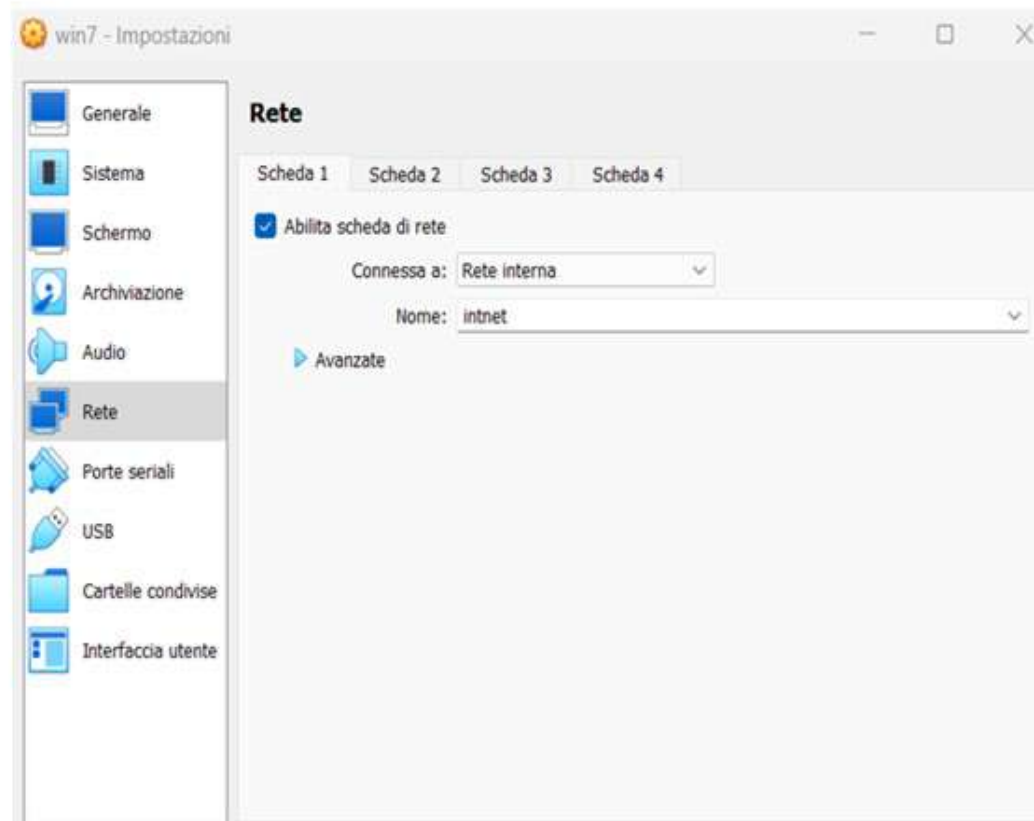


Esercitazione 29.09.2023 Epicode

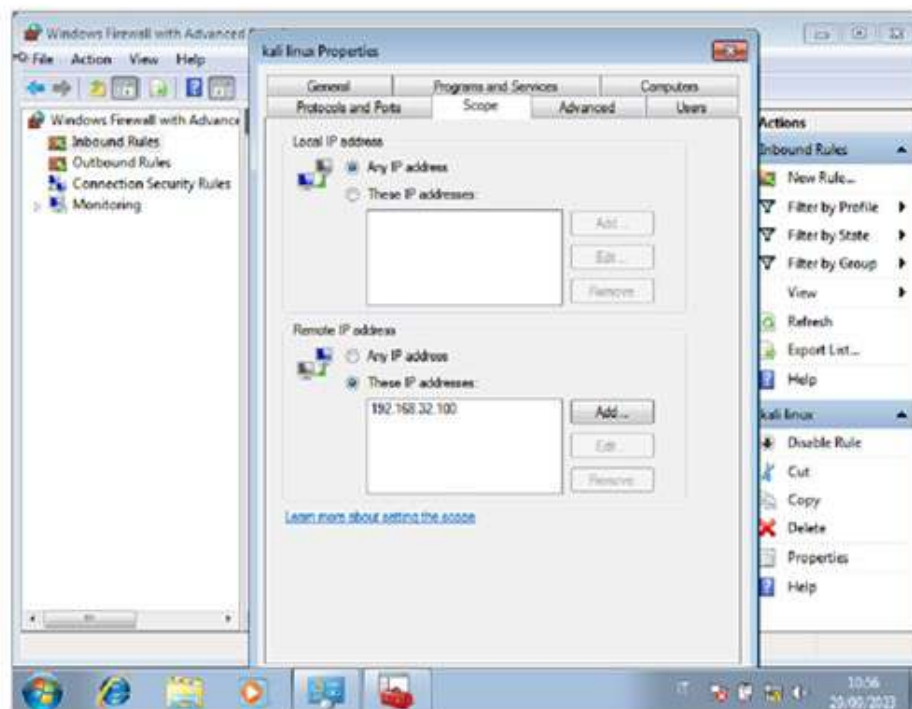
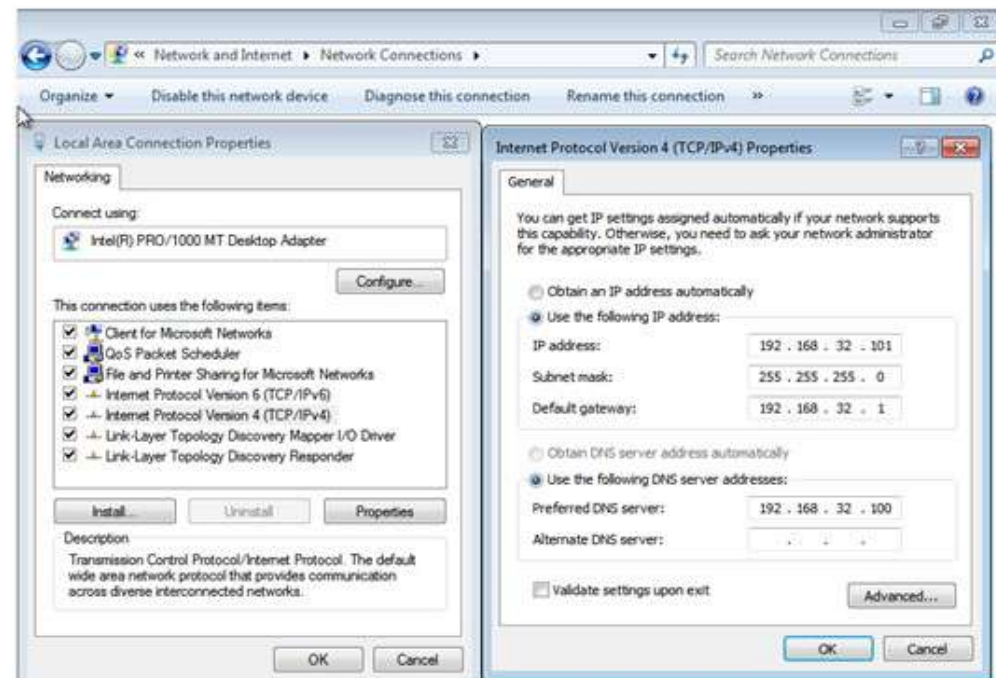
Creazione di un laboratorio virtuale con i sistemi operativi Kali Linux e Windows 7

- Simulazione di una rete client (Windows 7) server (Kali Linux)
- Il Client richiede tramite web browser una risorsa all'hostname epicode.internal (prima http e poi https)



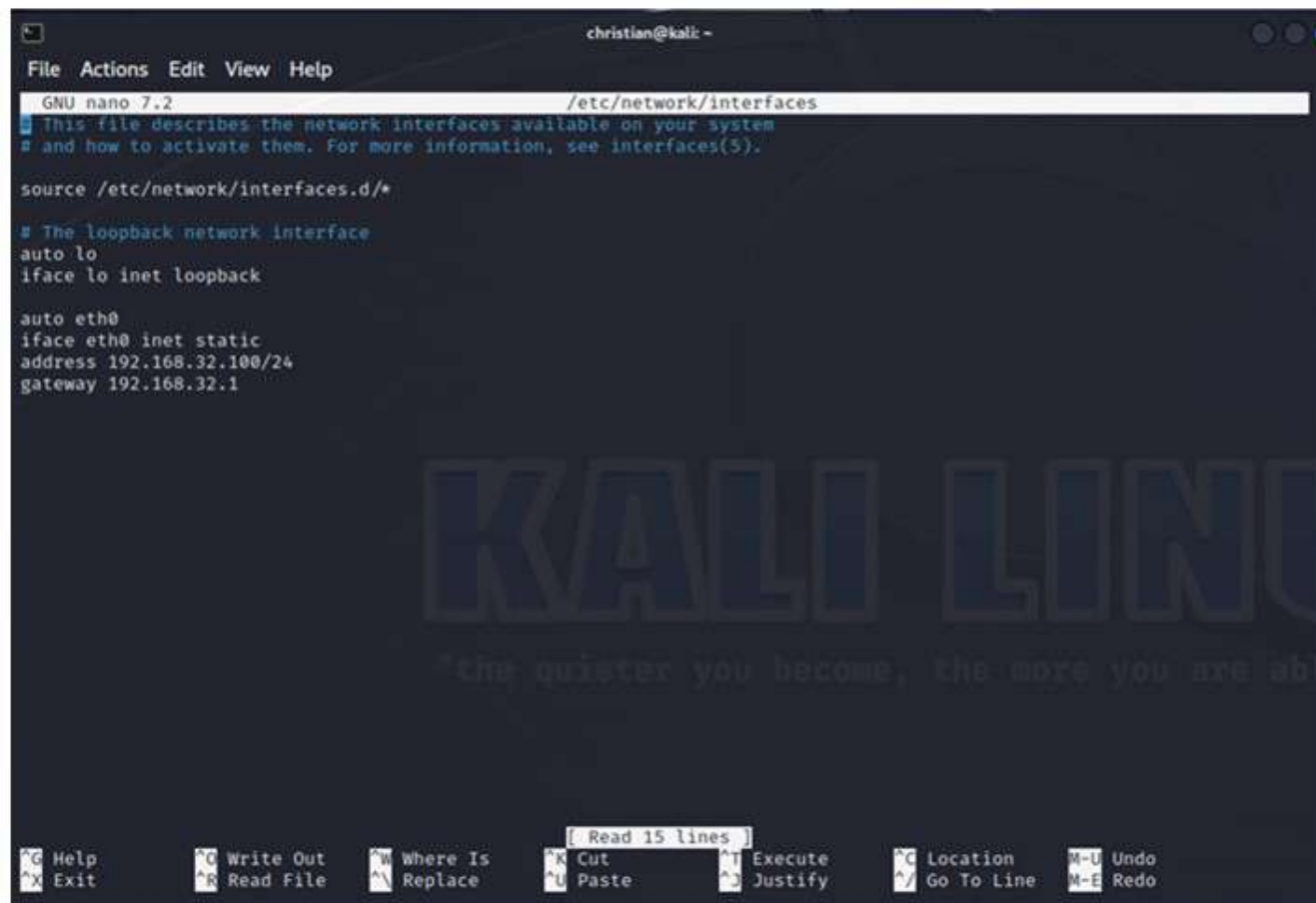
Impostare tutte e due i S.O. a “rete interna” perchè **KALI LINUX** effettuerà la funzione da DNS server per far visualizzare a **WINDOWS 7** il sito “epicode.internal”

Configurare l'ip address di **WINDOWS 7** a 192.168.32.101/24 e nella stessa schermata delle proprietà aggiungere sulla voce "Server DNS" l'indirizzo IP della macchina di **KALI LINUX** cioè 192.168.32.100/24



Dopo aver configurato l'IP bisogna creare una nuova regola delle reti in entrata del FIREWALL del client, aggiungendo il permesso di entrata al server

Configurare l'IP anche nella macchina **KALI LINUX** con il comando **sudo nano /etc/network/interfaces** per poter far lavorare entrambi i S.O. nella stessa network, IP address 192.168.32.100/24



```
christian@kali: ~  
File Actions Edit View Help  
GNU nano 7.2 /etc/network/interfaces  
# This file describes the network interfaces available on your system  
# and how to activate them. For more information, see interfaces(3).  
  
source /etc/network/interfaces.d/*  
  
# The loopback network interface  
auto lo  
iface lo inet loopback  
  
auto eth0  
iface eth0 inet static  
address 192.168.32.100/24  
gateway 192.168.32.1  
  
KALI LINUX  
"the quieter you become, the more you are able to hear"  
  
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute  
^X Exit      ^R Read File  ^N Replace    ^U Paste      ^J Justify  
^C Location  ^M-U Undo  
^_/ Go To Line ^M-E Redo  
Read 15 lines
```


Dopo aver impostato entrambi gli indirizzi IP effettuare un ping per verificare se le due macchine sono state collegate alla stessa network (sniff con wireshark per un'ulteriore conferma)

```

christian@kali: ~
File Actions Edit View Help

christian@kali:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.32.180 netmask 255.255.255.0 broadcast 192.168.32.255
    inet6 fe80::a00:27ff:fe58:bcca prefixlen 64 scopeid 0<2048link>
    ether 08:00:12:75:0b:ca txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 37 bytes 2496 (2.4 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0<1040host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 4 bytes 240 (240.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 4 bytes 240 (240.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

christian@kali:~$ ping 192.168.32.181
PING 192.168.32.181 (192.168.32.181) 56(84) bytes of data:
64 bytes from 192.168.32.181: icmp_seq=1 ttl=120 time=0.817 ms
64 bytes from 192.168.32.181: icmp_seq=2 ttl=120 time=0.688 ms
64 bytes from 192.168.32.181: icmp_seq=3 ttl=120 time=0.490 ms
64 bytes from 192.168.32.181: icmp_seq=4 ttl=120 time=0.602 ms
^C
    — 192.168.32.181 ping statistics —
    4 packets transmitted, 4 received, 0% packet loss, time 3073ms
    rtt min/avg/max/mdev = 0.490/0.651/0.817/0.110 ms

christian@kali:~$


```

The image shows a Wireshark packet capture on the 'eth0' interface. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. The toolbar contains icons for various functions like opening files, saving, and packet selection. The main display area shows a list of 16 packets, all of which are ICMP Echo (ping) requests or replies. The packet list has columns for No., Time, Source, Destination, Protocol, and Length. The packet details pane on the right shows the selected packet (No. 1) as an ICMP Echo (ping) request from 192.168.32.100 to 192.168.32.101. The packet bytes pane at the bottom shows the raw data of the selected packet, which is an ICMP Echo request. The packet list shows a sequence of 16 packets, alternating between requests and replies, with the last packet being a request from 192.168.32.100 to 192.168.32.101.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.00000000	192.168.32.100	192.168.32.101	ICMP	96	Echo (ping) request 192.168.32.100 seq=1540956, seq=1540956, ttl=64 (reply in 2)
2	0.00049189	192.168.32.101	192.168.32.100	ICMP	96	Echo (ping) reply 192.168.32.100 seq=1540956, seq=1540956, ttl=128 (request in 1)
3	0.002292475	192.168.32.100	192.168.32.101	ICMP	96	Echo (ping) request 192.168.32.100 seq=1540956, seq=1540956, ttl=64 (reply in 4)
4	0.003254262	192.168.32.101	192.168.32.100	ICMP	96	Echo (ping) reply 192.168.32.100 seq=1540956, seq=1540956, ttl=128 (request in 3)
5	0.004001379	192.168.32.100	192.168.32.101	ICMP	96	Echo (ping) request 192.168.32.100 seq=1540956, seq=1540956, ttl=64 (reply in 6)
6	0.0047102785	192.168.32.101	192.168.32.100	ICMP	96	Echo (ping) reply 192.168.32.100 seq=1540956, seq=1540956, ttl=128 (request in 5)
7	0.005378301	PcsCompu_50-bc:ca	PcsCompu_50-bc:ca	ARP	60	Who has 192.168.32.100? Tell 192.168.32.101
8	0.006383106	PcsCompu_50-bc:ca	PcsCompu_50-bc:ca	ARP	42	192.168.32.100 is at 08:00:27:50:bc:ca
9	0.0072681056	192.168.32.100	192.168.32.101	ICMP	96	Echo (ping) request 192.168.32.100 seq=1540956, seq=1540956, ttl=64 (reply in 10)
10	0.007327412	192.168.32.101	192.168.32.100	ICMP	96	Echo (ping) reply 192.168.32.100 seq=1540956, seq=1540956, ttl=128 (request in 9)
11	0.007485605	192.168.32.100	192.168.32.101	ICMP	96	Echo (ping) request 192.168.32.100 seq=205120, seq=205120, ttl=64 (reply in 12)
12	0.007516072	192.168.32.101	192.168.32.100	ICMP	96	Echo (ping) reply 192.168.32.100 seq=205120, seq=205120, ttl=128 (request in 11)
13	0.006670145	192.168.32.100	192.168.32.101	ICMP	96	Echo (ping) request 192.168.32.100 seq=2173376, seq=2173376, ttl=64 (reply in 14)
14	0.007390811	192.168.32.101	192.168.32.100	ICMP	96	Echo (ping) reply 192.168.32.100 seq=2173376, seq=2173376, ttl=128 (request in 13)
15	0.112260433	192.168.32.100	192.168.32.101	ICMP	96	Echo (ping) request 192.168.32.100 seq=2275822, seq=2275822, ttl=64 (reply in 16)

Frame 1: 96 bytes on wire (768 bits), 96 bytes captured (768 bits) on interface eth0, 0 bytes
 Ethernet II, Src: PcsCompu_50-bc:ca (08:00:27:50:bc:ca), Dst: PcsCompu_50-bc:ca (08:00:27:50:bc:ca)
 Internet Protocol Version 4, Src: 192.168.32.100, Dst: 192.168.32.101
 Internet Control Message Protocol

Configurazione Inetsim



```
christian@kali: ~  
File Actions Edit View Help  
GNU nano 7.2 /etc/inetsim/inetsim.conf  
#  
# Syntax: start_service <service name>  
#  
# Default: none  
#  
# Available service names are:  
# dns, http, smtp, pop3, tftp, ftp, ntp, time_tcp,  
# time_udp, daytime_tcp, daytime_udp, echo_tcp,  
# echo_udp, discard_tcp, discard_udp, quotd_tcp,  
# quotd_udp, chargen_tcp, chargen_udp, finger,  
# 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  
#  
# Help      # Write Out  # Where Is  # Cut       # Execute   # Location  # M-U      # Undo  
# Exit      # Read File  # Replace   # Paste     # Justify   # Go To Line # M-E      # Redo
```

Per far visualizzare il sito “epicode.internal” bisogna settare la inetsim che andrà a simulare i servizi internet all'interno della rete.

Come prima cosa togliamo dai commenti i servizi dns e http, cancellando il “#” prima del comando. Così facendo attiveremo nella inetsim entrambi i servizi. La DNS potrà associare un indirizzo IP ad un dominio alla pagina web, invece l'HTTP potrà far visualizzare il sito in maniera decriptata

```
#####
# dns_static
#
# Static mappings for DNS
#
# Syntax: dns_static <fqdn hostname> <IP address>
#
# Default: none
#
dns_static www.epicode.internal.com 192.168.32.100
#dns_static ns1.foo.com 10.70.50.30
#dns_static ftp.bar.net 10.10.20.30
```

```
#####
# service_bind_address
#
# IP address to bind services to
#
# Syntax: service_bind_address <IP address>
#
# Default: 127.0.0.1
#
service_bind_address 192.168.32.100
```

```
#####
# dns_default_ip
#
# Default IP address to return with DNS replies
#
# Syntax: dns_default_ip <IP address>
#
# Default: 127.0.0.1
#
dns_default_ip 192.168.32.100
```

Subito dopo aver messo i servizi su inetsim, si dovranno cambiare i comandi:

- `dns_static`: mettendo il dominio del sito web che poi andremo a cercare nella macchina client

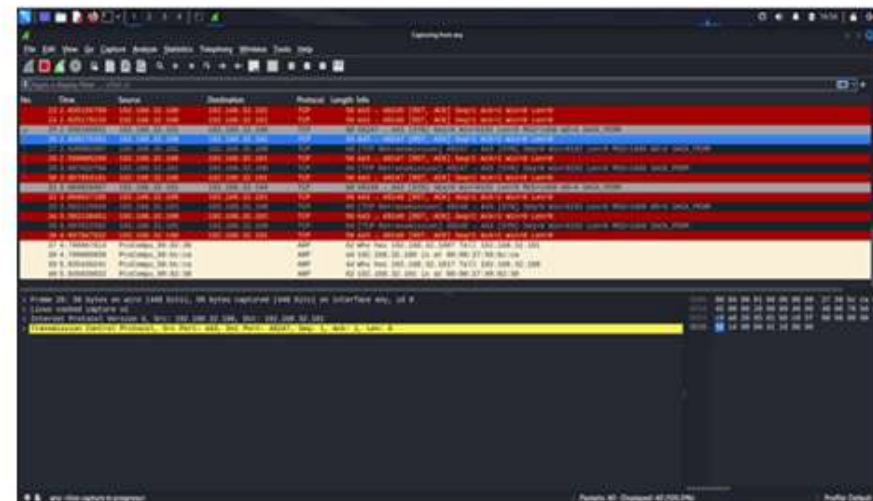
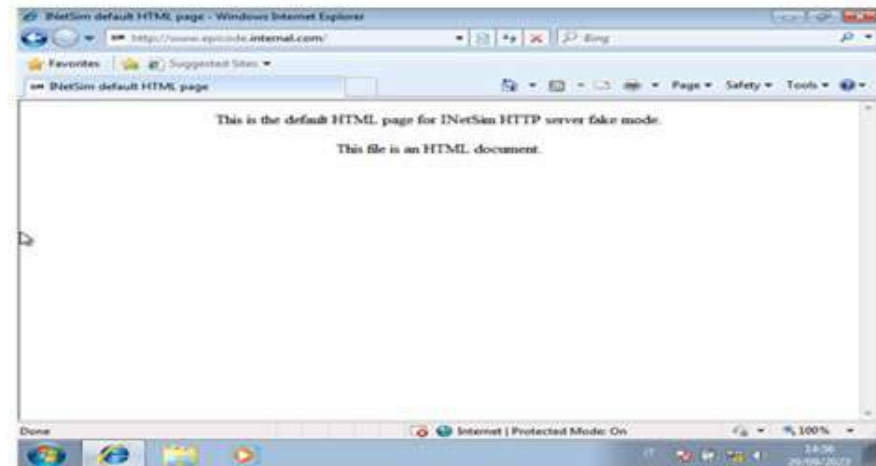
- `service_bind_address`: associerà l'ip all'indirizzo mac

- `dns_default_ip`: mettendo l'indirizzo IP della macchina server

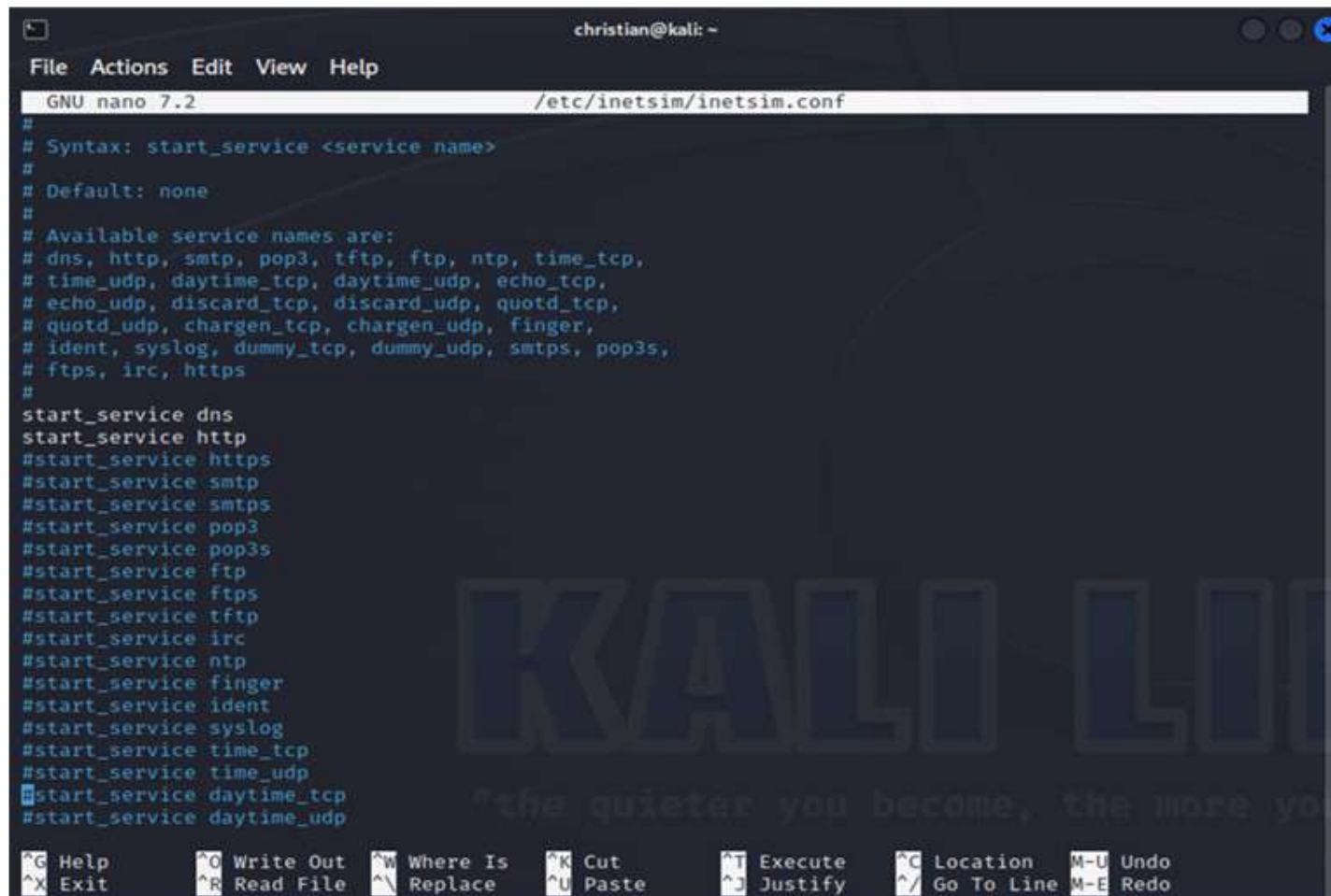
Una volta aver settato tutti i comandi per poter iniziare con la simulazione bisogna avviare sulla macchina **KALI LINUX** inetsim con il comando **sudo inetsim**

```
(christian@kali)-[~]  
$ 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 21342) ==  
Session ID: 21342  
Listening on: 192.168.32.100  
Real Date/Time: 2023-09-29 15:08:25  
Fake Date/Time: 2023-09-29 15:08:25 (Delta: 0 seconds)  
Forking services ...  
* dns_53_tcp_udp - started (PID 21352)  
print() on closed filehandle MLOG at /usr/share/perl5/Net/DNS/Nameserver.pm line 399.  
print() on closed filehandle MLOG at /usr/share/perl5/Net/DNS/Nameserver.pm line 399.  
* https_443_tcp - started (PID 21353)  
done.  
Simulation running.  
█
```


Dopo aver avviato il comando su **KALI LINUX**, accedere alla macchina client e aprendo il browser bisogna cercare sulla barra di ricerca <http://www.epicode.internal.com> facendo lo sniff con wireshark vedremo le reti decriptate



Disattivare i servizi di HTTPS e attivare quelli di HTTP nei comandi di inetsim



The image shows a terminal window with the nano text editor open, editing the file /etc/inetsim/inetsim.conf. The window title is 'christian@kali: ~'. The editor's status bar at the top shows 'GNU nano 7.2' and the file path. The configuration file content is as follows:

```
#  
# Syntax: start_service <service name>  
#  
# Default: none  
#  
# Available service names are:  
# dns, http, smtp, pop3, tftp, ftp, ntp, time_tcp,  
# time_udp, daytime_tcp, daytime_udp, echo_tcp,  
# echo_udp, discard_tcp, discard_udp, quotd_tcp,  
# quotd_udp, chargen_tcp, chargen_udp, finger,  
# 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
```

The bottom of the window displays a row of keyboard shortcuts: ^G Help, ^O Write Out, ^W Where Is, ^K Cut, ^T Execute, ^C Location, M-U Undo, ^X Exit, ^R Read File, ^_ Replace, ^U Paste, ^J Justify, ^/ Go To Line, M-E Redo.

Far partire inetsim sempre con il comando **sudo inetsim** e aprire il client e cercare sulla barra URL del browser <https://www.epicode.internal>.
Effettuando lo sniff con wire shark si noterà che i pacchetti di mandati sono criptati

