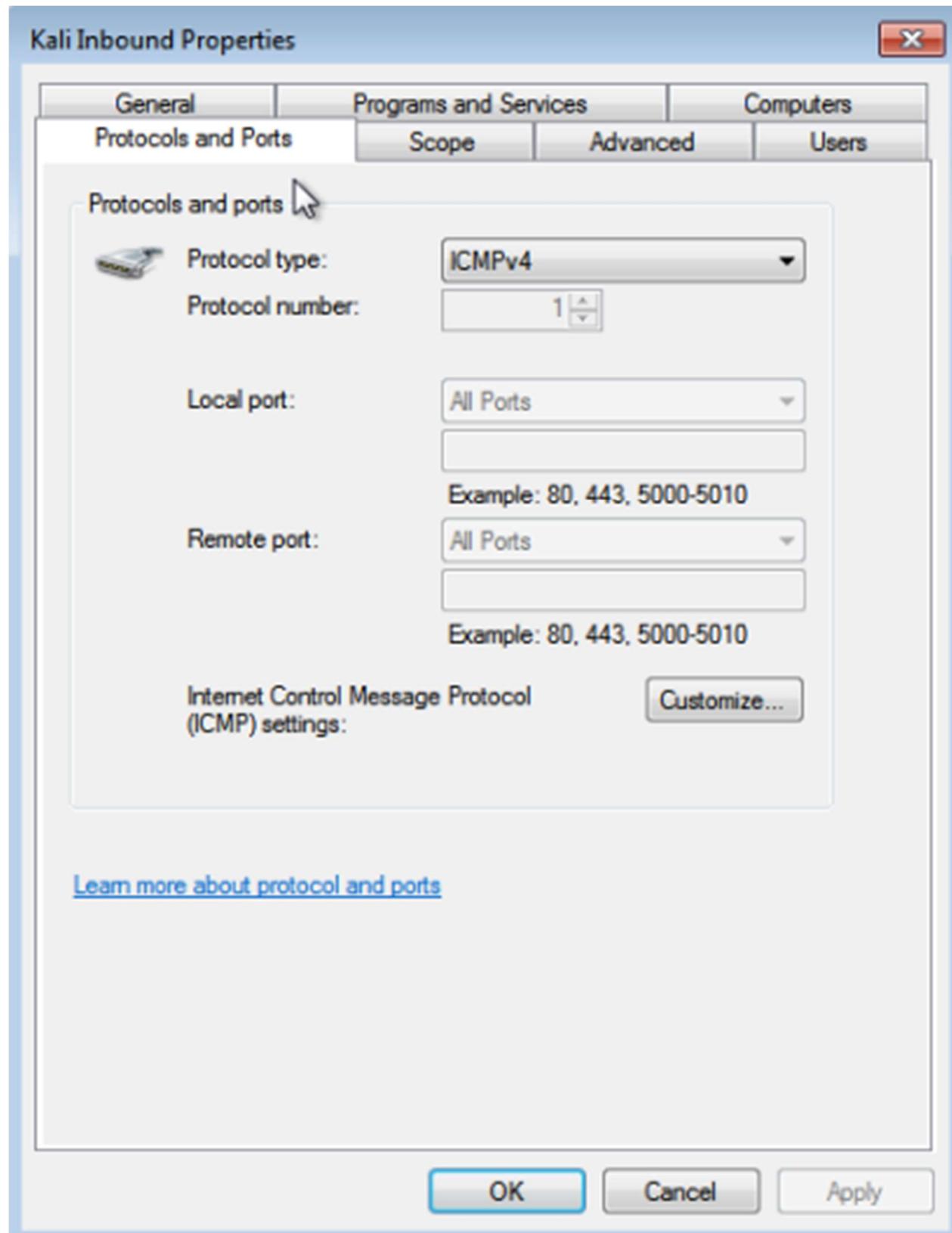
UTILIZZO DI FIREWALL WINDOWS E WIRESHARK

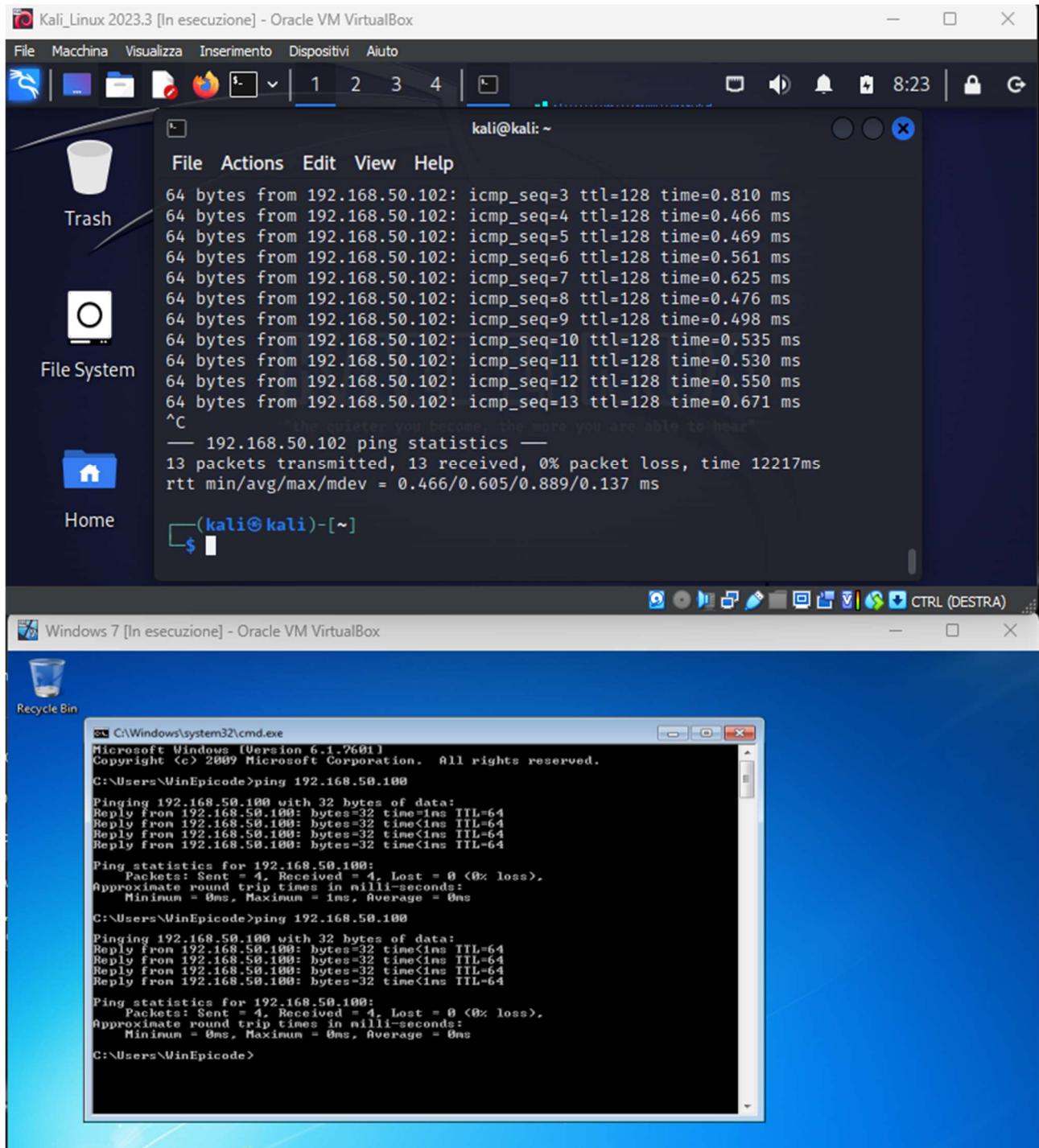
CONFIGURARE POLICY PER PERMETTERE IL PING DA FRA MACCHINE WINDOWS 7 E KALI LINUX.

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



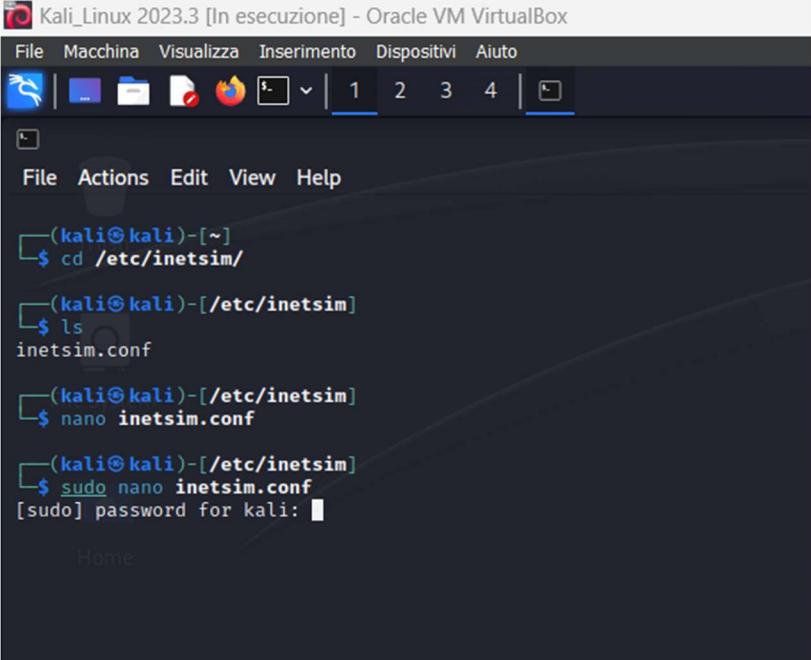
Ho creato una regola sia in ingresso che in uscita dalla macchina Windows7. Dopo di che sono entrato nelle Proprietà della regola e ho impostato il protocollo ICMPv4 (premetto che sulla pagina di creazione questa voce non era presente fra le opzioni e ho dovuto fare così). Successivamente ho lanciato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





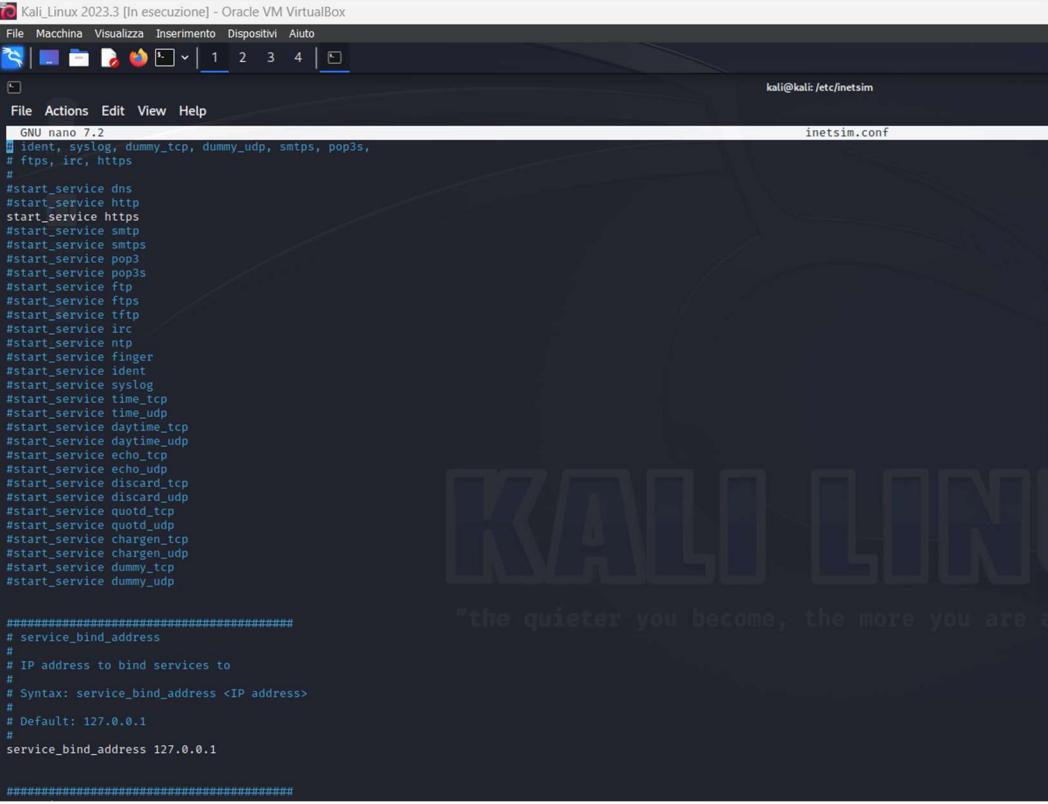
UTILIZZO DELLA UTILITY DELL'INETSIM PER L'EMULAZIONE DI SERVIZI INTERNET

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



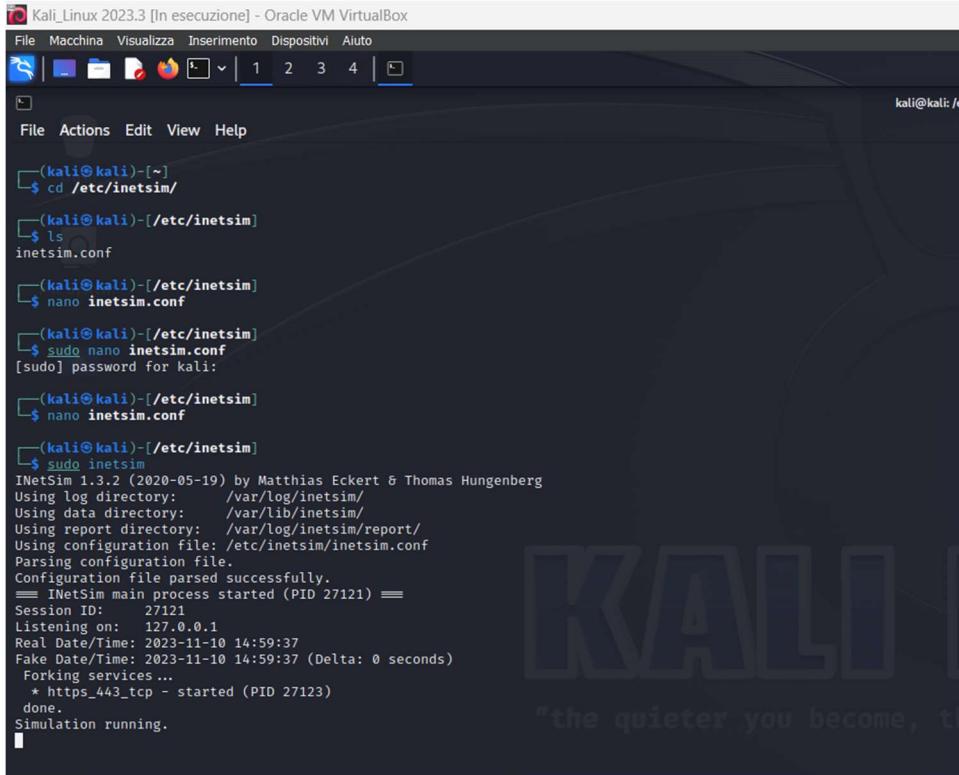
```
(kali㉿kali)-[~]
$ cd /etc/inetsim/
(kali㉿kali)-[/etc/inetsim]
$ ls
inetsim.conf
(kali㉿kali)-[/etc/inetsim]
$ nano inetsim.conf
(kali㉿kali)-[/etc/inetsim]
$ sudo nano inetsim.conf
[sudo] password for kali: ■
```

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



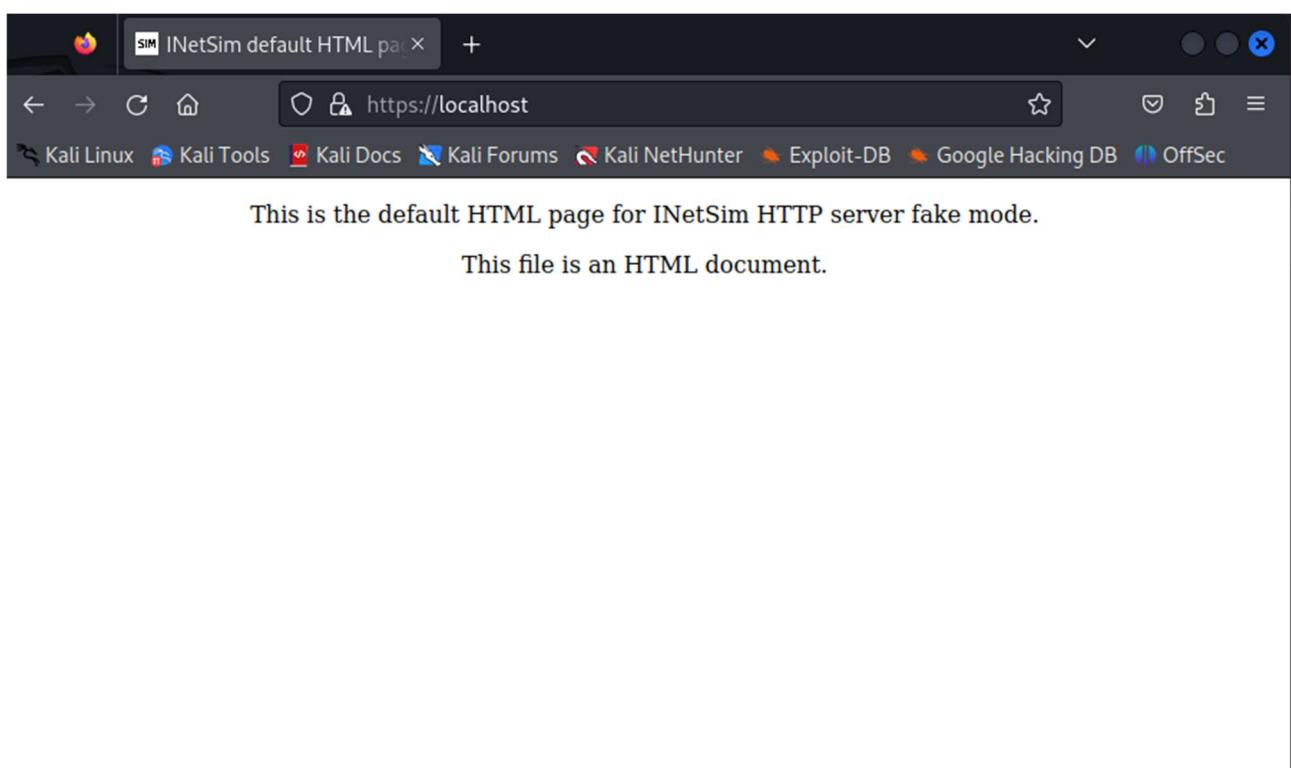
```
File Macchina Visualizza Inserimento Dispositivi Aiuto
File Actions Edit View Help
Home
File nano 7.2
inetsim.conf
GNU nano 7.2
# ident, syslog, dummy_tcp, dummy_udp, smtps, pop3s,
# ftps, irc, https
#
#start_service dns
#start_service http
start_service https
#start_service smtp
#start_service smtps
#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
#####
"the quieter you become, the more you are
```

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**.



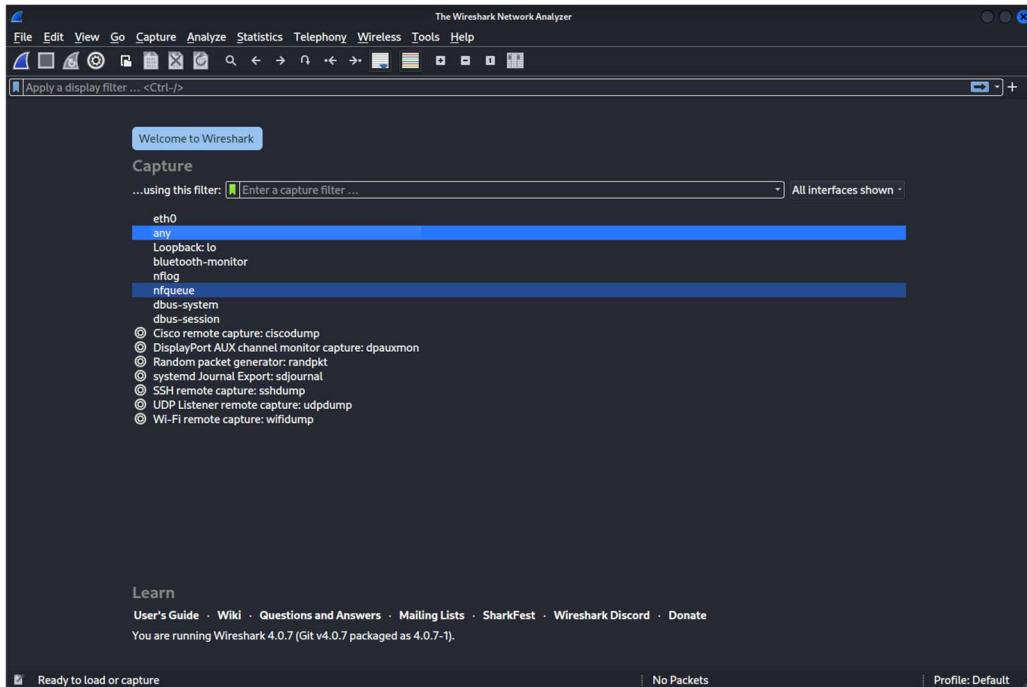
```
(kali㉿kali)-[~]
$ cd /etc/inetsim/
(kali㉿kali)-[/etc/inetsim]
$ ls
inetsim.conf
(kali㉿kali)-[/etc/inetsim]
$ nano inetsim.conf
(kali㉿kali)-[/etc/inetsim]
$ sudo nano inetsim.conf
[sudo] password for kali:
(kali㉿kali)-[/etc/inetsim]
$ nano inetsim.conf
(kali㉿kali)-[/etc/inetsim]
$ sudo inetsim
INetSim 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 sia effettivamente avviato, apro una pagina web puntando la ricerca su <https://localhost>.



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.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	76	60426 - 80 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=3126110339 TSecr=0 WS=128
2	0.000009311	127.0.0.1	127.0.0.1	TCP	76	80 - 60426 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM TSval=3126110339 TSecr=3126110339 WS=128
3	0.000019002	127.0.0.1	127.0.0.1	TCP	68	60426 - 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=3126110339 TSecr=3126110339
-	4.013837378	127.0.0.1	127.0.0.1	HTTP	499	GET / HTTP/1.1
5	0.013848548	127.0.0.1	127.0.0.1	TCP	68	80 - 60426 [ACK] Seq=432 Ack=432 Win=65152 Len=0 TSval=3126110353 TSecr=3126110353
6	0.059539614	127.0.0.1	127.0.0.1	TCP	218	80 - 60426 [PSH, ACK] Seq=1 Ack=432 Win=65536 Len=150 TSval=3126110399 TSecr=3126110353 [TCP segment of a reassembled PDU]
7	0.059566379	127.0.0.1	127.0.0.1	TCP	68	60426 - 80 [ACK] Seq=432 Ack=151 Win=65408 Len=0 TSval=3126110399 TSecr=3126110399
8	0.059588100	127.0.0.1	127.0.0.1	HTTP	326	HTTP/1.1 200 OK (text/html)
9	0.059592730	127.0.0.1	127.0.0.1	TCP	68	60426 - 80 [ACK] Seq=432 Ack=409 Win=65152 Len=0 TSval=3126110399 TSecr=3126110399
10	0.060456202	127.0.0.1	127.0.0.1	TCP	68	60426 - 80 [FIN, ACK] Seq=432 Ack=409 Win=65536 Len=0 TSval=3126110399 TSecr=3126110399
11	0.064766231	127.0.0.1	127.0.0.1	TCP	68	80 - 60426 [FIN, ACK] Seq=409 Ack=433 Win=65536 Len=0 TSval=3126110404 TSecr=3126110399
12	0.064793504	127.0.0.1	127.0.0.1	TCP	68	60426 - 80 [ACK] Seq=433 Ack=410 Win=65536 Len=0 TSval=3126110404 TSecr=3126110404
13	0.165089300	127.0.0.1	127.0.0.1	TCP	76	60434 - 80 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=3126110504 TSecr=0 WS=128
14	0.165102846	127.0.0.1	127.0.0.1	TCP	76	80 - 60434 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM TSval=3126110504 TSecr=3126110504 WS=128
15	0.165115757	127.0.0.1	127.0.0.1	TCP	68	60434 - 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=3126110504 TSecr=3126110504
16	0.165243304	127.0.0.1	127.0.0.1	HTTP	431	GET /favicon.ico HTTP/1.1
17	0.165249149	127.0.0.1	127.0.0.1	TCP	68	80 - 60434 [ACK] Seq=1 Ack=364 Win=65152 Len=0 TSval=3126110504 TSecr=3126110504
18	0.221649709	127.0.0.1	127.0.0.1	TCP	221	80 - 60434 [PSH, ACK] Seq=1 Ack=364 Win=65536 Len=150 TSval=3126110561 TSecr=3126110504 [TCP segment of a reassembled PDU]
19	0.221676286	127.0.0.1	127.0.0.1	TCP	68	60434 - 80 [ACK] Seq=364 Ack=154 Win=65408 Len=0 TSval=3126110561 TSecr=3126110561
20	0.221697249	127.0.0.1	127.0.0.1	HTTP	266	HTTP/1.1 200 OK (image/x-icon)
21	0.221702154	127.0.0.1	127.0.0.1	TCP	68	60434 - 80 [ACK] Seq=364 Ack=352 Win=65280 Len=0 TSval=3126110561 TSecr=3126110561
22	0.221949256	127.0.0.1	127.0.0.1	TCP	68	60434 - 80 [FIN, ACK] Seq=364 Ack=352 Win=65536 Len=0 TSval=3126110561 TSecr=3126110561
23	0.241515812	127.0.0.1	127.0.0.1	TCP	68	80 - 60434 [FIN, ACK] Seq=352 Ack=365 Win=65536 Len=0 TSval=3126110580 TSecr=3126110581
24	0.241545850	127.0.0.1	127.0.0.1	TCP	68	60434 - 80 [ACK] Seq=365 Ack=353 Win=65536 Len=0 TSval=3126110581 TSecr=3126110580

Below the table, there is a detailed analysis of the selected GET request:

```

HyperText Transfer Protocol
GET / HTTP/1.1\r\n
Host: localhost\r\n
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\n
Accept-Language: en-US,en;q=0.5\r\n
Accept-Encoding: gzip, deflate, br\r\n
Connection: keep-alive\r\n
Upgrade-Insecure-Requests: 1\r\n
Sec-Fetch-Dest: document\r\n
Sec-Fetch-Mode: navigate\r\n
Sec-Fetch-Site: none\r\n
Sec-Fetch-User: ?1\r\n
\r\n
[Full request URI: http://localhost/]
[HTTP request 1/1]
[Response in frame: 8]

```

The status bar at the bottom left says "The full requested URL (including host name) (http.request.full_uri)" and the bottom right says "Packets: 24 - Displayed: 24 (100.0%) - Dropped: 0 (0.0%)".

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>

*any

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

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 TStamp=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 TStamp=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 TStamp=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 TStamp=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 TStamp=3126431305 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 TStamp=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 TStamp=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 TStamp=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 TStamp=3126431357 TSecr=3126431357
12	0.067342499	127.0.0.1	127.0.0.1	TCP	68	41582 → 80 [ACK] Seq=443 Ack=249 Win=65536 Len=0 TStamp=3126431357 TSecr=3126431357

```

> 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\r\n
    Host: localhost\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate, br\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    Sec-Fetch-Dest: document\r\n
    Sec-Fetch-Mode: navigate\r\n
    Sec-Fetch-Site: none\r\n
    Sec-Fetch-User: ?1\r\n
    \r\n
  [Full request URI: http://localhost/sample.txt]

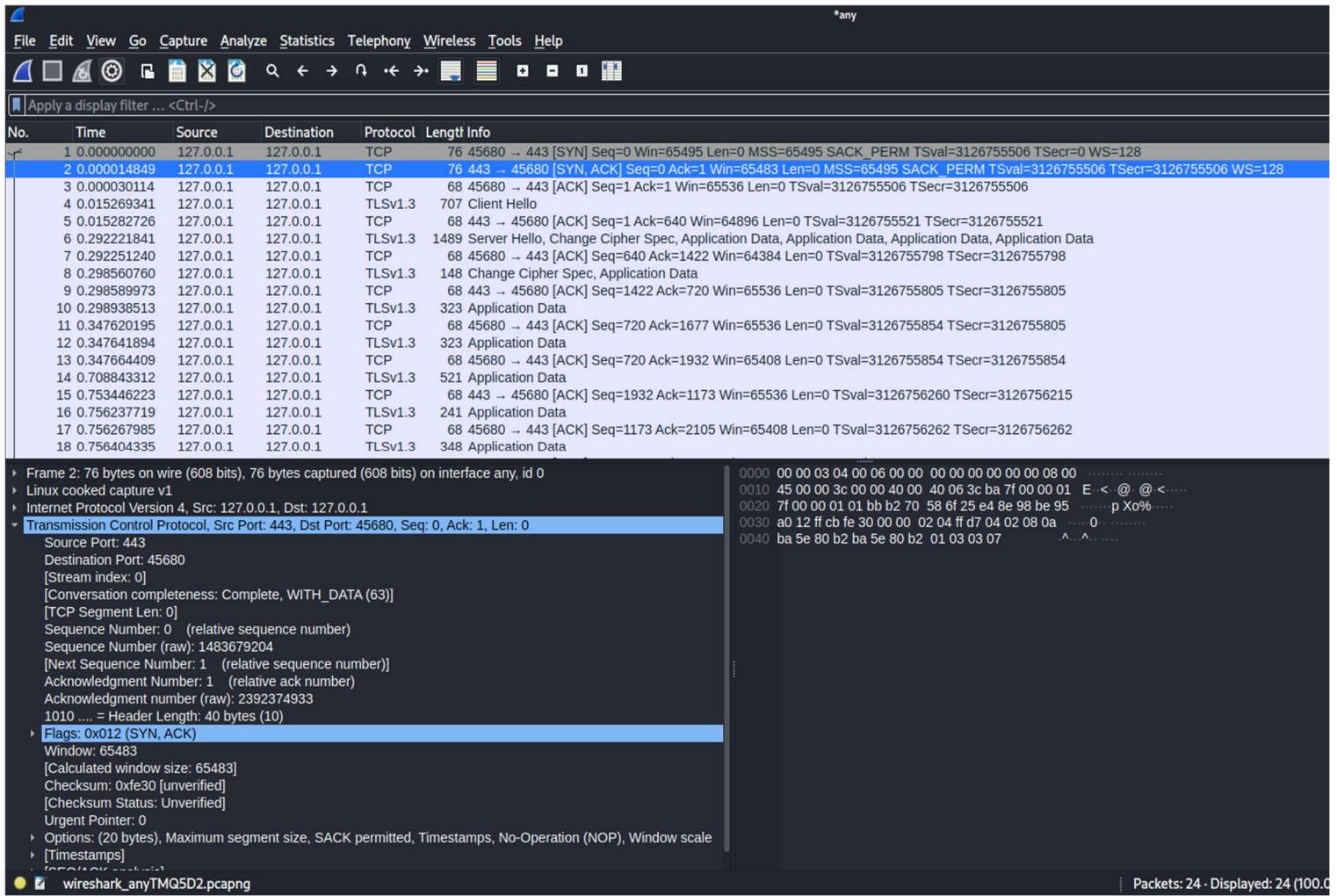
```

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>



The screenshot shows a Wireshark capture of network traffic. The packet list pane shows 24 captured packets, mostly TCP and TLSv1.3 frames. The details pane shows the structure of a selected packet, which is a Transmission Control Protocol (TCP) frame with flags set to SYN and ACK. The bytes pane shows the raw hex and ASCII data of the selected packet. The status bar at the bottom right indicates "Packets: 24 - Displayed: 24 (100.0%)".

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>

