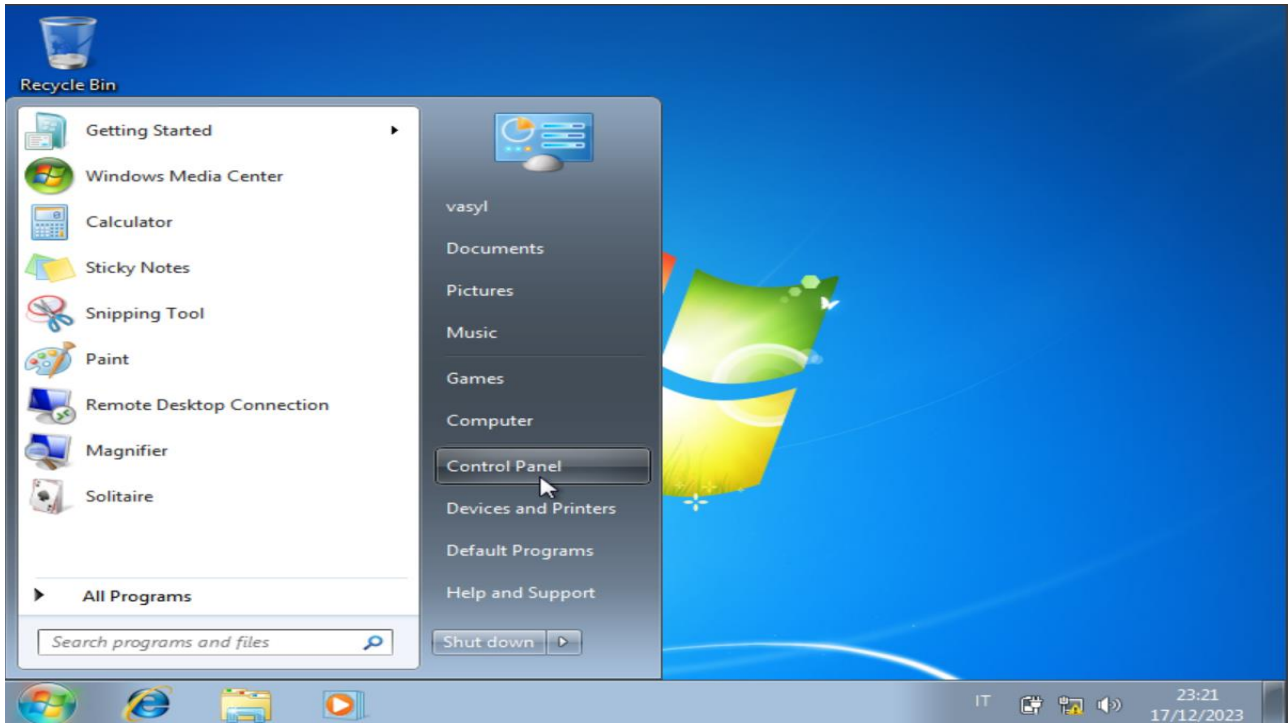


## Configurazione delle policy del FIREWALL di W7

Prima di tutto impostiamo il Firewall di windows 7 a finche riesca a fare ping con Kali e viceversa seguendo questi passaggi:

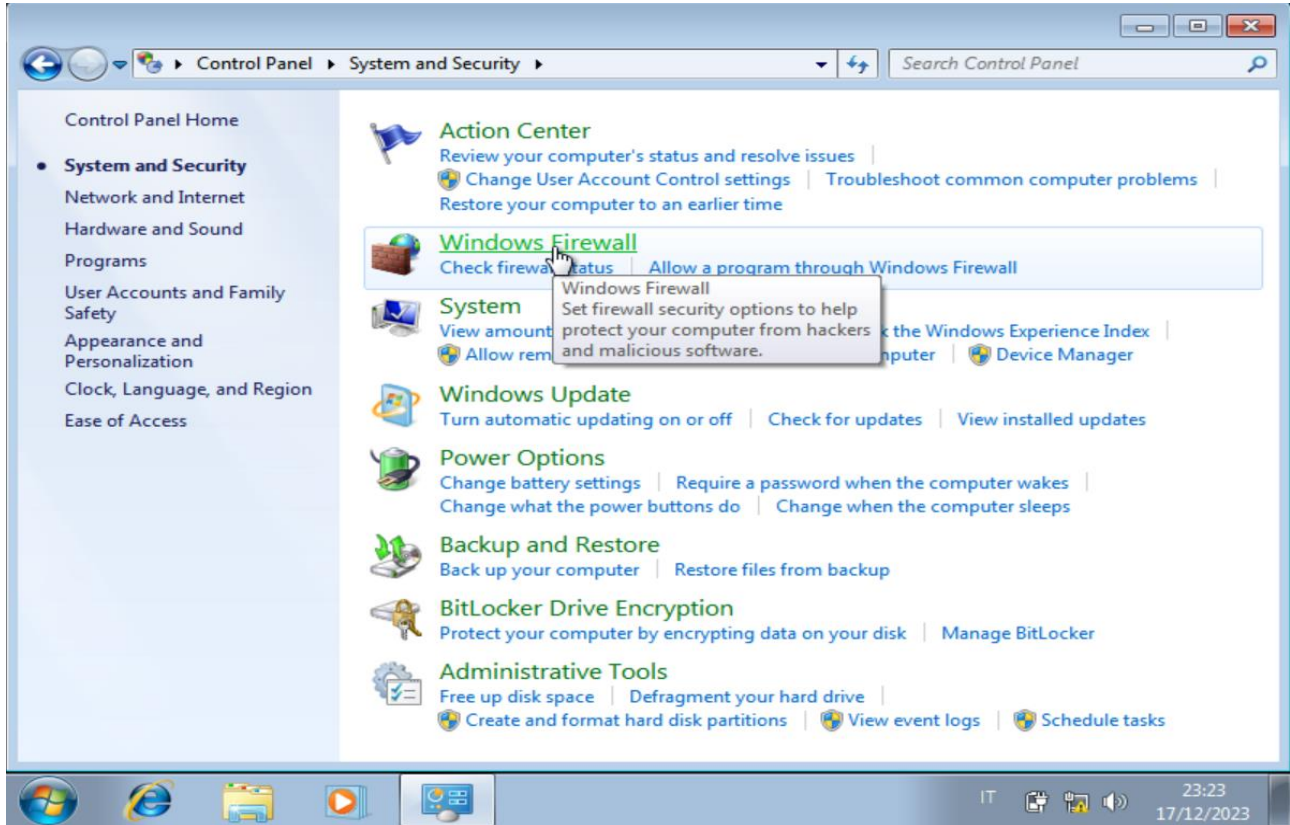
1



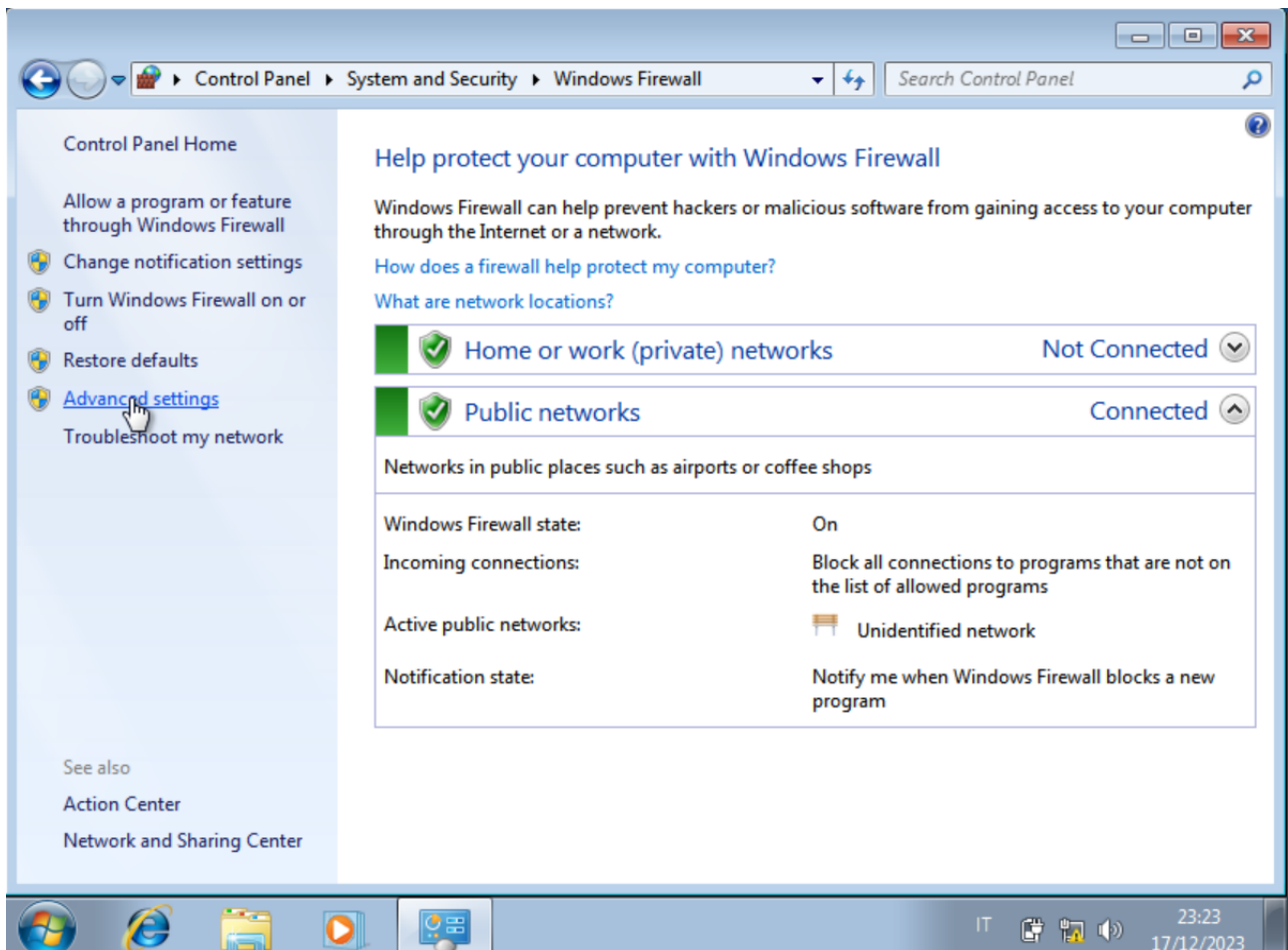
2



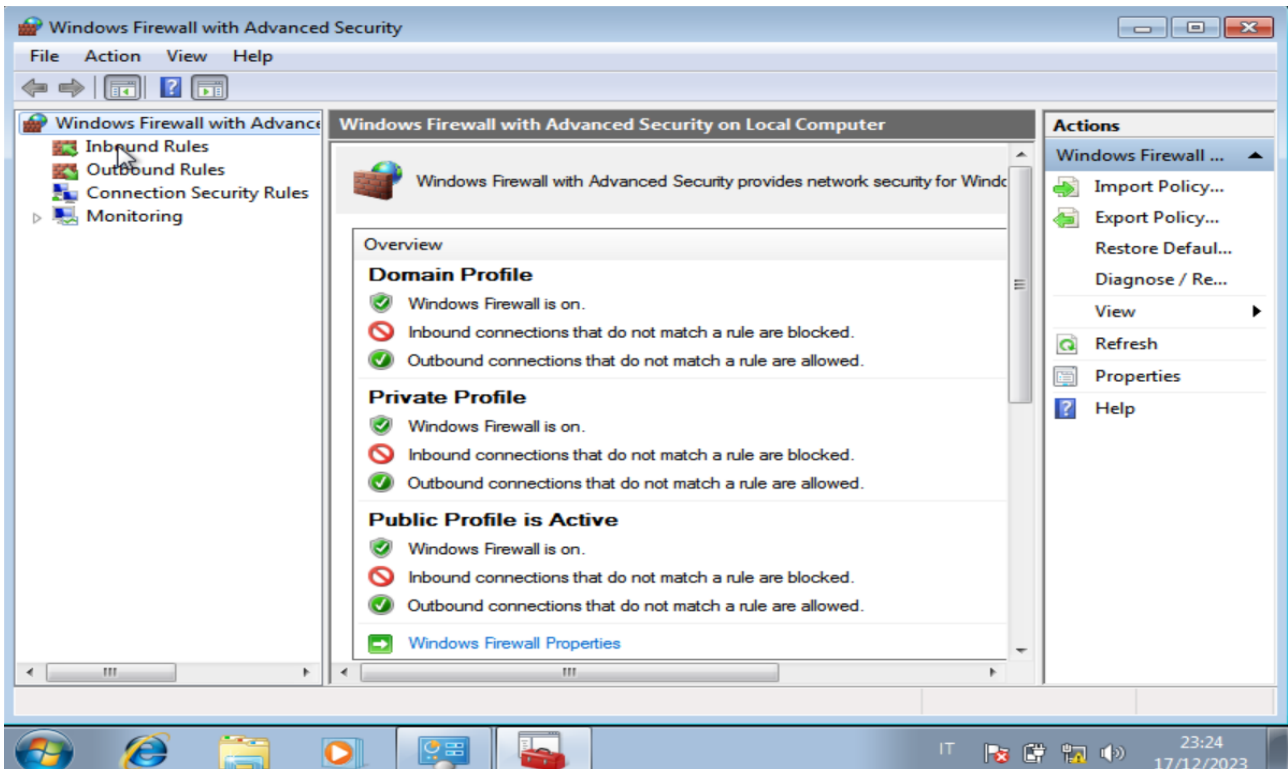
3



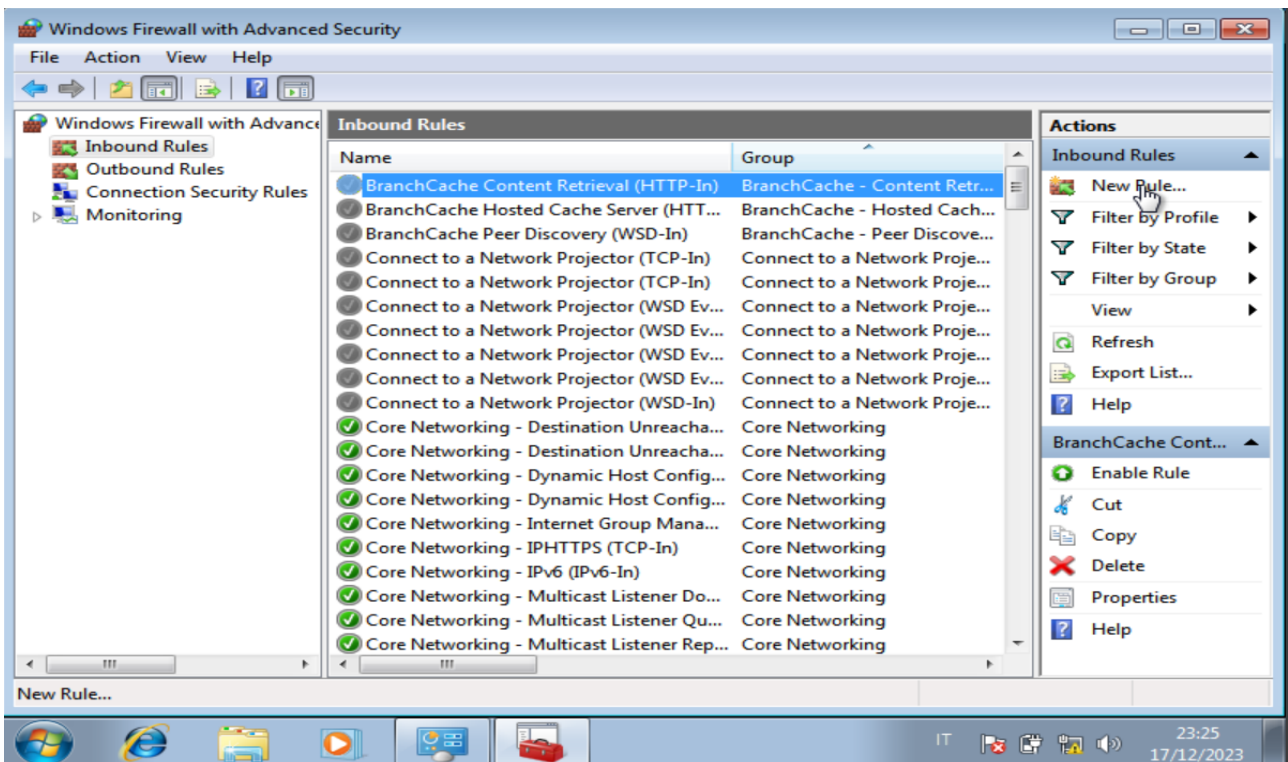
4



5



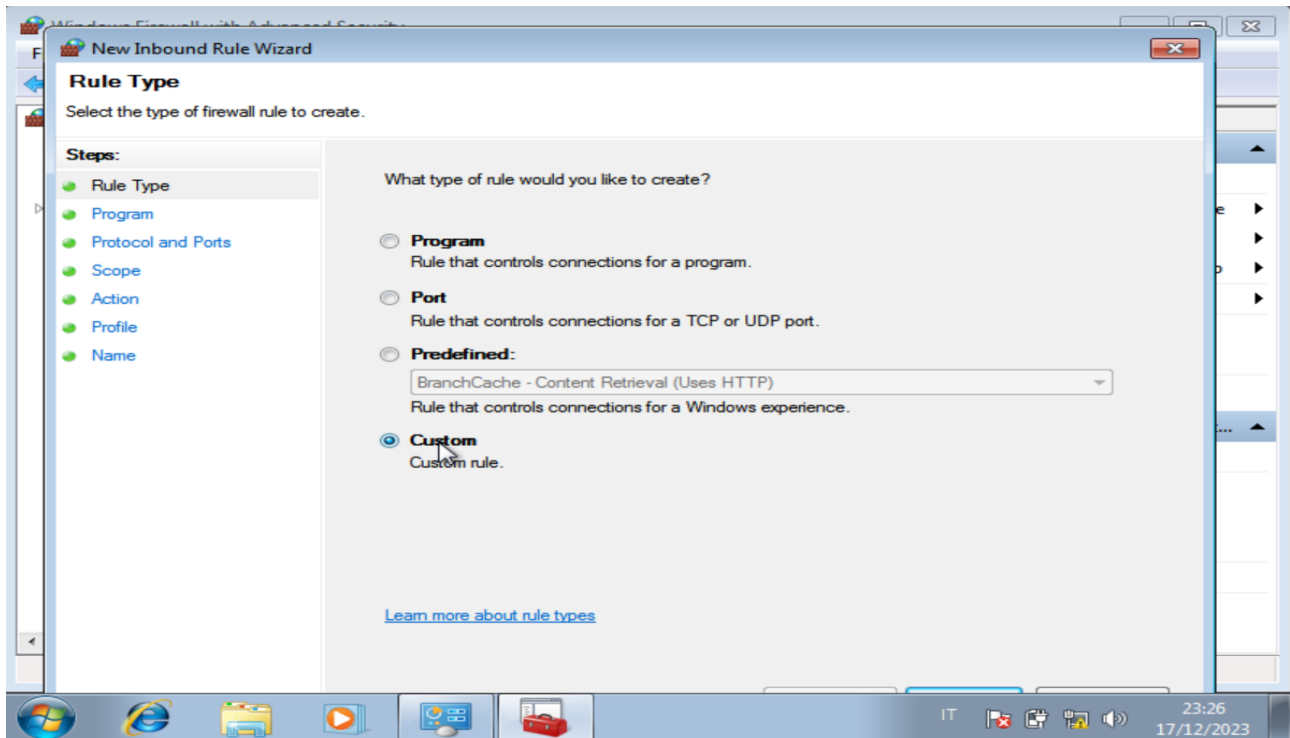
6



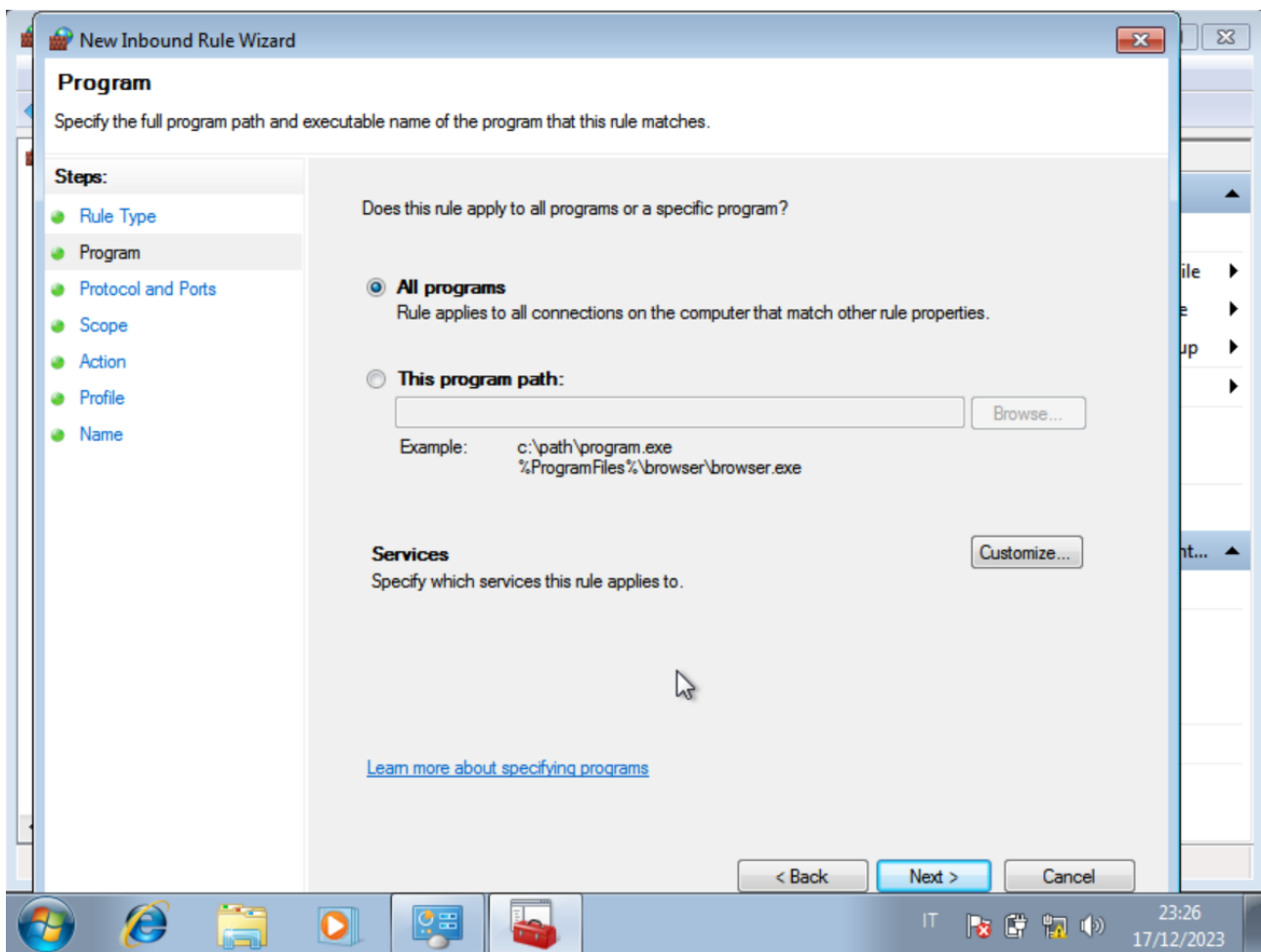
A questo punto settiamo una nuova regola d'ingaggio a finché il ping (scambio pacchetti fittizi)

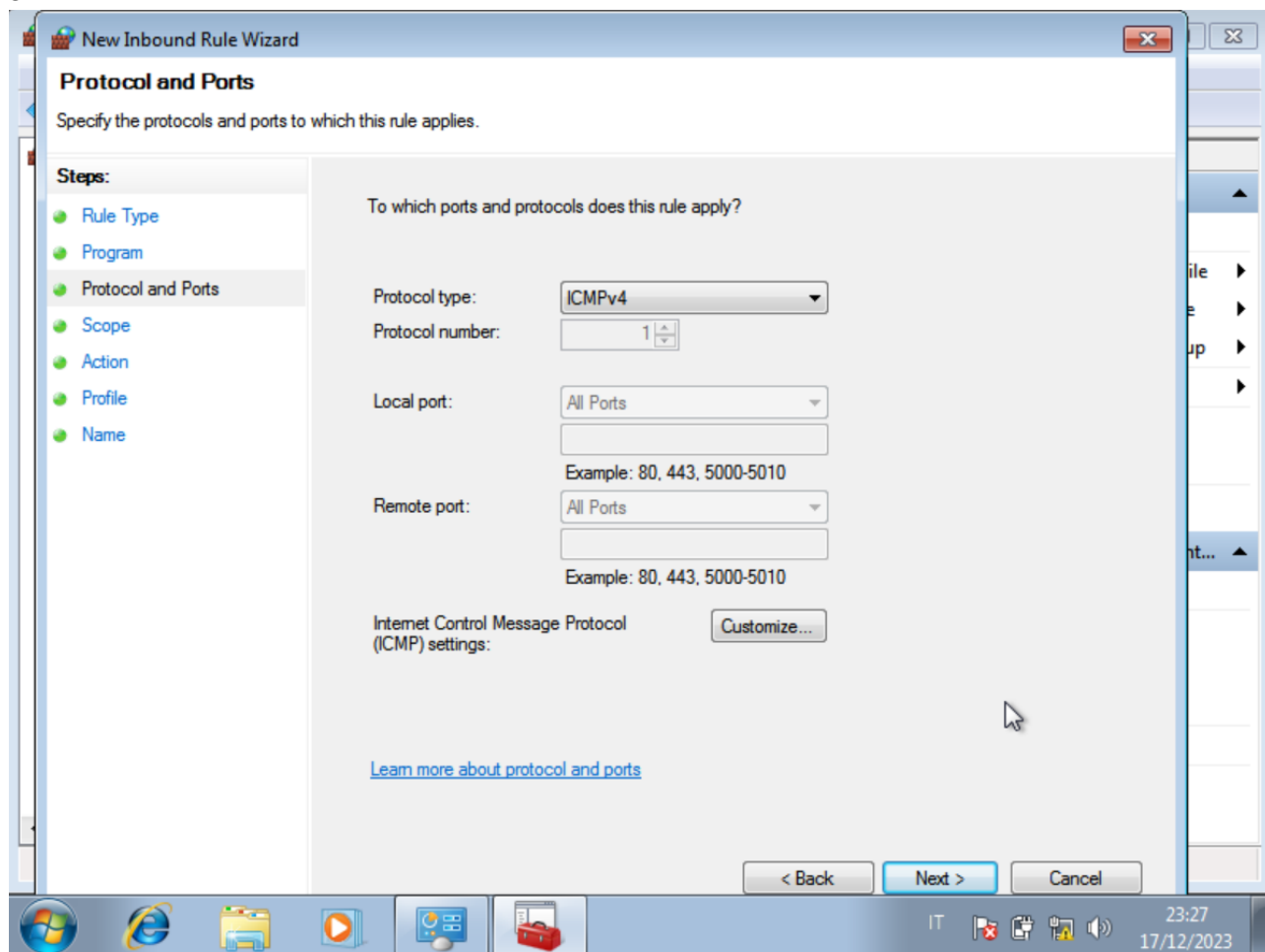
Possa avvenire senza essere bloccato dal FIREWALL.

1



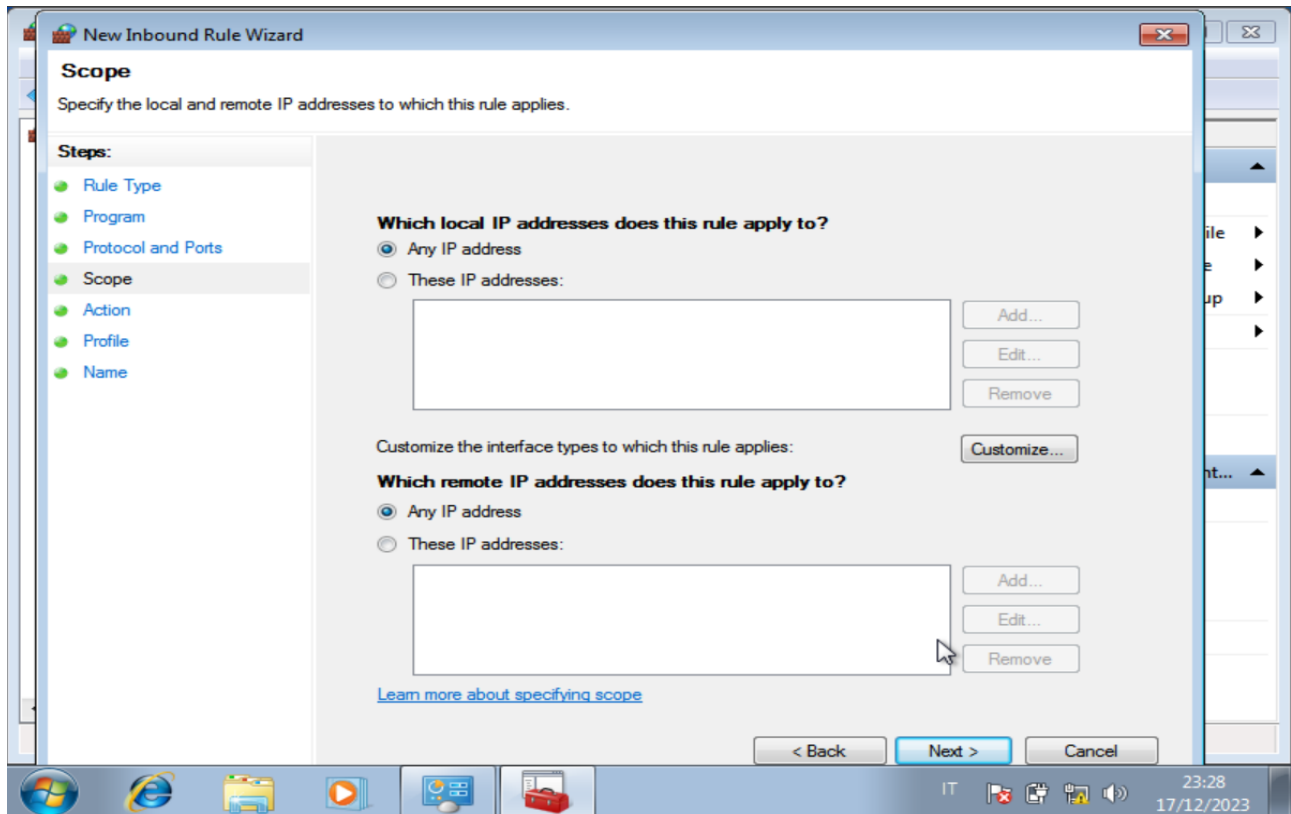
2



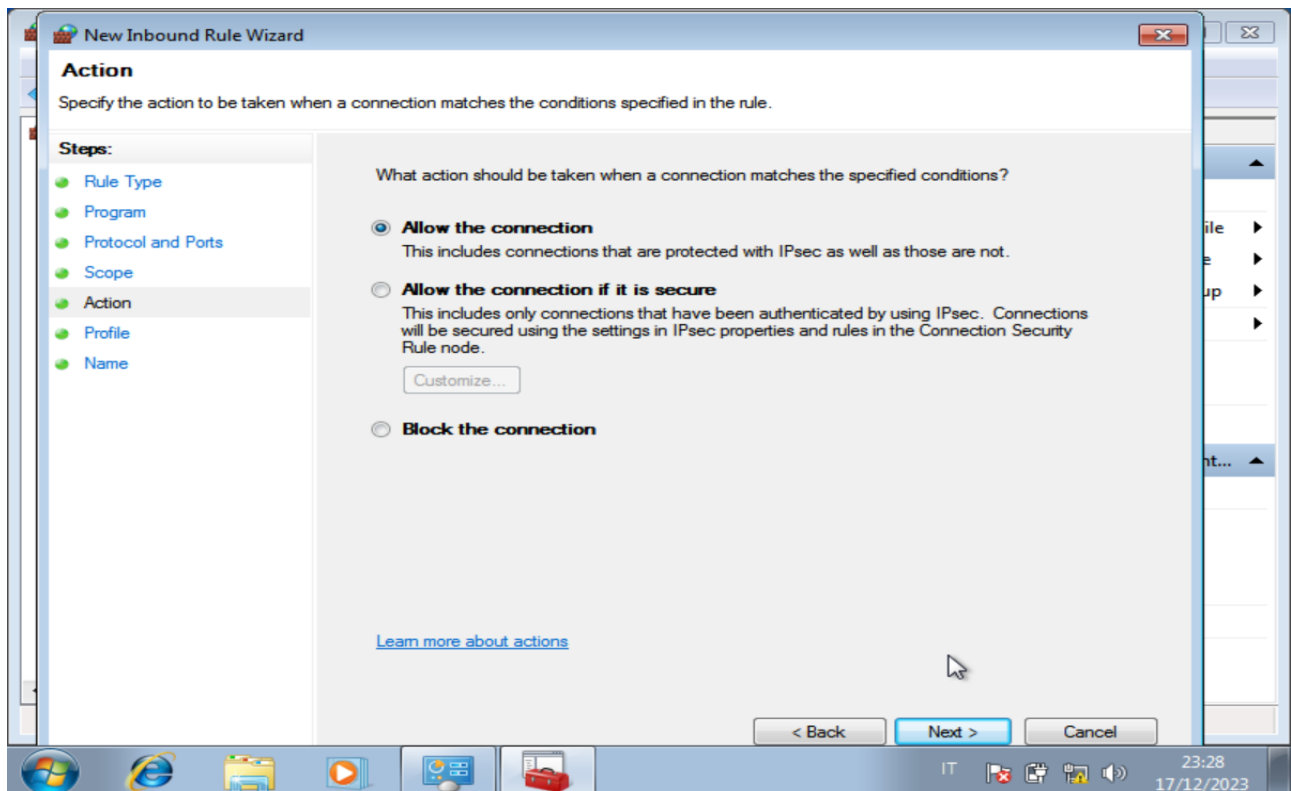


Impostare il tipo di protocollo ICMPv4

4

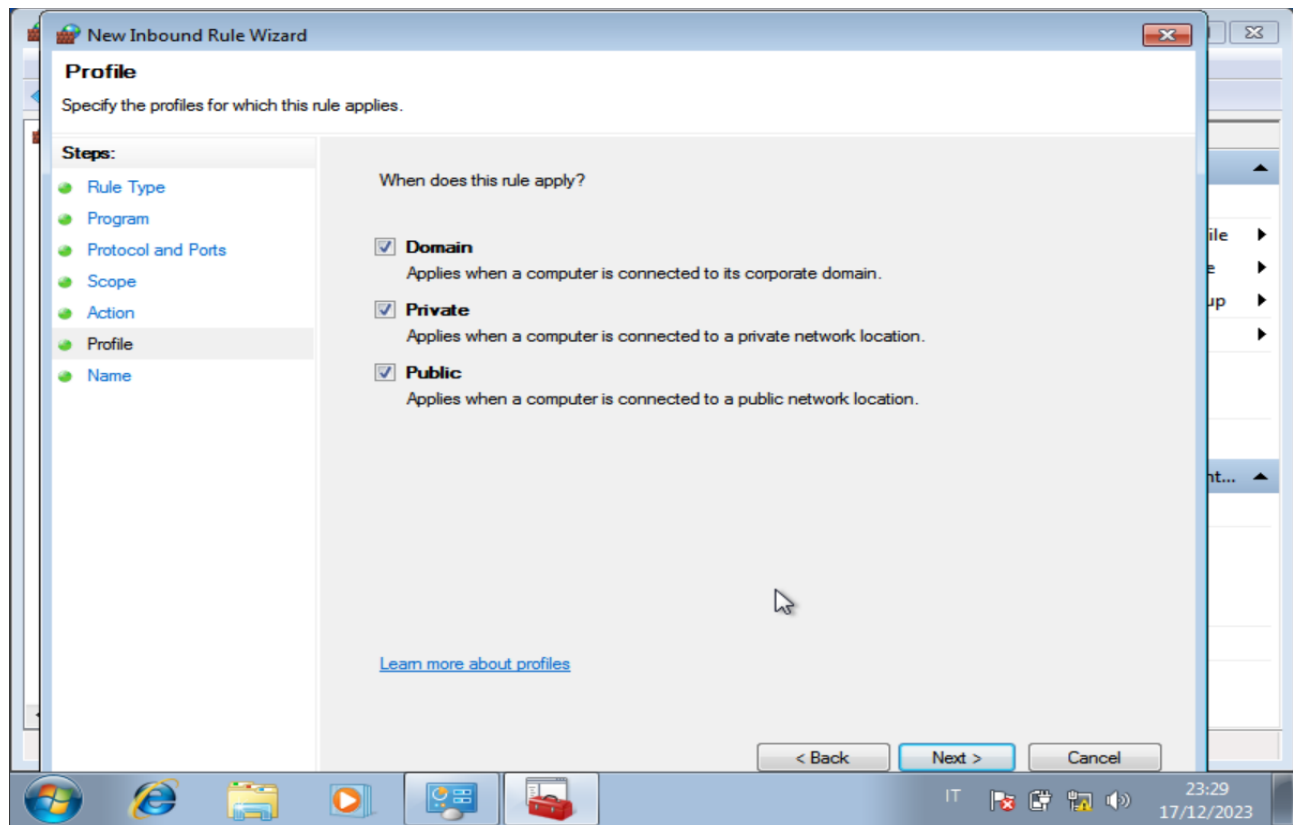


5

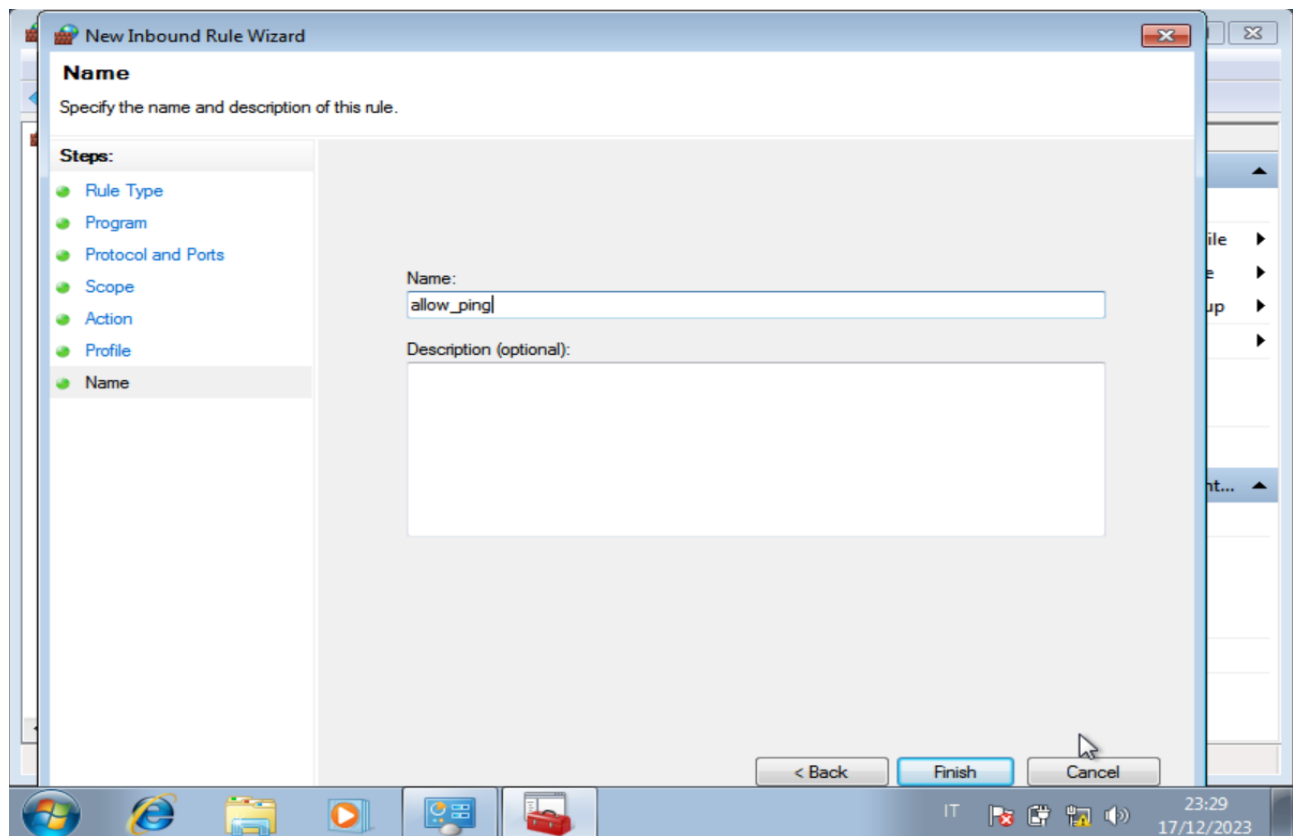


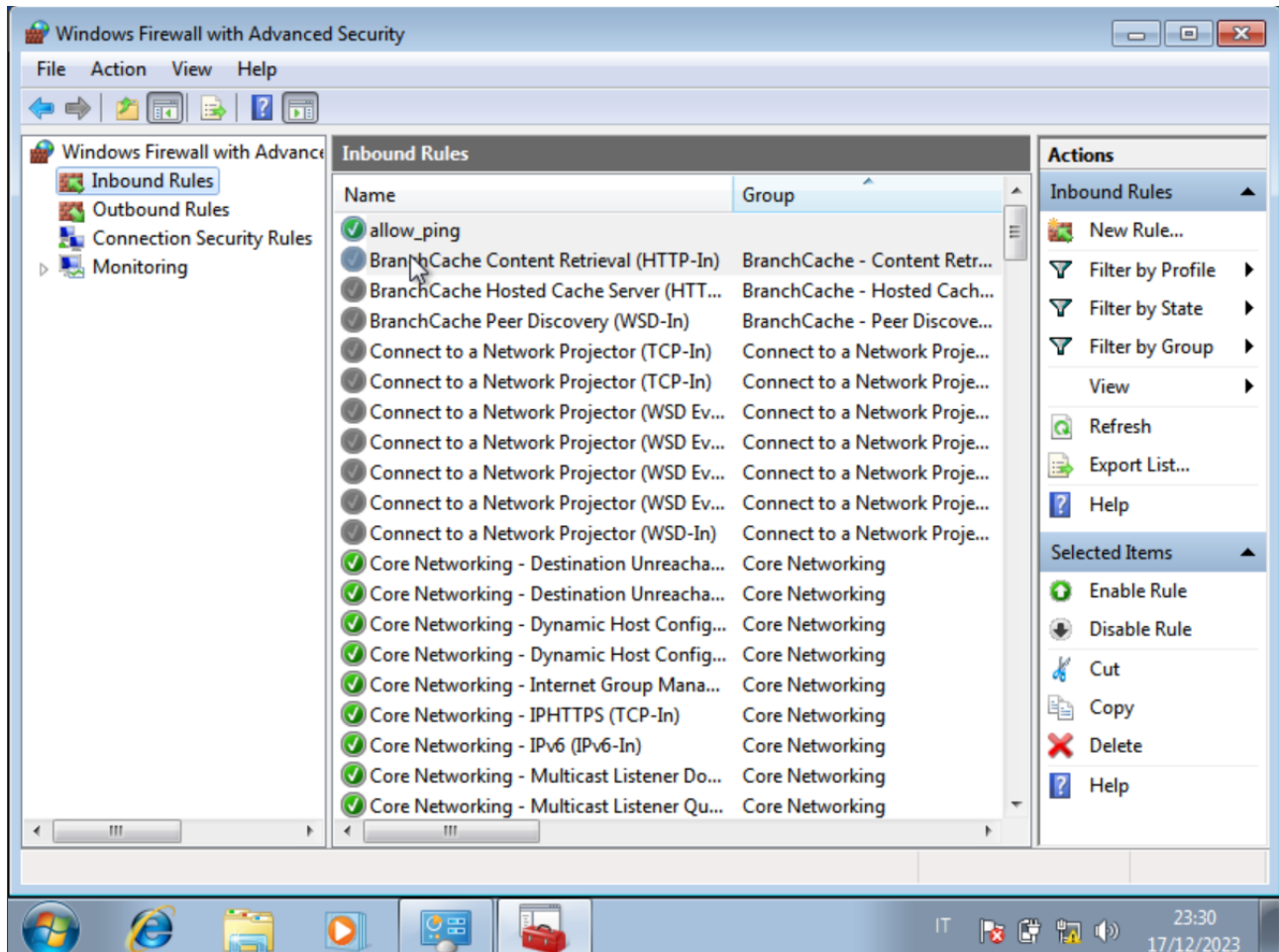


6



7





Impostata la policy , ora passiamo su Kali per impostare INETSIM.

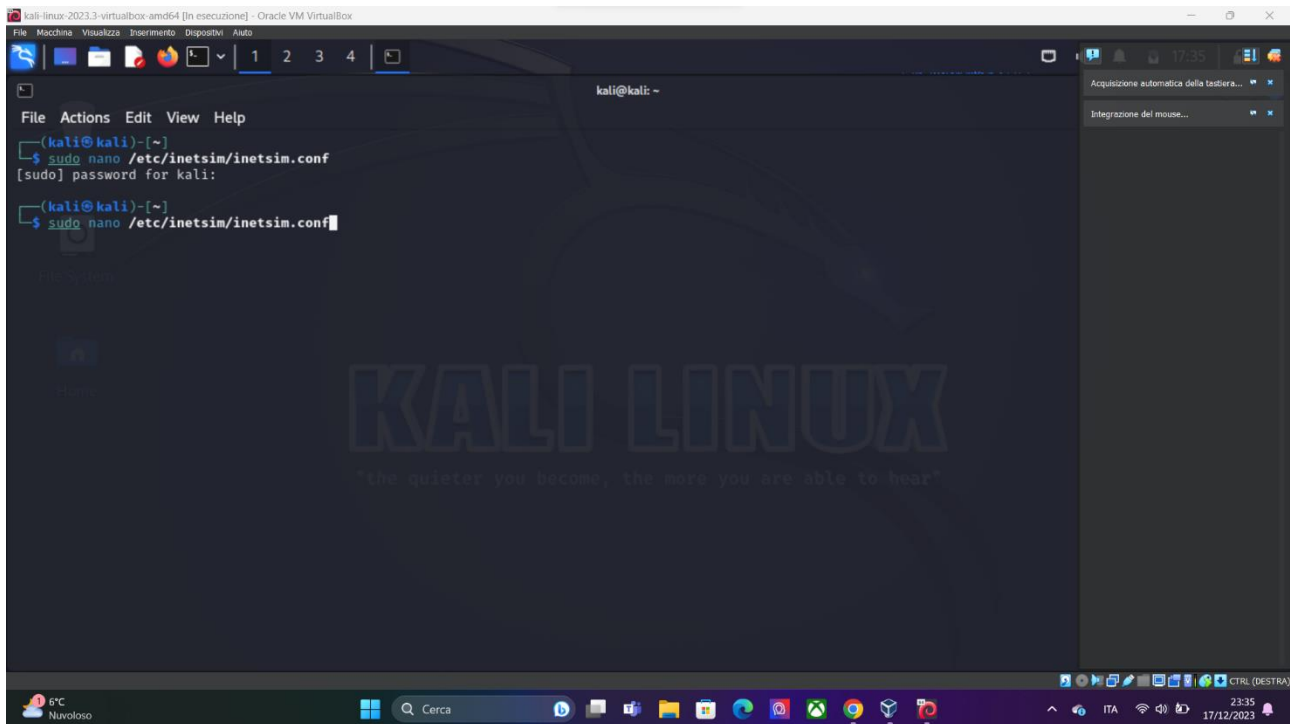
### Emulazione di servizi internet tramite **INETSIM**

Inetsim è un software creato con lo scopo di simulare servizi internet all'interno di un laboratorio (virtualBox)

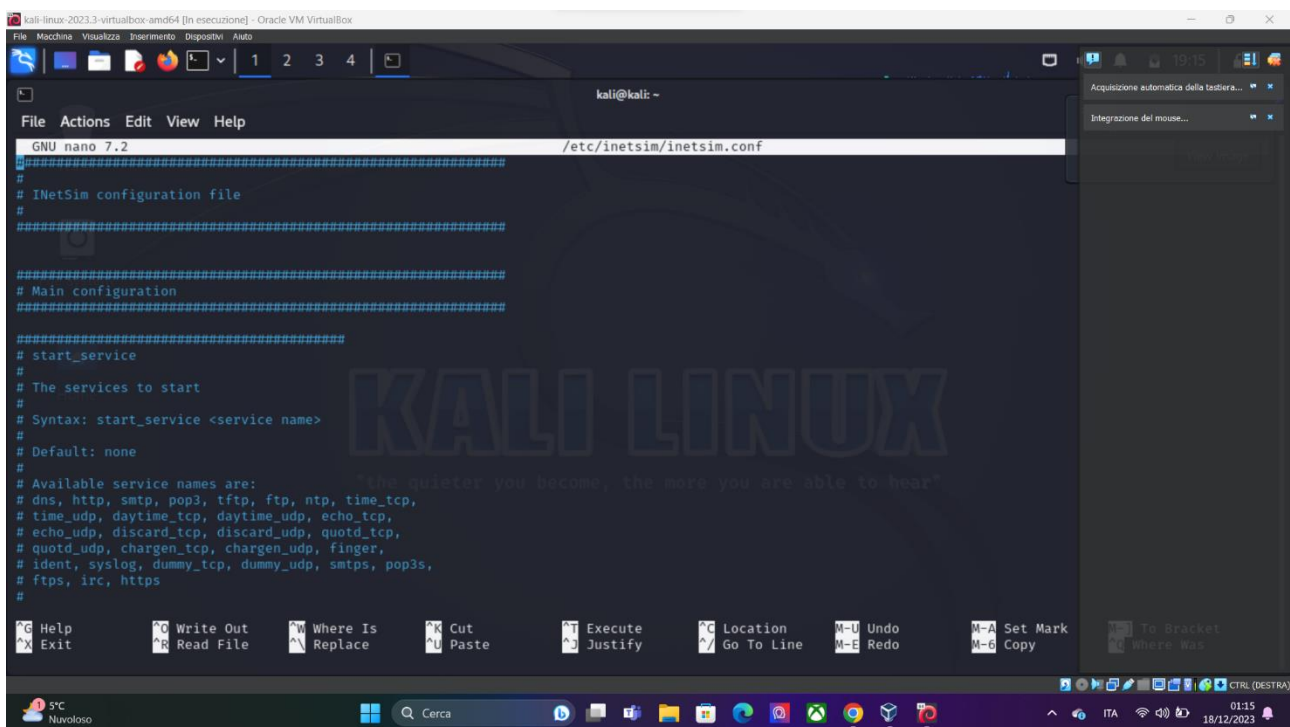
Avviato Kali entriamo nella prompt dei comandi e scriviamo in comando :

**sudo nano /etc/inetsim/inetsim.conf**



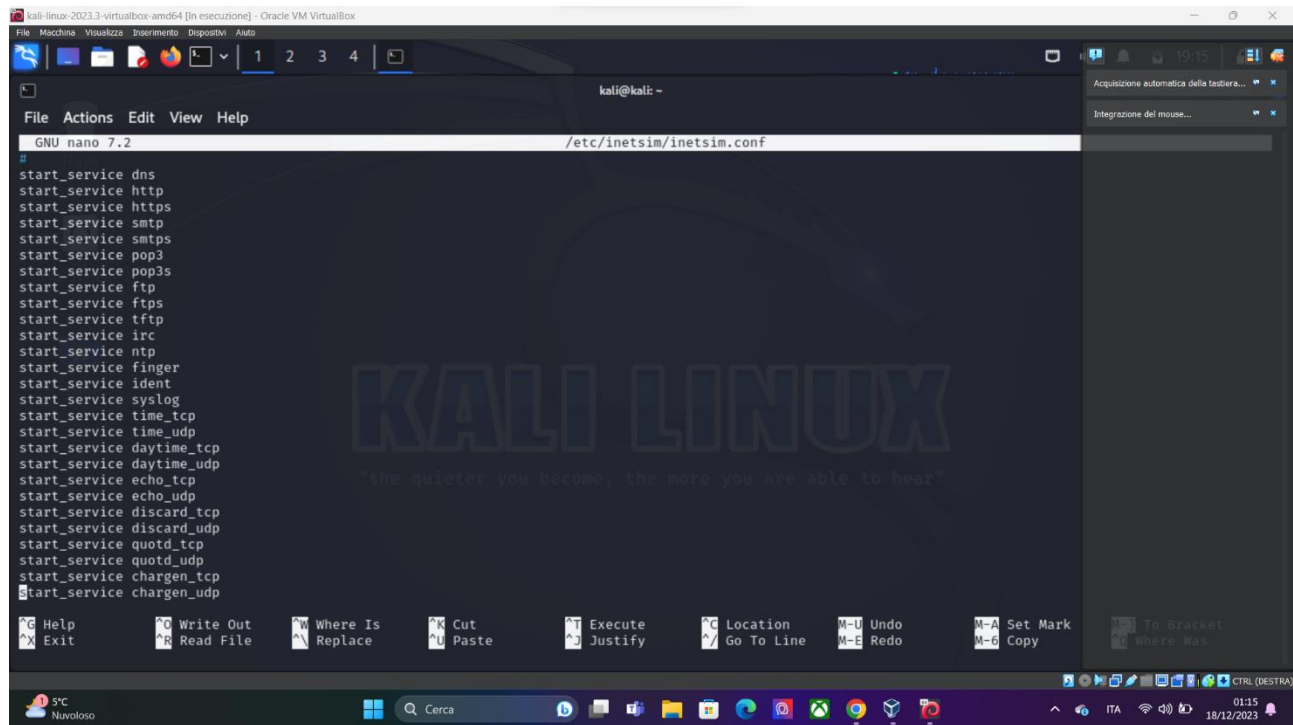


Apriamo così l'editor nano dove sono presenti le configurazioni di inetsim



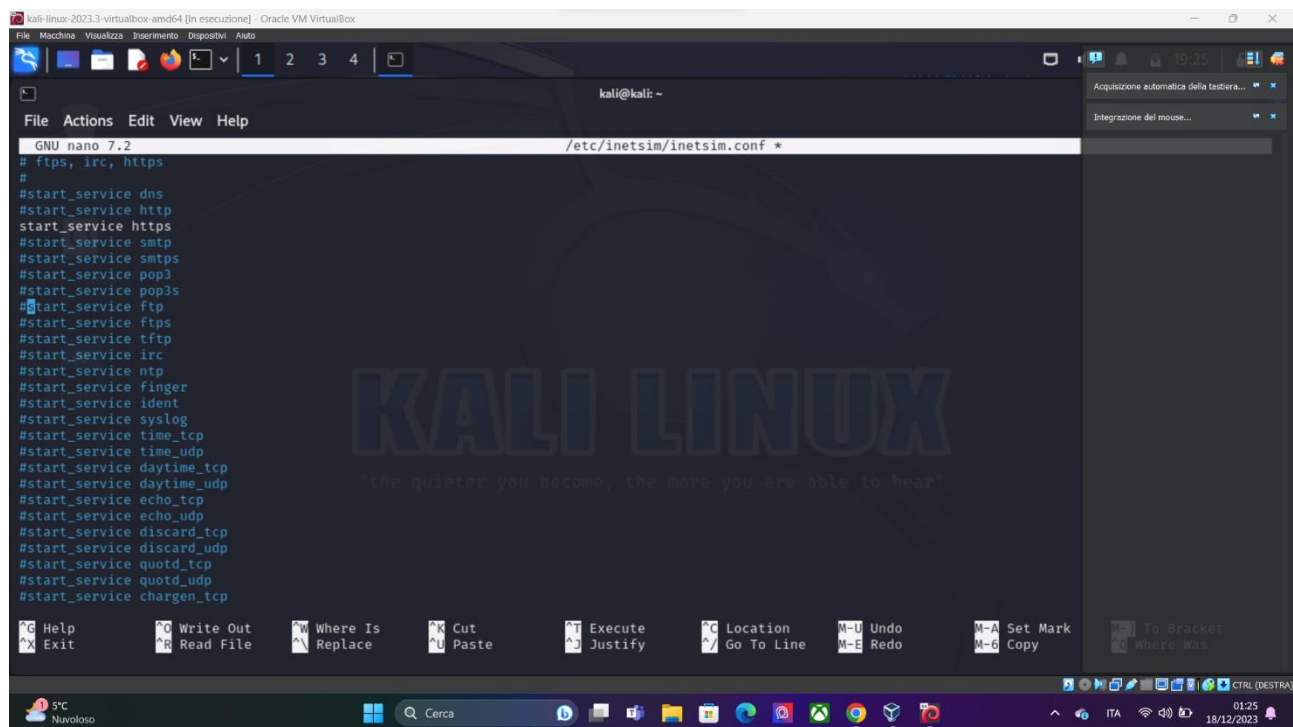
Scorriamo in basso e troviamo i protocolli che inetsim è in grado di simulare ( per questa simulazione prendiamo in esempio HTTP e HTTPS) quindi commentiamo i protocolli da escludere con un #

Da così:



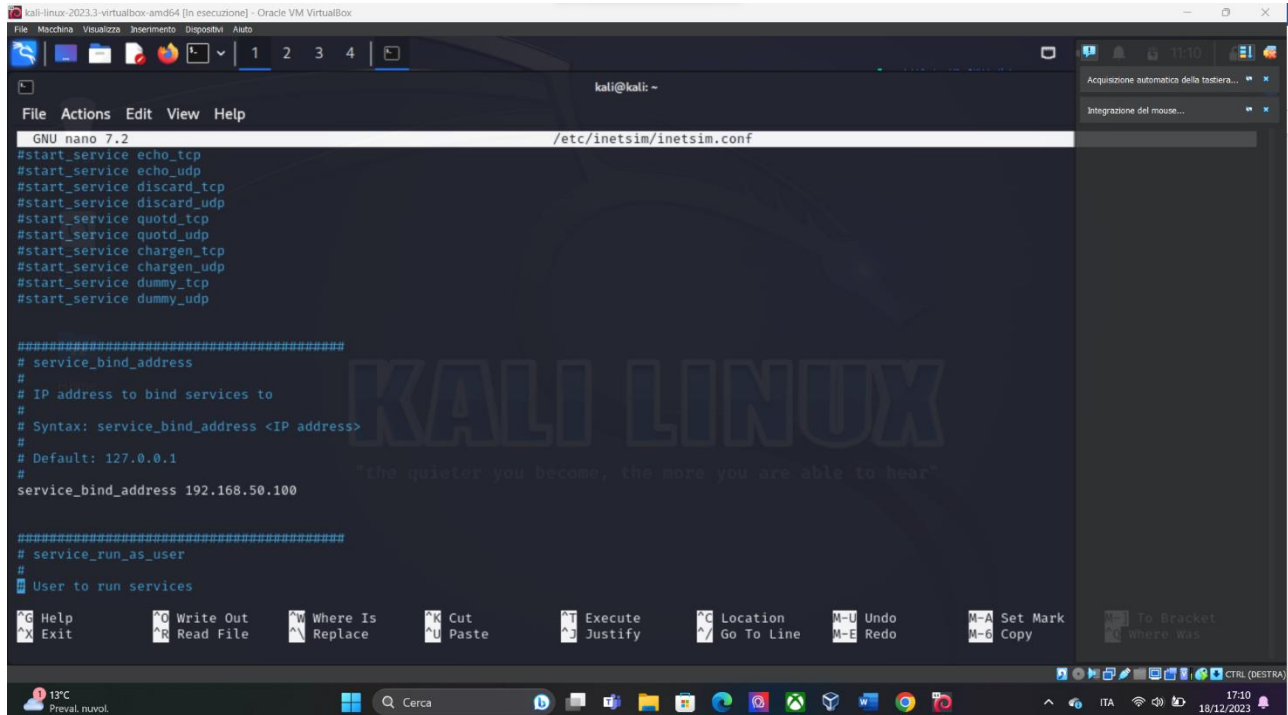
```
GNU nano 7.2 /etc/inetsim/inetsim.conf
#
start_service dns
start_service http
start_service https
start_service smtp
start_service smtps
start_service pop3
start_service pop3s
start_service ftp
start_service ftps
start_service tftp
start_service irc
start_service ntp
start_service finger
start_service 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
```

A così :



```
GNU nano 7.2 /etc/inetsim/inetsim.conf *
# 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
```

Poco più giù nel file impostiamo anche l'IP (deve essere uguale a quello di Kali)



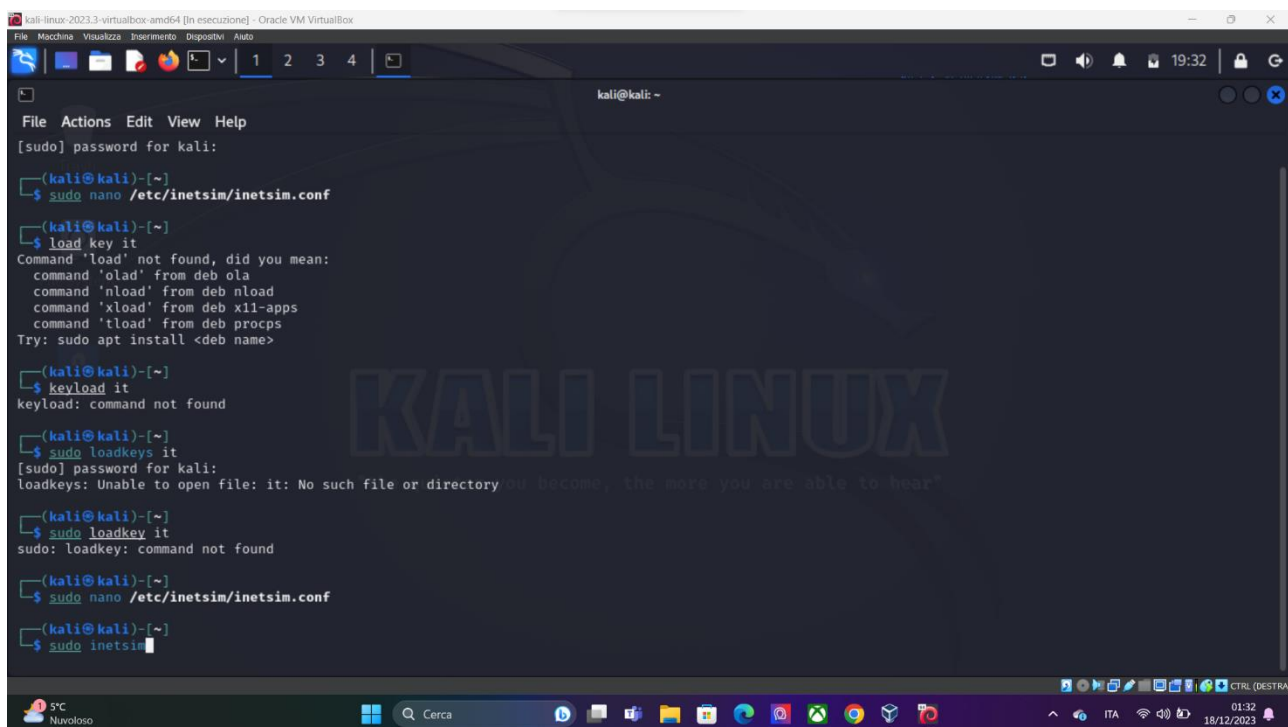
```
GNU nano 7.2 /etc/inetsim/inetsim.conf
#start_service echo_tcp
#start_service echo_udp
#start_service discard_tcp
#start_service discard_udp
#start_service quotd_tcp
#start_service quotd_udp
#start_service chargen_tcp
#start_service chargen_udp
#start_service dummy_tcp
#start_service dummy_udp

#####
# service_bind_address
#
# IP address to bind services to
#
# Syntax: service_bind_address <IP address>
#
# Default: 127.0.0.1
#
service_bind_address 192.168.50.100

#####
# service_run_as_user
#
# User to run services
```

Salviamo (Ctrl+o) e usciamo (Ctrl+X)

Ora facciamo partire inetsim con lo script sudo inetsim (ripetere la sequenza secessiva impostando una volta http e un'altra https)



```
(kali@kali)-[~]
$ sudo nano /etc/inetsim/inetsim.conf
(kali@kali)-[~]
$ load key it
Command 'load' not found, did you mean:
  command 'olad' from deb ola
  command 'nload' from deb nload
  command 'xload' from deb x11-apps
  command 'tload' from deb procs
Try: sudo apt install <deb name>
(kali@kali)-[~]
$ keyload it
keyload: command not found
(kali@kali)-[~]
$ sudo loadkeys it
[sudo] password for kali:
loadkeys: Unable to open file: it: No such file or directory
(kali@kali)-[~]
$ sudo loadkey it
sudo: loadkey: command not found
(kali@kali)-[~]
$ sudo nano /etc/inetsim/inetsim.conf
(kali@kali)-[~]
$ sudo inetsim
```

In base a http / https avremo 2 porte diverse di utilizzo

https 443

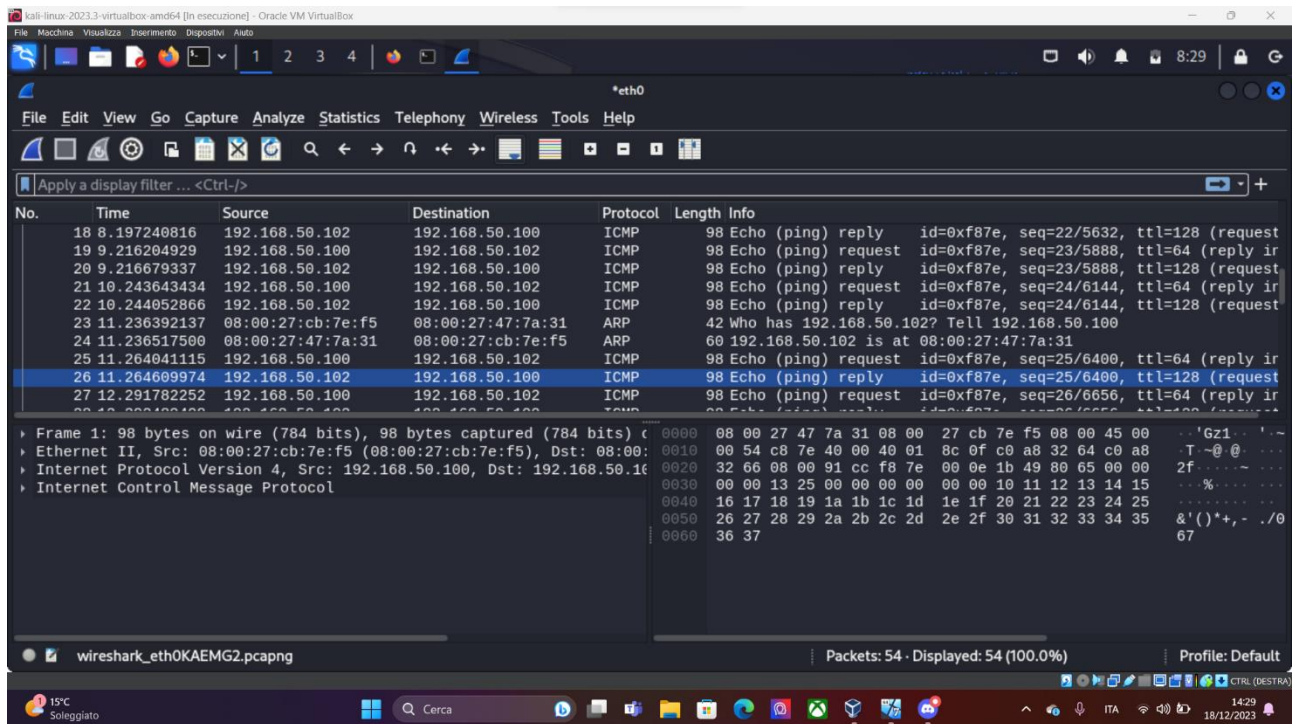
http 80

## Packet capture tramite **wireshark**

**Wireshark** è uno software dedicato all'analisi dei protocolli o packet sniffer dedicato per la soluzione dei problemi di rete.

Testiamo wireshark provando con il ping tra Kali e W7 .

Apriamo wireshark e selezioniamo eth0 , e facciamo partire il ping



Questo è ciò che vedremo :

• i 2 IP dei due Host coinvolti

• e i protocolli ICMP ( un protocollo semplice dedicato alla trasmissione di pacchetti tra 2 calcolatori ) e ARP ( protocollo utile per la comunicazione / mappatura che si basa sull'interazione tra indirizzo IP e MAC dei 2 calcolatori coinvolti )

Ora analizziamo lo scambio di pacchetti e i protocolli coinvolti durante la connessione ad un sito ( nel nostro caso fittizio )

Torniamo sull'interfaccia principale di Wireshark , impostiamo Loopback e facciamo partire l'ascolto .

Successivamente apriamo un browser di Kali e nella barra di ricerca digitiamo <https://IPkali>

Questo è quello che catturerà Wireshark:



https

Wireshark capture of an HTTPS connection. The packet list shows a sequence of TCP and TLSv1.3 packets. The packet details pane shows the structure of a TLSv1.3 packet, including the Client Hello and Server Hello messages.

No.	Time	Source	Destination	Protocol	Length	Info
7	0.061302432	192.168.50.100	192.168.50.100	TCP	66	59292 → 443 [ACK] Seq=518 Ack=1422 Win=64384 Len=0 TSval=1964
8	0.064396117	192.168.50.100	192.168.50.100	TLSv1.3	146	Change Cipher Spec, Application Data
9	0.064404476	192.168.50.100	192.168.50.100	TCP	66	443 → 59292 [ACK] Seq=1422 Ack=598 Win=65536 Len=0 TSval=1964
10	0.064511829	192.168.50.100	192.168.50.100	TLSv1.3	524	Application Data
11	0.064514841	192.168.50.100	192.168.50.100	TCP	66	443 → 59292 [ACK] Seq=1422 Ack=1056 Win=65152 Len=0 TSval=1964
12	0.064735241	192.168.50.100	192.168.50.100	TLSv1.3	321	Application Data
13	0.081821225	192.168.50.100	192.168.50.100	TLSv1.3	798	Application Data, Application Data, Application Data, Application Data
14	0.086063646	192.168.50.100	192.168.50.100	TCP	66	59292 → 443 [ACK] Seq=1056 Ack=2410 Win=65536 Len=0 TSval=1964
15	0.086142403	192.168.50.100	192.168.50.100	TLSv1.3	90	Application Data
16	0.086152604	192.168.50.100	192.168.50.100	TCP	54	443 → 59292 [RST] Seq=2410 Win=0 Len=0

Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0  
Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00  
Internet Protocol Version 4, Src: 192.168.50.100, Dst: 192.168.50.100  
Transmission Control Protocol, Src Port: 59292, Dst Port: 443, Seq: 518, Win: 64384, Len: 0

come vediamo sono stati usati 2 tipi di protocollo il TCP (dedito a creare un canale di trasmissione primaria e alla trasmissione completa dei dati da A pc B), e il TLS (protocollo dedito alla crittografia dei pacchetti trasportati)

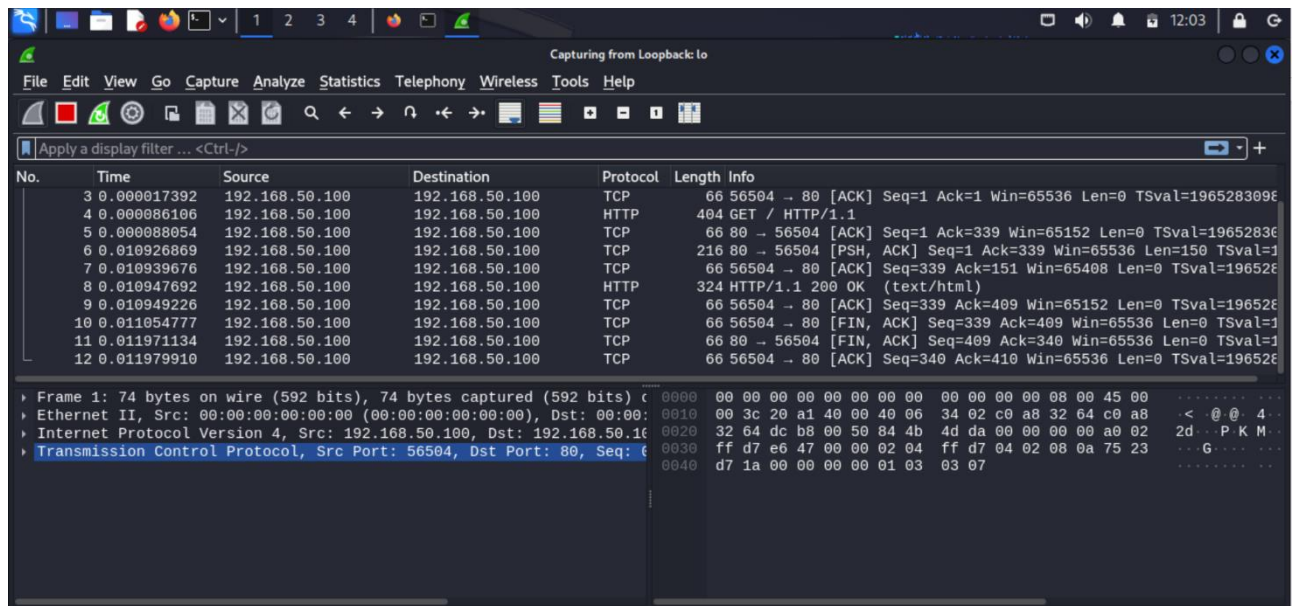
Se invece nella barra di ricerca aggiungiamo /sample.txt

Wireshark capture of an HTTP connection. The packet list shows a sequence of TCP and HTTP packets. The packet details pane shows the structure of an HTTP packet, including the GET request and the 200 OK response.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.50.100	192.168.50.100	TCP	74	55224 → 443 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1
2	0.000007551	192.168.50.100	192.168.50.100	TCP	74	443 → 55224 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495
3	0.000013938	192.168.50.100	192.168.50.100	TCP	66	55224 → 443 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=196497435
4	0.000769823	192.168.50.100	192.168.50.100	TLSv1.3	687	Client Hello
5	0.000773113	192.168.50.100	192.168.50.100	TCP	66	443 → 55224 [ACK] Seq=1 Ack=622 Win=64896 Len=0 TSval=1964974
6	0.022979381	192.168.50.100	192.168.50.100	TLSv1.3	1487	Server Hello, Change Cipher Spec, Application Data, Application Data
7	0.022991270	192.168.50.100	192.168.50.100	TCP	66	55224 → 443 [ACK] Seq=622 Ack=1422 Win=64384 Len=0 TSval=1964
8	0.025785083	192.168.50.100	192.168.50.100	TLSv1.3	146	Change Cipher Spec, Application Data
9	0.025864169	192.168.50.100	192.168.50.100	TLSv1.3	534	Application Data
10	0.025972594	192.168.50.100	192.168.50.100	TLSv1.3	321	Application Data

Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0  
Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00  
Internet Protocol Version 4, Src: 192.168.50.100, Dst: 192.168.50.100  
Transmission Control Protocol, Src Port: 55224, Dst Port: 443, Seq: 55224, Win: 65495, Len: 0

Ora proviamo lo stesso procedimento, ma questa volta settiamo inetsim solo su http



La differenza che salta subito all'occhi e l'assenza di protocolli di crittografia

Anche con l'aggiunta di : sample.txt la situazione resta immutata ,le nostre richieste al server e le risposte che riceviamo sono alla portata di qualsiasi malintenzionato intento ad ascoltare / rubare le nostre attività in rete

FINE