

## PROGETTO S1/L5

### CATTURARE IL TRAFFICO HTTP E HTTPS CON WIRESHARK

Su Kali, attraverso il comando `sudo nano /etc/network/interfaces`, ho aggiunto le righe di codice “inet static” che mi hanno permesso di impostare in modo statico l’ip dell’interfaccia “eth0”. Successivamente ho assegnato, manualmente, l’ip(192.168.32.100), subnetmask(255.255.255.0 equivalente a /24), ip di network(192.168.32.0), broadcast(192.168.32.255) e gateway(192.168.32.1).

The image shows two terminal windows side-by-side on a Kali Linux desktop environment.

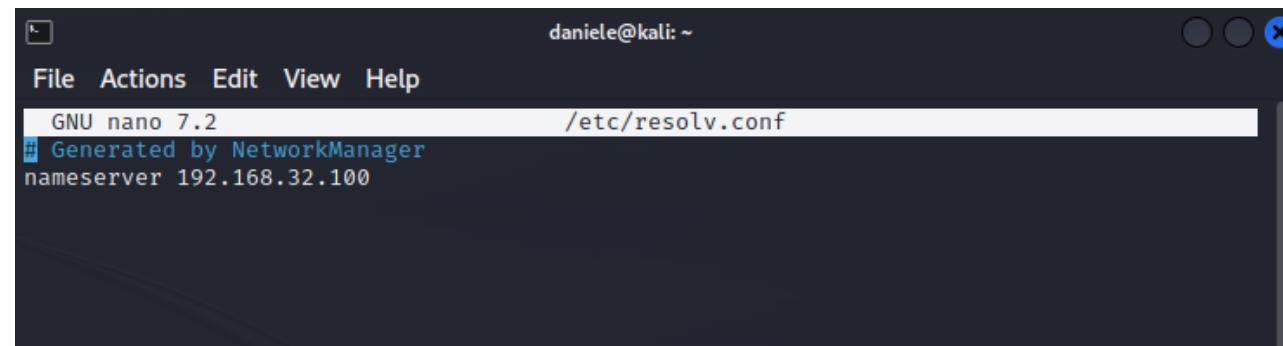
**Terminal Window 1 (Left):**

```
daniele@kali:~  
File Actions Edit View Help  
Report written to '/var/log/inetsim/report/report.22280.txt' (81 lines)  
== INetSim main process stopped (PID 22280) ==  
. .  
(daniele@kali)-[~]  
$ ifconfig  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.32.100 netmask 255.255.255.0 broadcast 192.168.32.255  
        inet6 fe80::51ba:d214:ffae:6e49 prefixlen 64 scopeid 0x20<link>  
        ether 08:00:27:53:43:47 txqueuelen 1000 (Ethernet)  
          RX packets 2473 bytes 252199 (246.2 KiB)  
          RX errors 0 dropped 0 overruns 0 frame 0  
          TX packets 2049 bytes 350919 (342.6 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 0x10<host>  
    loop txqueuelen 1000 (Local Loopback)  
      RX packets 1353 bytes 132058 (128.9 KiB)  
      RX errors 0 dropped 0 overruns 0 frame 0  
      TX packets 1353 bytes 132058 (128.9 KiB)  
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
(daniele@kali)-[~]  
$
```

**Terminal Window 2 (Right):**

```
daniele@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(5).  
  
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  
netmask 255.255.255.0  
network 192.168.32.0  
broadcast 192.168.32.255  
gateway 192.168.32.1  
  
[ Wrote 16 lines ]  
^G Help ^O Write Out ^W Where Is ^K Cut  
^X Exit ^R Read File ^\ Replace ^U Paste  
^T Execute  
^J Justify
```

Con il comando sudo nano /etc/resolv.conf, ho modificato il server DNS. Affinché l'ip corrisponda alla macchina che eroga il servizio DNS, ovvero kali stesso.

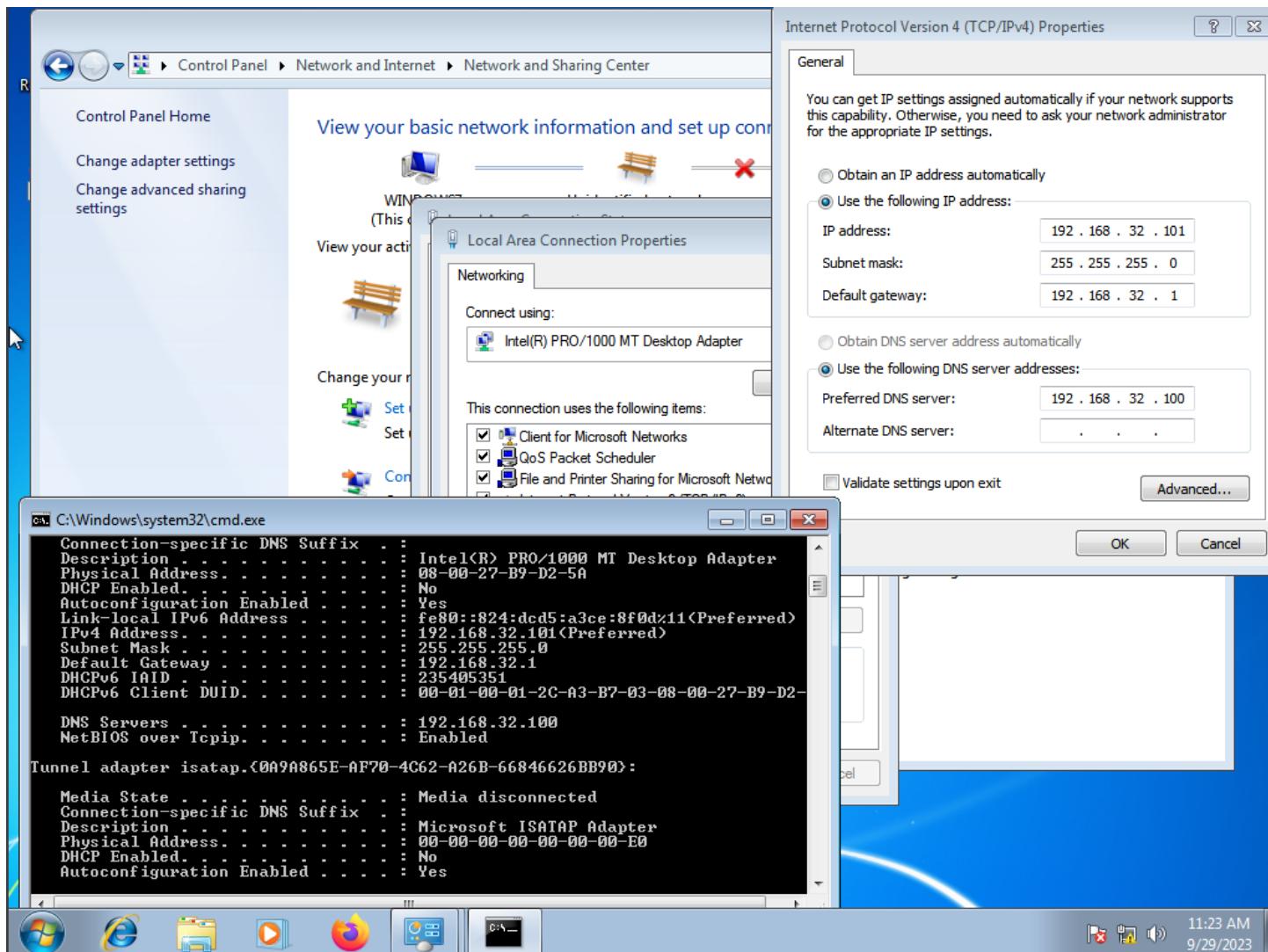


A screenshot of a terminal window titled "daniele@kali: ~". The window title bar also shows "File Actions Edit View Help" and the file path "/etc/resolv.conf". The terminal content is as follows:

```
GNU nano 7.2          /etc/resolv.conf
# Generated by NetworkManager
nameserver 192.168.32.100
```

Su Windows 7, ho aperto le impostazioni della scheda di rete ed ho modificato le impostazioni riguardanti l'ipv4.

Ho assegnato alla macchina l'ip 192.168.32.101, subnetmask 255.255.255.0, default gateway 192.168.32.1 e server DNS 192.168.32.100(Macchina Virtuale Kali).



Su Kali, attraverso l'utilizzo di inetsim, che è un ONEPOT che simula i servizi, con il comando sudo nano /etc/inetsim/inetsim.conf, ho abilitato il servizio dns e https, togliendo “#” prima del comando. L'Hashtag viene utilizzato per rendere un commento una riga di codice.

Per il primo test del traffico con l'utilizzo dell'https. Per il secondo test, invece, ho disabilitato l'https ed ho abilitato l'http.

```
File Actions Edit View Help
GNU nano 7.2          /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, tftp, ftp, ntp, time_tcp,
# time_udp, daytime_tcp, daytime_udp, echo_tcp,
# echo_udp, discard_tcp, discard_udp, quod_tcp,
# quod_udp, chargen_tcp, chargen_udp, finger,
# ident, syslog, dummy_tcp, dummy_udp, smtsp, pop3s,
# ftps, irc, https
#
start_service dns
start_service http
start_service https
start_service smtp
start_service smtsp
start_service pop3
start_service pop3s
start_service ftp
start_service ftsp
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 quod_tcp
start_service quod_udp
start_service chargen_tcp
start_service chargen_udp
start_service dummy_tcp
start_service dummy_udp
start_service dummy_udp

File Actions Edit View Help
GNU nano 7.2          /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, tftp, ftp, ntp, time_tcp,
# time_udp, daytime_tcp, daytime_udp, echo_tcp,
# echo_udp, discard_tcp, discard_udp, quod_tcp,
# quod_udp, chargen_tcp, chargen_udp, finger,
# ident, syslog, dummy_tcp, dummy_udp, smtsp, pop3s,
# ftps, irc, https
#
start_service dns
start_service http
#start_service https
start_service smtp
start_service smtsp
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 quod_tcp
start_service quod_udp
start_service chargen_tcp
start_service chargen_udp
start_service dummy_tcp
start_service dummy_udp
start_service dummy_udp
```

Sempre nella stessa pagina di configurazione, ma più in basso, ho impostato che i vari servizi vengano erogati dall'ip della macchina, quindi 192.168.32.100.

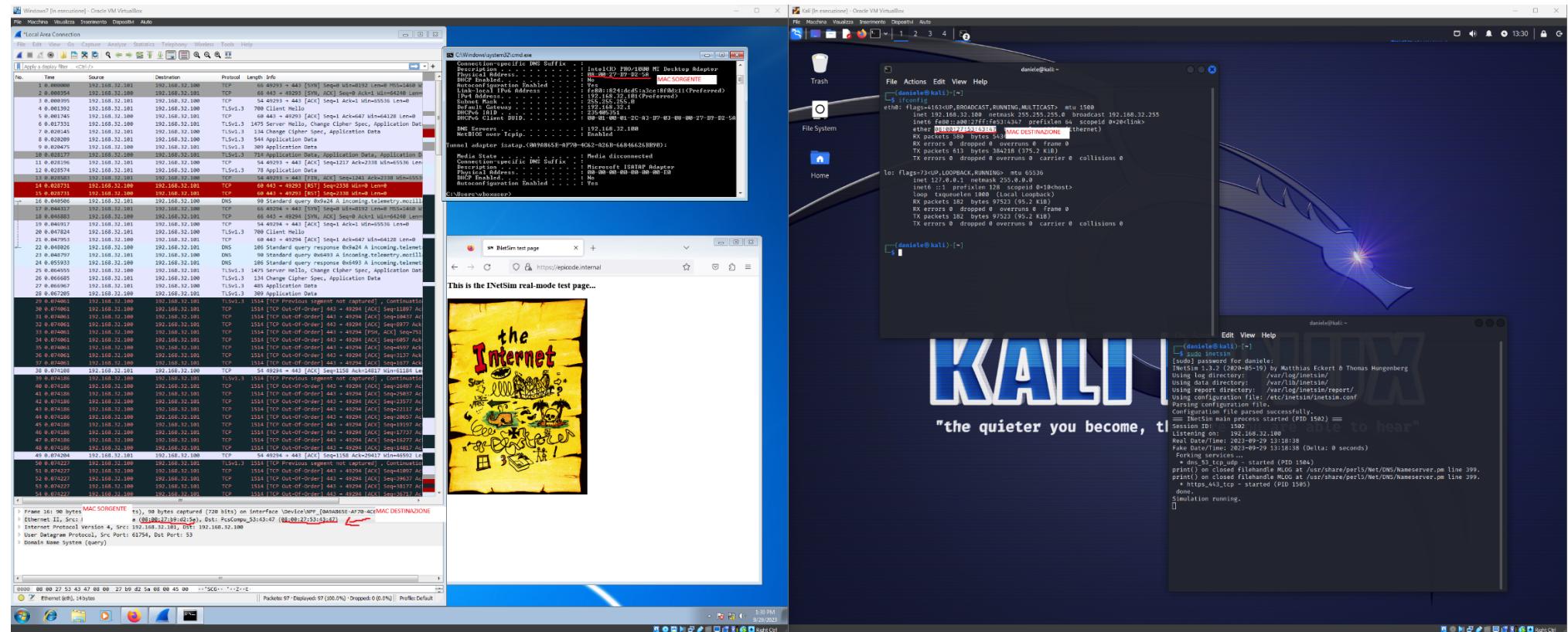
```
#####
# service_bind_address
#
# IP address to bind services to
#
# Syntax: service_bind_address <IP address>
#
# Default: 127.0.0.1
#
service_bind_address 192.168.32.100
```

Ho configurato il servizio DNS, affinché il dominio “epicode.internal”, venga tradotto in indirizzo ip 192.168.32.100

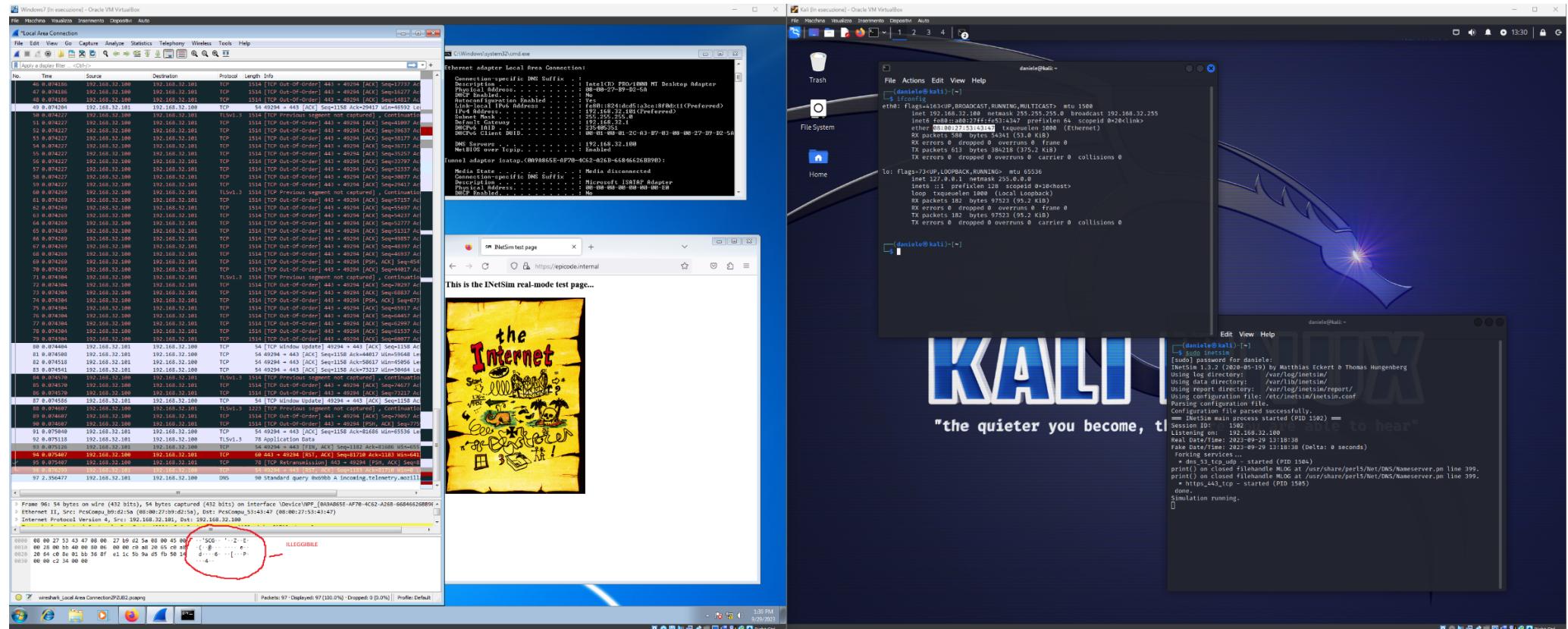
```
#####
#dns_static
#
# Static mappings for DNS
#
# Syntax: dns_static <fqdn hostname> <IP address>
#
# Default: none
#
#dns_static www.foo.com 10.10.10.10
#dns_static ns1.foo.com 10.70.50.30
#dns_static ftp.bar.net 10.10.20.30
dns_static epicode.internal 192.168.32.100
```

Di seguito, ho attivato l'emulazione dei servizi di inetsim, utilizzando il comando sudo inetsim, per avviare il servizio.

Ho catturato i pacchetti attraverso l'utilizzo dell'applicativo "wireshark", mettendo in risalto gli indirizzi MAC delle due macchine, ed ho aperto la pagina web digitando sulla barra degli indirizzi "epicode.internal", e non l'ip del server https, mostrando quindi, che il servizio DNS è perfettamente funzionante.



L'HTTPS utilizza la crittografia, TLS in questo caso, pertanto non è possibile leggere il messaggio prima che arrivi alla destinazione, senza avere la chiave di cifratura.



Ho ripetuto il tutto utilizzando il servizio HTTP, mostrando che, mediante l'utilizzo di questo protocollo, non vi è alcuna cifratur, pertanto, è possibile conoscere il contenuto del pacchetto, prima che arrivi alla macchina di destinazione.

