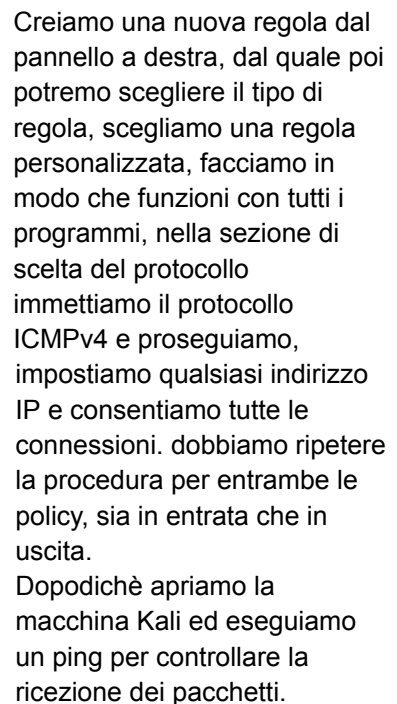
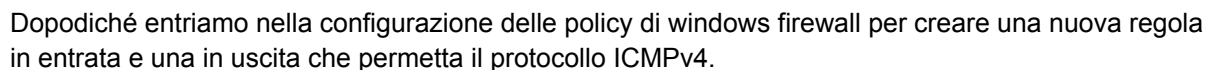
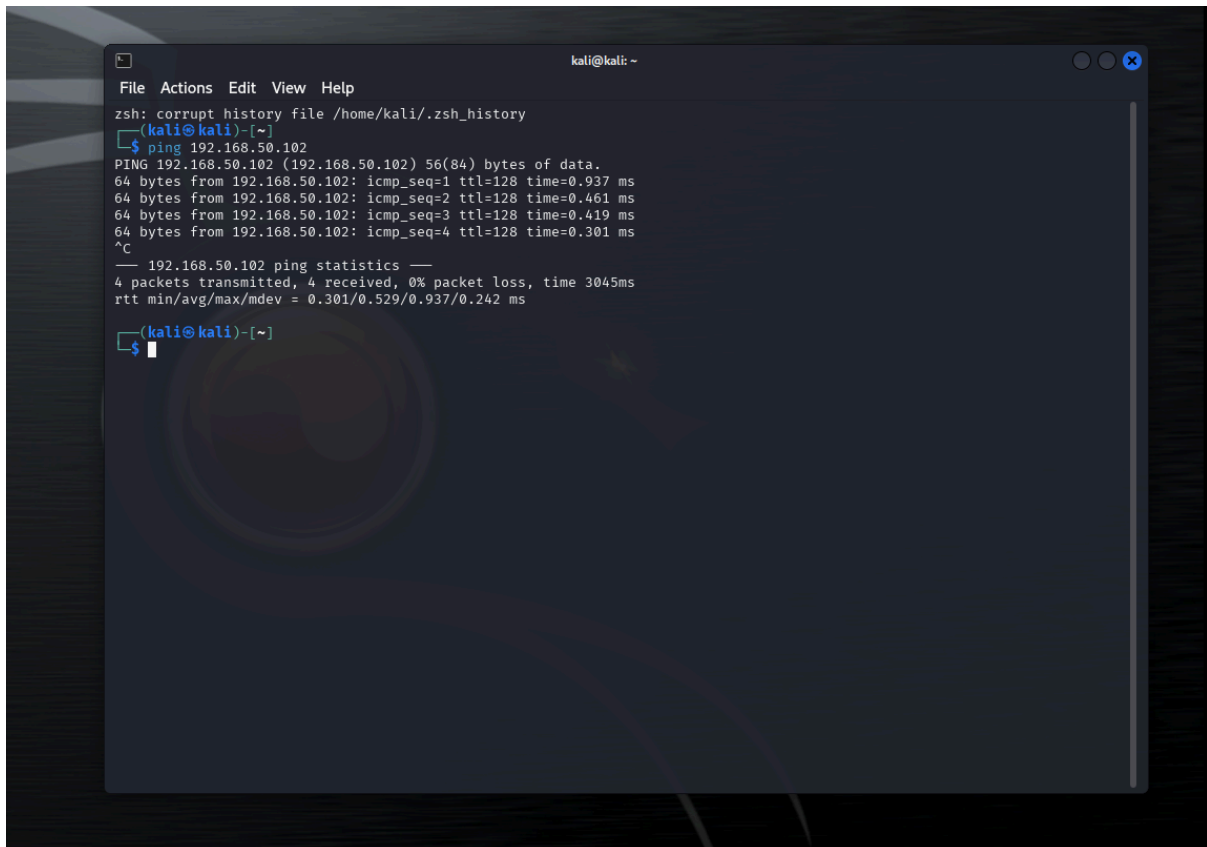


- **Configurazione Policy su Windows per permettere il ping da Kali Linux**

Per permettere la comunicazione tra Kali Linux e Windows 10, dobbiamo impostare il windows firewall in avvio manuale come in figura:



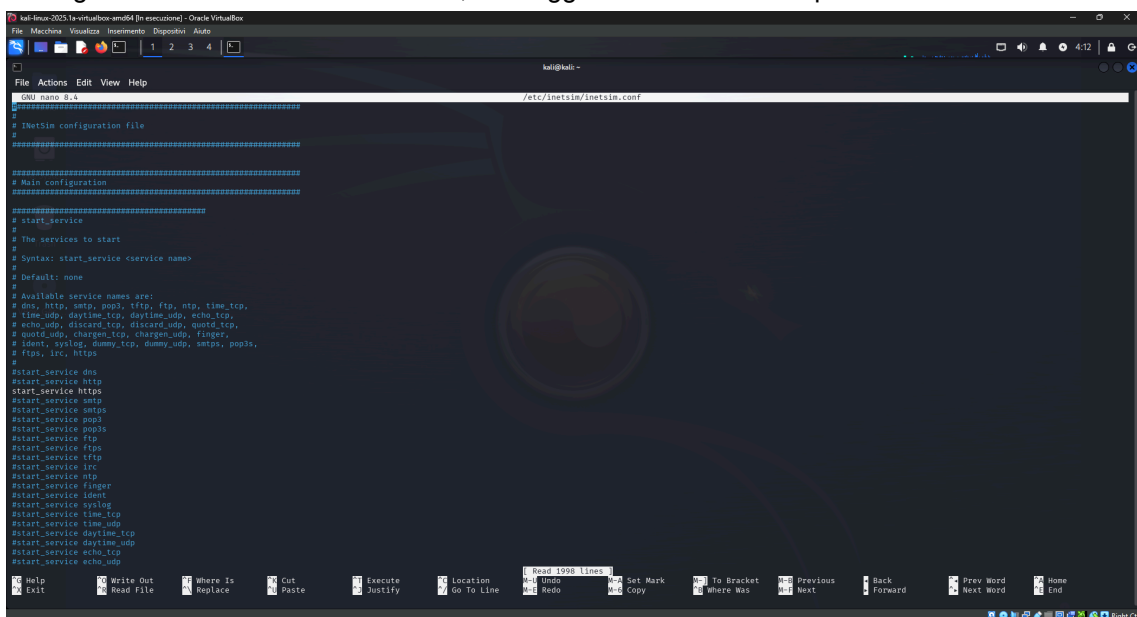


```
kali@kali: ~  
File Actions Edit View Help  
zsh: corrupt history file /home/kali/.zsh_history  
(kali@kali)~  
$ ping 192.168.50.102  
PING 192.168.50.102 (192.168.50.102) 56(84) bytes of data:  
64 bytes from 192.168.50.102: icmp_seq=1 ttl=128 time=0.937 ms  
64 bytes from 192.168.50.102: icmp_seq=2 ttl=128 time=0.461 ms  
64 bytes from 192.168.50.102: icmp_seq=3 ttl=128 time=0.419 ms  
64 bytes from 192.168.50.102: icmp_seq=4 ttl=128 time=0.301 ms  
^C  
--- 192.168.50.102 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3045ms  
rtt min/avg/max/mdev = 0.301/0.529/0.937/0.242 ms  
(kali@kali)~  
$
```

Come possiamo notare, tutti i pacchetti sono stati ricevuti ed ora la connessione tra le macchine e' attiva.

## - Utilizzo dell'utility InetSim per l'emulazione di servizi Internet

Apriamo una console e digitiamo "sudo nano etc/inetsim/inetsim.conf" e dalle impostazioni di inetsim disattiviamo tutti i servizi all'infuori di HTTPS. In questo modo avremo solo i servizi indicati attivati, per disattivare gli altri basta aggiungere # davanti alla riga per farla diventare un "commento" di conseguenza una volta avviata inetsim, non leggerà tutto cio' che e' preceduto da #.

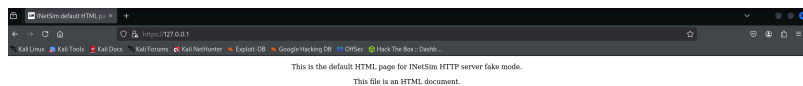


```
GNU nano 2.9.4 /etc/inetsim/inetsim.conf  
#####  
# InetSim configuration file  
#####  
#####  
# Main configuration  
#####  
#####  
# start_service  
#  
# The services to start  
# Syntax: start_service <service name>  
# Default: none  
# Available service names are:  
# dns, http, smtp, pop3, ftp, ntp, time_tcp,  
# time_udp, daytime_tcp, daytime_udp, echo_tcp,  
# echo_udp, discard_tcp, discard_udp, quid_tcp,  
# quid_udp, chargen_tcp, chargen_udp, finger,  
# ident, syslog, dummy_tcp, dummy_udp, setps, 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 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
```

Una volta completato, salviamo e chiudiamo la configurazione.  
Facciamo poi partire inetsim dalla console.

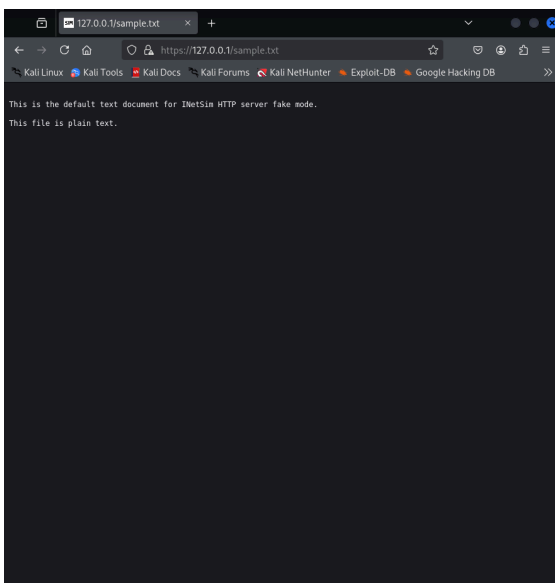
```
kali@kali:~$ nano /etc/inetsim/inetsim.conf
zsh: corrupt history file /home/kali/.zsh_history
--(kali@kali)~$
--(kali@kali)~$ sudo nano /etc/inetsim/inetsim.conf
[sudo] password for kali:
[sudo] password for kali:
--(kali@kali)~$
--(kali@kali)~$ sudo nano /etc/inetsim/inetsim.conf
--(kali@kali)~$
--(kali@kali)~$ sudo nano /etc/inetsim/inetsim.conf
--(kali@kali)~$
--(kali@kali)~$ sudo inetsim
Inetsim 1.0.2 (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 19983) ==
Session ID: 19983
Listening on: 127.0.0.1
Fake Date/Time: 2025-07-12 04:15:42
Fake Date/Time: 2025-07-12 04:15:42 (Delta: 0 seconds)
Parking services...
* https_443_tcp - started (PID 19985)
done.
Simulation running.
```

Controlliamo ora tramite browser se possiamo connetterci alla porta 443 del localhost.



## - Cattura di pacchetti con Wireshark

Avviamo ora Wireshark in ascolto sull'interfaccia di loopback e connettiamoci al localhost, richiedendo una delle risorse disponibili come "sample.txt"



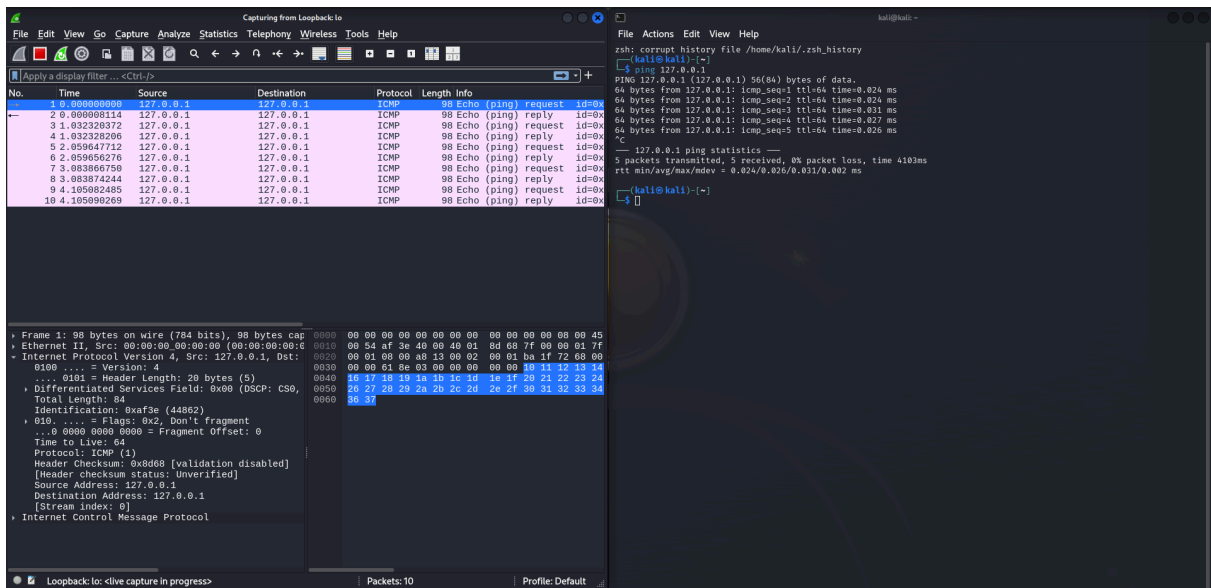
Possiamo notare come ci abbia restituito un file di testo fittizio che era già presente nelle configurazioni della InteSim.

Ora vediamo cosa wireshark ha catturato durante la nostra richiesta del file "sample.txt"



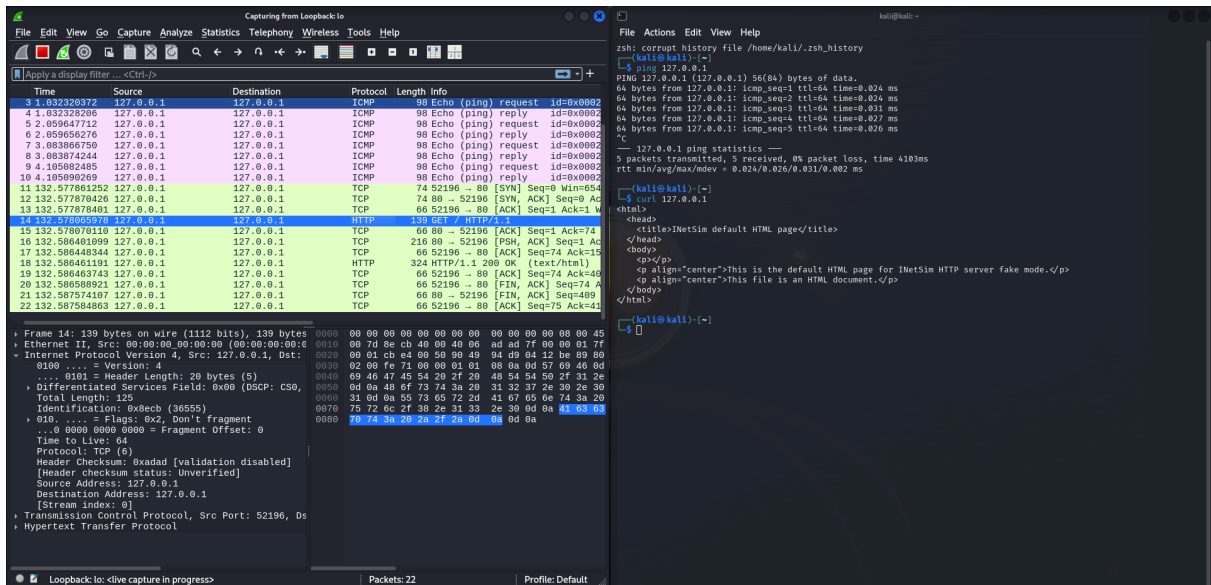
## - Procedere con lo sniffing delle comunicazioni

Apriamo ora wireshark per iniziare a sniffare le comunicazioni e mettiamolo in ascolto di loopback come prima.



Iniziamo utilizzando un ping per vedere come vengono fatte le richieste di comunicazione con il protocollo ICMP.

Ora facciamo un “curl 127.0.0.1” per vedere come lavorano le richieste HTTP.



## - Analizzare il contenuto dei pacchetti

Analizziamo ora il contenuto dei pacchetti, partendo dal ping svolto all'inizio per verificare la connessione.

```

> Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface lo, id 0
> Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0xaf3e (44862)
  > 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: ICMP (1)
    Header Checksum: 0x8d68 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 127.0.0.1
    Destination Address: 127.0.0.1
    [Stream index: 0]
> Internet Control Message Protocol
  Type: 8 (Echo (ping) request)
  Code: 0
  Checksum: 0xa813 [correct]
  [Checksum Status: Good]
  Identifier (BE): 2 (0x0002)
  Identifier (LE): 512 (0x0200)
  Sequence Number (BE): 1 (0x0001)
  Sequence Number (LE): 256 (0x0100)
  [Response frame: 2]
  Timestamp from icmp data: Jul 12, 2025 04:41:30.233057000 EDT
  [Timestamp from icmp data (relative): 0.000016132 seconds]
> Data (40 bytes)
  Data: 101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f3031323334353637
  [Length: 40]
```

Possiamo notare in questa sezione come lavora il ping in richiesta e possiamo vedere anche i dati che sono stati inviati per fare la richiesta.

Vediamo anche che tiene nota del source address e del destination address che in questo caso sono gli stessi.

Andiamo a vedere ora un comando “curl” come lavora e come vengono sniffati i pacchetti da wireshark.

Possiamo vedere comunque come vengono rappresentati i vari livelli ISO/OSI nella cattura dei pacchetti.

```
No.    Time           Source                Destination            Protocol Length Info
14    0.000000  127.0.0.1            127.0.0.1              Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
15    0.000000  127.0.0.1            127.0.0.1              Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
16    0.000000  127.0.0.1            127.0.0.1              Transmission Control Protocol, Src Port: 52196, Dst Port: 80, Seq: 1, Ack: 1, Len: 73
17    0.000000  127.0.0.1            127.0.0.1              Hypertext Transfer Protocol
18    0.000000  127.0.0.1            127.0.0.1              GET / HTTP/1.1\r\n
19    0.000000  127.0.0.1            127.0.0.1              HTTP/1.1 200 OK
20    0.000000  127.0.0.1            127.0.0.1              Content-Length: 258
21    0.000000  127.0.0.1            127.0.0.1              Server: INetSim HTTP Server
22    0.000000  127.0.0.1            127.0.0.1              Connection: Close
23    0.000000  127.0.0.1            127.0.0.1              Content-Type: text/html
24    0.000000  127.0.0.1            127.0.0.1              Date: Sat, 12 Jul 2025 08:43:42 GMT
25    0.000000  127.0.0.1            127.0.0.1              <html>
26    0.000000  127.0.0.1            127.0.0.1              <head>
27    0.000000  127.0.0.1            127.0.0.1              <title>INetSim default HTML page</title>
28    0.000000  127.0.0.1            127.0.0.1              </head>
29    0.000000  127.0.0.1            127.0.0.1              <body>
30    0.000000  127.0.0.1            127.0.0.1              <p></p>
31    0.000000  127.0.0.1            127.0.0.1              <p align="center">This is the default HTML page for INetSim HTTP server fake mode.</p>
32    0.000000  127.0.0.1            127.0.0.1              <p align="center">This file is an HTML document.</p>
33    0.000000  127.0.0.1            127.0.0.1              </body>
34    0.000000  127.0.0.1            127.0.0.1              </html>
```

Possiamo notare come wireshark tiene nota dei comandi utilizzati per inviare il comando “curl”.

Vediamo come e' stata fatta la richiesta e anche che e' stata accettata.

Se noi utilizziamo il comando follow sul nostro pacchetto HTTP possiamo vedere esattamente come e' stata inviata la richiesta di “curl”

```
Wireshark - Follow HTTP Stream (tcp.stream eq 0) - Loopback: lo

GET / HTTP/1.1
Host: 127.0.0.1
User-Agent: curl/8.13.0
Accept: */*

HTTP/1.1 200 OK
Content-Length: 258
Server: INetSim HTTP Server
Connection: Close
Content-Type: text/html
Date: Sat, 12 Jul 2025 08:43:42 GMT

<html>
<head>
<title>INetSim default HTML page</title>
</head>
<body>
<p></p>
<p align="center">This is the default HTML page for INetSim HTTP server fake mode.</p>
<p align="center">This file is an HTML document.</p>
</body>
</html>
```

Packet 14. 1 client pkt, 1 server pkt, 1 turn. Click to select.

Entire conversation (481 bytes) Show as ASCII No delta times Stream 0

Find: Case sensitive Find Next

Filter Out This Stream Print Save as... Back Close Help