

Beatrice Folino

[Windows7 firewall - inetsim - Wireshark]

EPICODE - CYBERSECURITY CLASS [W3D4 Pratica]

12 novembre 2023





L'esercizio consiste nella configurazione di una policy sul firewall del nostro laboratorio virtuale, in questo caso su Windows7, e in una packet capture con Wireshark attraverso l'utilizzo di InetSim su Kali Linux.

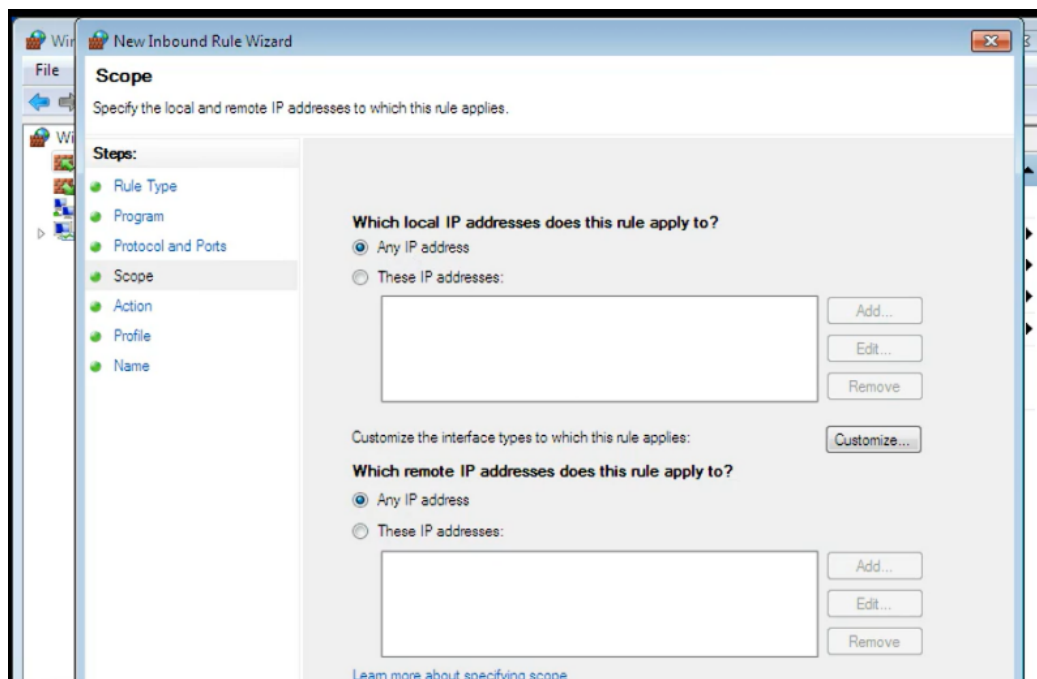
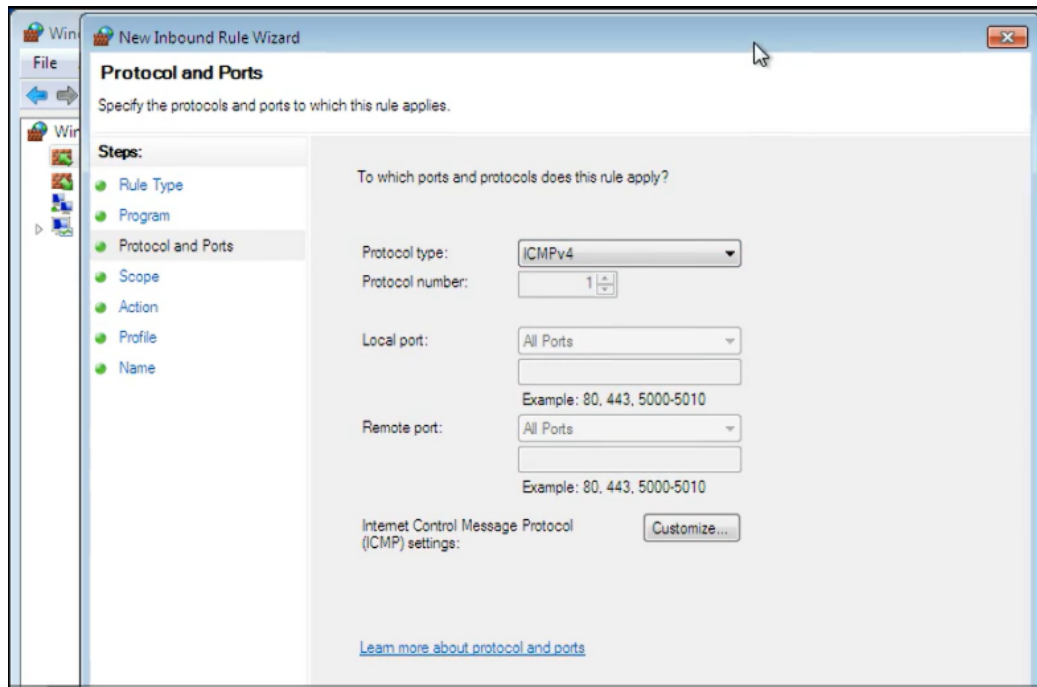
CONFIGURAZIONE POLICY W7 FIREWALL

Per permettere il ping tra le varie macchine virtuali del nostro laboratorio, è necessario stabilire una policy su Windows7; per questa esercitazione prenderemo in considerazione solo Kali Linux e Windows7 (da qui in poi chiamato anche W7 per brevità) che sono direttamente coinvolti nello svolgersi dell'esercizio.

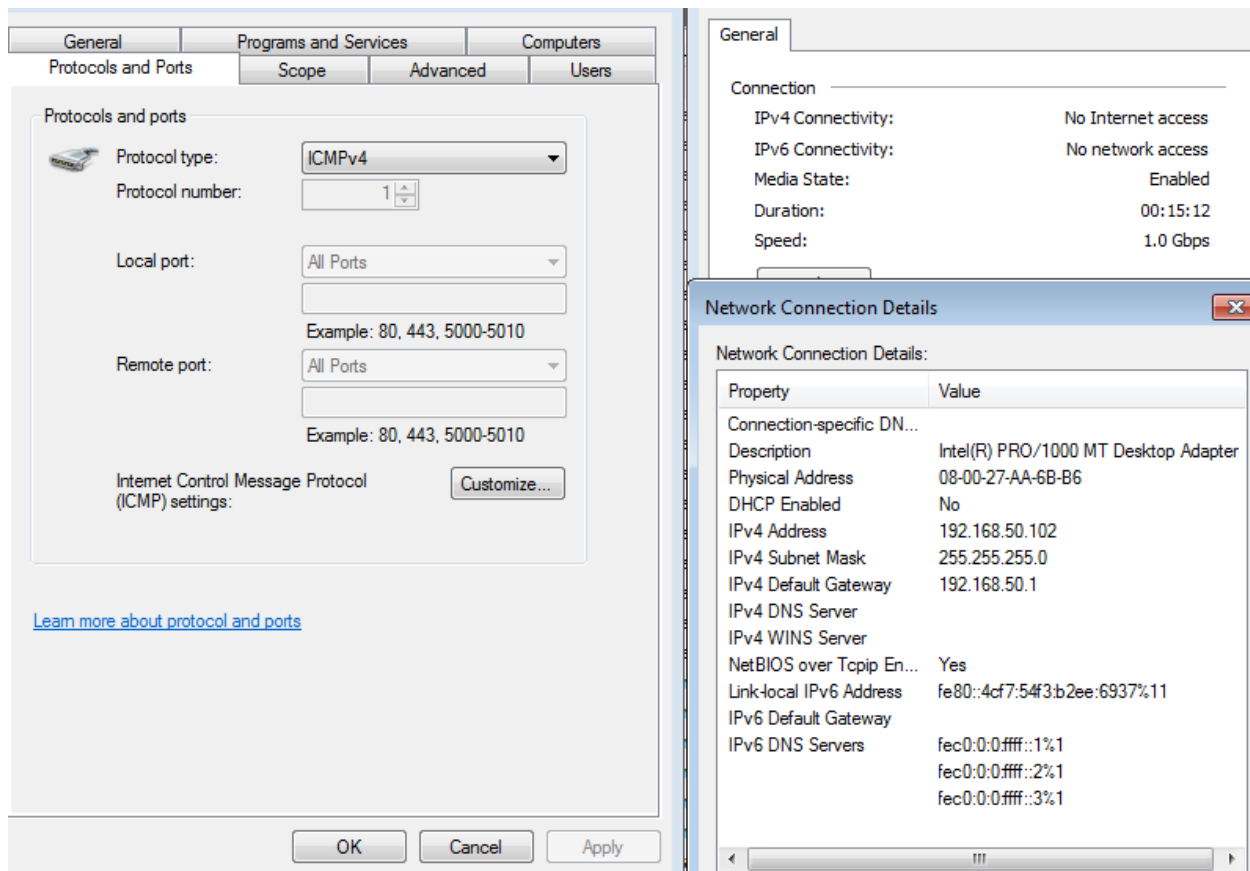
Per prima cosa, dopo aver settato gli IP statici, come da prima esercitazione [W1D4] presente in questa repository, occorre dirigersi nella sezione del pannello di controllo di W7 seguendo il percorso

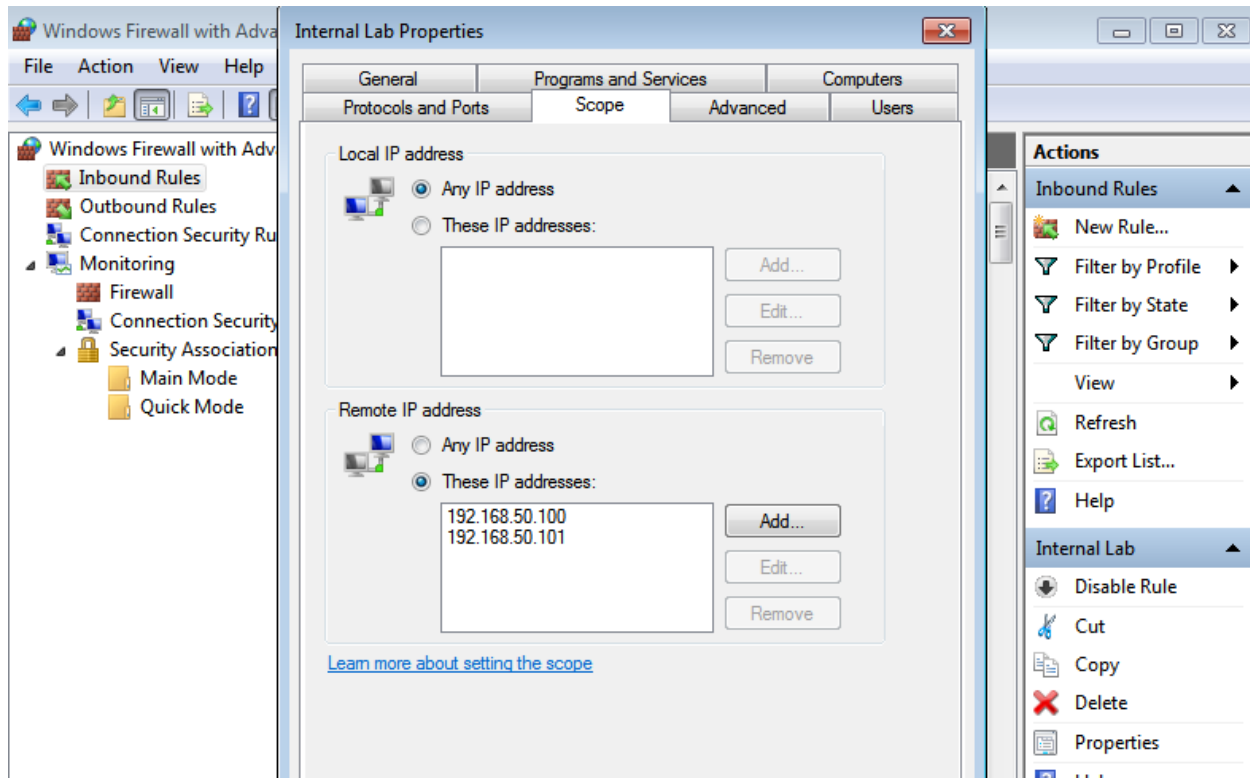
Control Panel → System and security → Windows Firewall → Advanced settings → Inbound rules → New rule → custom rule

Da qui è possibile impostare una regola che permetta le connessioni in entrata di qualsiasi indirizzo IP, come da immagini seguenti:

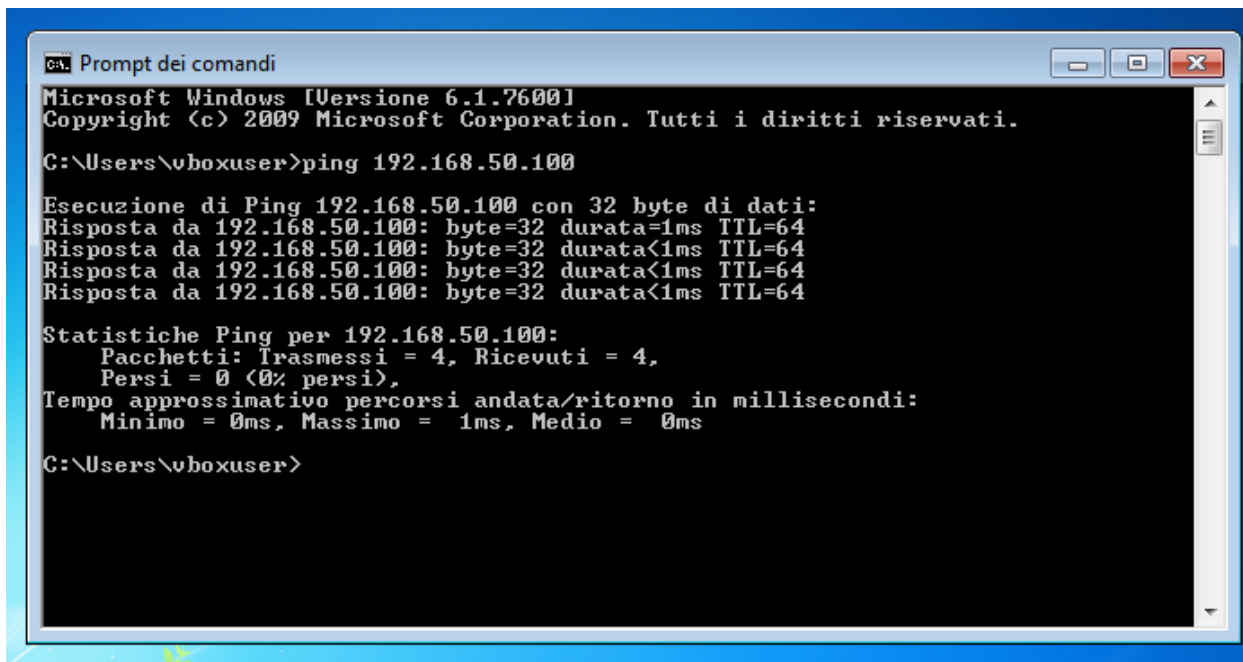


Successivamente, nelle proprietà della regola appena creata, ho specificato per quali IP dovesse tenere in considerazione sempre, corrispondenti agli IP di Kali Linux e Metasploitable (vedasi sempre es. [W1D4] presente in questa repository):





Il ping avviene correttamente da e verso Kali Linux:



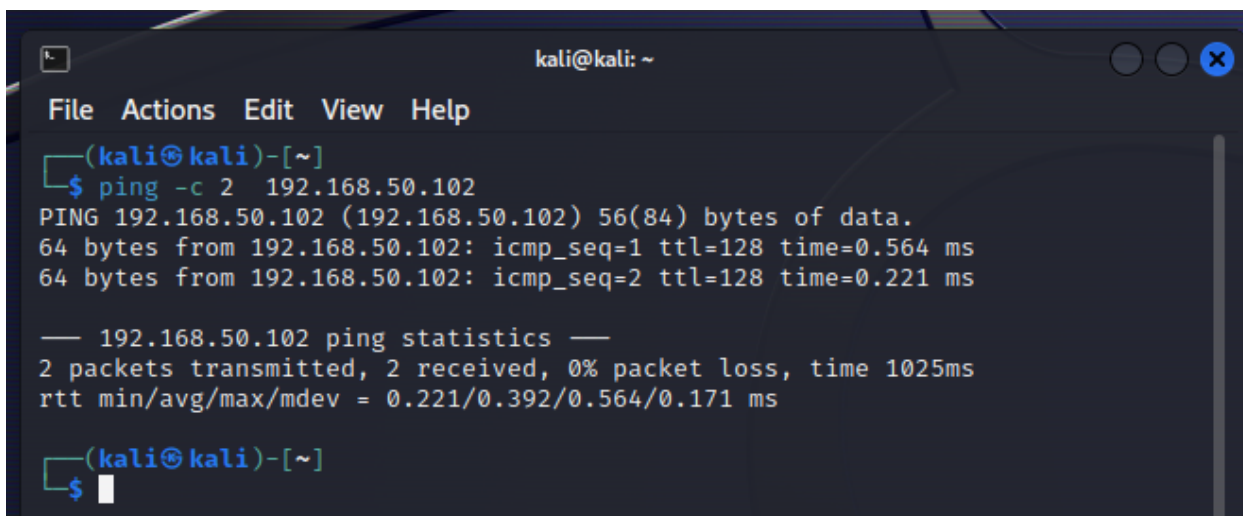
```
CA Prompt dei comandi
Microsoft Windows [Versione 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. Tutti i diritti riservati.

C:\Users\ vboxuser>ping 192.168.50.100

Esecuzione di Ping 192.168.50.100 con 32 byte di dati:
Risposta da 192.168.50.100: byte=32 durata=1ms TTL=64
Risposta da 192.168.50.100: byte=32 durata<1ms TTL=64
Risposta da 192.168.50.100: byte=32 durata<1ms TTL=64
Risposta da 192.168.50.100: byte=32 durata<1ms TTL=64

Statistiche Ping per 192.168.50.100:
    Pacchetti: Trasmessi = 4, Ricevuti = 4,
    Persi = 0 (0% persi),
    Tempo approssimativo percorsi andata/ritorno in millisecondi:
        Minimo = 0ms, Massimo = 1ms, Medio = 0ms

C:\Users\ vboxuser>
```



```
kali@kali: ~
File Actions Edit View Help

(kali@kali)-[~]
└─$ ping -c 2 192.168.50.102
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=0.564 ms
64 bytes from 192.168.50.102: icmp_seq=2 ttl=128 time=0.221 ms

— 192.168.50.102 ping statistics —
2 packets transmitted, 2 received, 0% packet loss, time 1025ms
rtt min/avg/max/mdev = 0.221/0.392/0.564/0.171 ms

(kali@kali)-[~]
└─$
```



SETTAGGIO INETSIM

Apriamo Kali Linux su macchina virtuale e da terminale seguiamo i seguenti passaggi per aprire il file **inetsim.conf**

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

Una volta inserita la password, ci verrà data la possibilità di modificare il file **inetsim.conf** e di settarlo lasciando attivi solo i servizi di nostro interesse (http e https). Antepoendo il “#”, infatti, si “commenteranno” le stringhe da escludere, rendendole non eseguibili.

```
#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
```

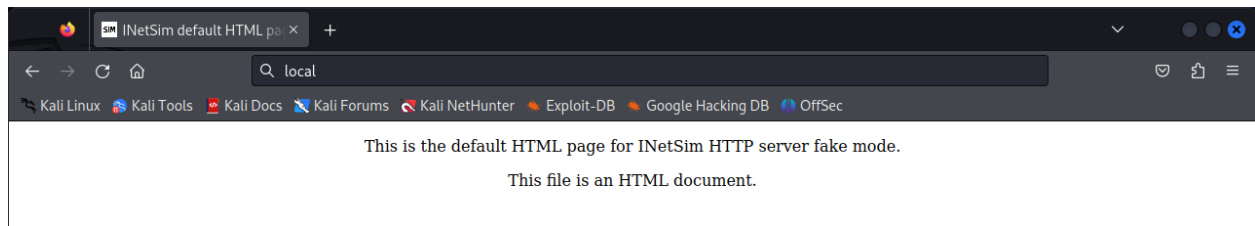
Inoltre, andremo a impostare l'indirizzo 127.0.0.1 come IP address per il fake service che andremo a utilizzare.

```
#####  
# 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 il file coi comandi **Ctrl+X / Y**, potremo andare a digitare sul terminale il comando **sudo inetsim** che dovrebbe restituirci l'avvio della simulazione.

```
(kali@kali)-[/etc/inetsim]  
$ sudo inetsim  
INetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg  
Main logfile '/var/log/inetsim/main.log' does not exist. Trying to create it.  
..  
Main logfile '/var/log/inetsim/main.log' successfully created.  
Sub logfile '/var/log/inetsim/service.log' does not exist. Trying to create i  
t...  
Sub logfile '/var/log/inetsim/service.log' successfully created.  
Debug logfile '/var/log/inetsim/debug.log' does not exist. Trying to create i  
t...  
Debug logfile '/var/log/inetsim/debug.log' successfully created.  
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 9635) ==  
Session ID: 9635  
Listening on: 127.0.0.1  
Real Date/Time: 2023-11-12 05:42:45  
Fake Date/Time: 2023-11-12 05:42:45 (Delta: 0 seconds)  
Forking services...  
* http_80_tcp - started (PID 9637)  
* https_443_tcp - started (PID 9638)  
done.  
Simulation running.  
█
```

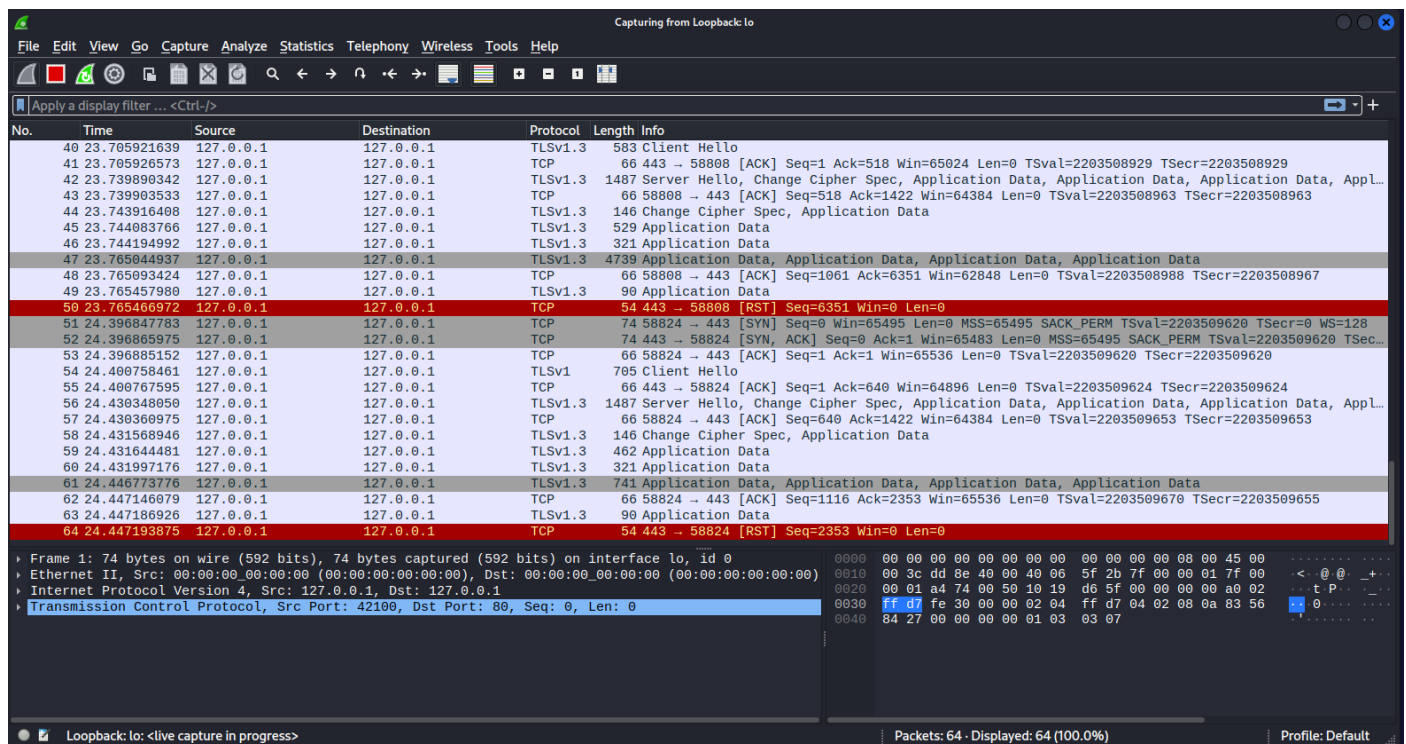

A questo punto, potremo aprire mozilla da Kali Linux sulla pagina <http://localhost> e vedremo che in risposta verrà visualizzata la pagina di “server fake mode”.



PACKET CAPTURE CON WIRESHARK

Avviamo ora Wireshark su Kali Linux e per comodità scegliamo di analizzare, per ora, la rete LOOPBACK.

Se andremo a richiamare delle pagine su mozilla, es. comandi GET come `http://localhost/sample.txt` oppure `https://localhost/sample.jpg` Wireshark sarà in grado di catturare i pacchetti richiamati.



© 2005 Blackwell Publishing Ltd

Wireshark

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

SS

No.	Time	Source	Destination	Protocol	Length	Info
20	8.949852416	127.0.0.1	127.0.0.1	TLSv1.3	148	Change Cipher Spec, Application Data
21	8.949957444	127.0.0.1	127.0.0.1	TLSv1.3	531	Application Data
22	8.950888216	127.0.0.1	127.0.0.1	TLSv1.3	323	Application Data
23	8.953499111	127.0.0.1	127.0.0.1	TLSv1.3	1489	Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data, ...
24	8.955466836	127.0.0.1	127.0.0.1	TCP	68	47146 → 443 [ACK] Seq=648 Ack=1422 Win=64384 Len=0 TSval=2203813915 TSecr=2203813915
25	8.955375581	127.0.0.1	127.0.0.1	TLSv1.3	148	Change Cipher Spec, Application Data
26	8.955386069	127.0.0.1	127.0.0.1	TCP	68	443 → 47146 [ACK] Seq=1422 Ack=729 Win=65536 Len=0 TSval=2203813917 TSecr=2203813917
27	8.955664542	127.0.0.1	127.0.0.1	TLSv1.3	323	Application Data
28	8.969319671	127.0.0.1	127.0.0.1	TLSv1.3	4741	Application Data, Application Data, Application Data, Application Data
29	8.969366738	127.0.0.1	127.0.0.1	TCP	68	47158 → 443 [ACK] Seq=1183 Ack=6351 Win=62848 Len=0 TSval=2203813931 TSecr=2203813932
30	8.969884475	127.0.0.1	127.0.0.1	TLSv1.3	92	Application Data
31	8.968513943	127.0.0.1	127.0.0.1	TCP	56	443 → 47158 [RST] Seq=6351 Win=0 Len=0
32	8.997549777	127.0.0.1	127.0.0.1	TCP	68	47146 → 443 [ACK] Seq=729 Ack=1677 Win=65536 Len=0 TSval=2203813960 TSecr=2203813918
33	8.997562093	127.0.0.1	127.0.0.1	TLSv1.3	323	Application Data
34	8.997566893	127.0.0.1	127.0.0.1	TCP	68	47146 → 443 [ACK] Seq=729 Ack=1932 Win=65408 Len=0 TSval=2203813960 TSecr=2203813960
35	9.005597178	SkyUK 55:5e:9e		6x7374	68	Broadcast
36	11.598503646	192.168.0.1	224.0.0.1	IGMPv3	62	Membership Query, general
37	11.598504040	f800::c2a3:6eff:feb...ff02::1		ICMPv6	92	Multicast Listener Query
38	11.673903956	f800::dc91:4deb:34c...ff02::16		ICMPv6	132	Multicast Listener Report Message v2
39	12.007265192	SkyUK 55:5e:9e		6x7374	68	Broadcast
40	13.917720671	127.0.0.1	127.0.0.1	TLSv1.3	92	Application Data
41	13.917766189	127.0.0.1	127.0.0.1	TLSv1.3	92	Application Data
42	13.921292760	127.0.0.1	127.0.0.1	TLSv1.3	92	Application Data
43	13.921229241	127.0.0.1	127.0.0.1	TCP	56	47146 → 443 [RST] Seq=745 Win=0 Len=0
44	15.617071903	SkyUK 55:5e:9e		6x7374	68	Broadcast

Frame 24: 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: 47146, Dst Port: 443, Seq: 648, Ack: 1422, Len: 0

```

0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00  ....E....
0010  45 00 00 34 0b 0d 00 00  40 0e 01 34 7f 00 00 01  .....
0020  7f 00 00 01 b8 2a 01 bb  d2 ec 38 f8 a8 09 5c e2  .....
0030  80 10 01 01 ff 7e 28 00  00 01 09 0a 83 5b 88 1b  .....
0040  83 5b 88 1b

```

Per agevolarci, una volta lanciata la packet capture, potremo filtrare i risultati attraverso il comando `ip.src == 127.0.0.1`, saranno così visualizzati solo i pacchetti provenienti dal quell'IP sorgente.

