

1 INDICE

1 INDICE	1
TRACCIA.....	2
1.1 IMPOSTARE GLI INDIRIZZI IP	2
1.2 DISABILITARE IL FIREWALL SU WINDOWS	3
1.3 ESEGUIRE IL PING.....	3
1.4 CONFIGURAZIONE DEL SERVER DNS.....	4
1.5 INSTALLAZIONE DNMASQ	5
1.6 CONFIGURAZIONE DNMASQ	6
1.7 CONFIGURAZIONE INETSIM	8
1.8 RICERCA WEB BROWSER.....	9
1.9 SNIFFING CON WIRESHARK.....	10

TRACCIA: Simulare, usando il laboratorio virtuale, un’architettura client server in cui un client con indirizzo 192.168.32.101 (Windows) richiede tramite Web Browser una risorsa all’hostname “**epicode.internal**” che risponde all’indirizzo 192.168.32.100 (Kali Linux).

- Intercettare la comunicazione usando **WireShark**, individuando il **MAC Address** di sorgente e destinazione e il contenuto della richiesta **HTTPS**.
 - Sostituire il server **HTTPS** con server **HTTP**, intercettare nuovamente il traffico ed evidenziare le differenze con lo sniffing precedente.

1.1 IMPOSTARE GLI INDIRIZZI IP

- Impostare gli indirizzi IP di nostro interesse, prima su Kali Linux inserendo nel terminale il comando:
 - `sudo nano /etc/network/interfaces`
 - Compilare i campi come riportato di seguito nella figura 1

```
[kali㉿kali: ~]
File Actions Edit View Help
GNU nano 8.3                                     /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/*                [Mon Jul 30 14:42:35 2018] UTC
# The loopback network interface
auto lo                                         [Mon Jul 30 14:42:35 2018] seconds
#iface lo inet loopback                         [Mon Jul 30 14:42:35 2018] seconds
auto eth0                                       [Mon Jul 30 14:42:35 2018] seconds
iface eth0 inet static                          [Mon Jul 30 14:42:35 2018] seconds
address 192.168.32.100/24                      [Mon Jul 30 14:42:35 2018] seconds
gateway 192.168.32.1                           [Mon Jul 30 14:42:35 2018] seconds
[Mon Jul 30 14:42:35 2018] seconds]
```

Figura 1 Configurazione indirizzo IP su Kali Linux

- Successivamente si effettuerà lo stesso passaggio su Windows 7 entrando su "**centro connessioni di rete e condivisione**" > **modifica impostazioni scheda > connessione alla rete locale LAN** > premere su "**proprietà**" e selezionare **IPv4**, procedendo alla configurazione dell'IP come in foto, facendo attenzione a compilare anche il campo relativo al DNS.

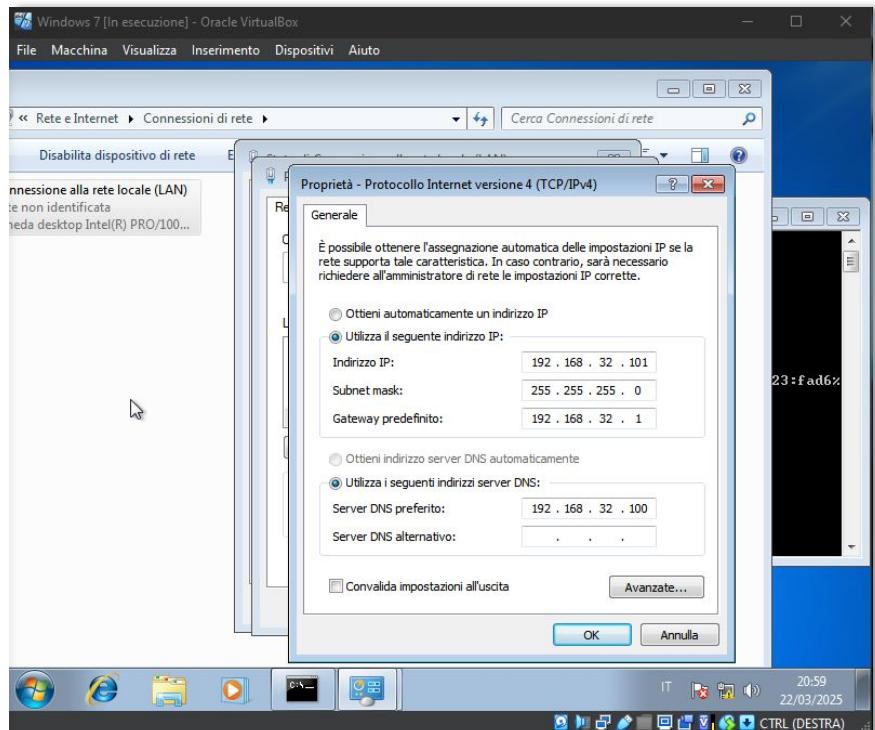


Figura 2 Configurazione Indirizzo IP su Windows 7

1.2 DISABILITARE IL FIREWALL SU WINDOWS

Per capire se le due macchine comunicano, eseguiamo un semplice “ping” per entrambe. Fondamentale disabilitare il Firewall su Windows 7 per permettere la corretta ricezione dei pacchetti

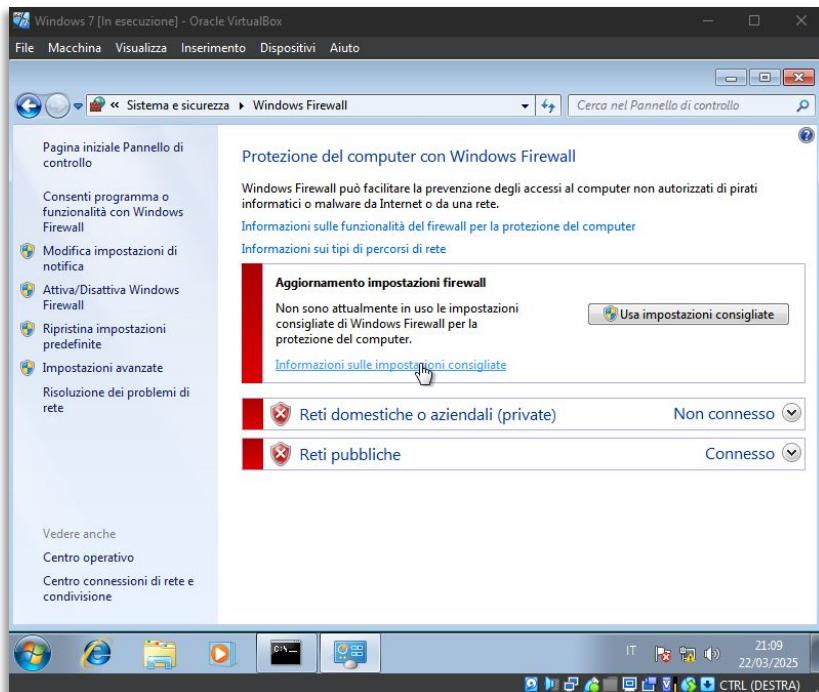


Figura 3 Firewall disabilitato su Windows 7

1.3 ESEGUIRE IL PING

Per effettuare il ping su Kali Linux, basterà digitare sul terminale il comando:

- **Ping 192.168.32.101**
- Noteremo subito l'invio di numerosi pacchetti verso la macchina Windows che risponderà perfettamente

A screenshot of a terminal window on Kali Linux. The terminal shows the output of the 'ifconfig' command, which lists network interfaces eth0 and eth1. It then shows the execution of the 'ping' command to the IP address 192.168.32.101. The terminal output includes numerous ICMP echo requests sent to the target host, with responses from the Windows 7 machine.

Figura 4 Ping eseguito da Kali Linux all'indirizzo 192.168.32.101 di Windows 7

Su Windows 7 sarà necessario:

- Aprire il Prompt dei comandi;
 - Scrivere il comando `Ping 192.168.32.100`

```
cmd. Prompt dei comandi
Risposta da 192.168.32.100: byte=32 durata<1ms TTL=64
Risposta da 192.168.32.100: byte=32 durata<1ms TTL=64
Risposta da 192.168.32.100: byte=32 durata<1ms TTL=64

Statistiche Ping per 192.168.32.100:
  Pacchetti: Trasmessi = 4, Ricevuti = 4,
  Persi = 0 <0% persi>.
Tempo approssimativo percorsi andata/ritorno in millisecondi:
  Minimo = 0ms, Massimo = 0ms, Medio = 0ms

C:\Users\Win7>ping 192.168.32.100

Esecuzione di Ping 192.168.32.100 con 32 byte di dati:
Risposta da 192.168.32.100: byte=32 durata<1ms TTL=64

Statistiche Ping per 192.168.32.100:
  Pacchetti: Trasmessi = 4, Ricevuti = 4,
  Persi = 0 <0% persi>.
Tempo approssimativo percorsi andata/ritorno in millisecondi:
  Minimo = 0ms, Massimo = 0ms, Medio = 0ms

C:\Users\Win7>ping_
```

Figura 5 Ping eseguito da Windows 7 all'indirizzo 192.168.32.100 su Kali Linux

1.4 CONFIGURAZIONE DEL SERVER DNS

Ora che abbiamo avuto conferma della corretta comunicazione tra le due VM, possiamo procedere con la fase successiva, ovvero la **configurazione del server DNS**. Apriamo quindi il servizio **INetSim** digitando sul terminale di Kali il comando:

- `sudo inetsim`, il quale esito sarà il seguente:

```
(kali㉿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 20002) ==
Session ID: 20002
Listening on: 192.168.32.100
Real Date/Time: 2025-03-22 17:25:25
Fake Date/Time: 2025-03-22 17:25:25 (Delta: 0 seconds)
Forking services...
* dns_53_tcp_udp - started (PID 20004)
Can't locate object method "main_loop" via package "Net::DNS::Nameserver" at /usr/share/perl5/INetSim/DNS.pm line 69.
* irc_6667_tcp - started (PID 20014)
* finger_79_tcp - started (PID 20016)
* ntp_123_udp - started (PID 20015)
* daytime_13_tcp - started (PID 20021)
* https_443_tcp - started (PID 20006)
* smtp_25_tcp - started (PID 20007)
* pop3_110_tcp - started (PID 20009)
* time_37_tcp - started (PID 20019)
* pop3s_995_tcp - started (PID 20010)
* syslog_514_udp - started (PID 20018)
* daytime_13_udp - started (PID 20022)
* echo_7_udp - started (PID 20024)
* smtps_465_tcp - started (PID 20008)
* discard_9_udp - started (PID 20026)
* quotd_17_tcp - started (PID 20027)
* discard_9_tcp - started (PID 20025)
* ident_113_tcp - started (PID 20017)
* tftp_69_udp - started (PID 20013)
* http_80_tcp - started (PID 20005)
* time_37_tcp - started (PID 20020)
* tftp_23_tcp - started (PID 20011)
* ftpt_990_tcp - started (PID 20012)
* echo_7_tcp - started (PID 20023)
* quotd_17_udp - started (PID 20028)
* chargen_19_udp - started (PID 20030)
* chargen_19_tcp - started (PID 20029)
* dummy_1_tcp - started (PID 20031)
* dummy_1_udp - started (PID 20032)
done.
Simulation running.
```

Figura 6 Esito del comando "sudo inetsim" su terminale Kali Linux

Da qui possiamo notare come la **porta 53** del DNS non sia funzionante. Facendo delle ricerche, si è saputo che questa versione in particolare presenta dei bug proprio relativi a questa porta, che diventa quindi inutilizzabile.

A questo punto che fare?

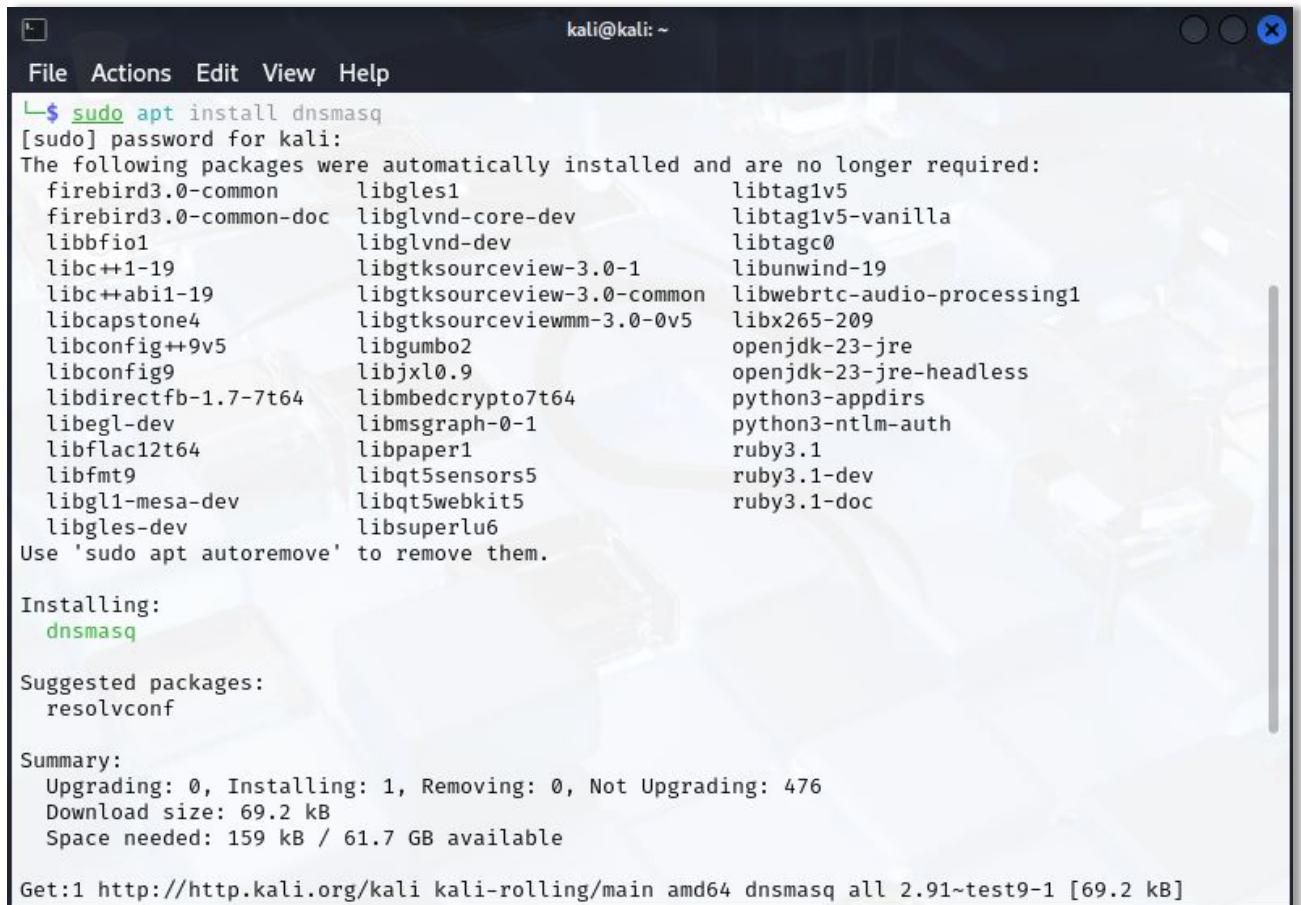
La soluzione alternativa scelta per proseguire con l'esercizio, è stata quella di appellarsi al servizio “**dnsmasq**”, che non essendo pre-installato sul sistema, è stato necessario effettuare il download dalla **repository ufficiale**.

Essendo la macchina impostata su “**Rete Interna**”, è stato necessario settare momentaneamente l'accesso alla rete esterna per il download del pacchetto. Successivamente è stata ristabilita la rete isolata.

1.5 INSTALLAZIONE DNSMASQ

A questo punto, si può procedere con l'installazione del servizio. Digitare su terminale il comando:

- `sudo apt install dnsmasq`
- Kali chiederà di inserire la password, quindi digitare “kali” e premere “invio”.
- Il sistema procederà con il download e l'installazione del pacchetto.



The screenshot shows a terminal window titled "kali@kali: ~". The command `sudo apt install dnsmasq` is being run. The terminal output shows the following:

```
kali@kali: ~
File Actions Edit View Help
└$ sudo apt install dnsmasq
[sudo] password for kali:
The following packages were automatically installed and are no longer required:
  firebird3.0-common   libgles1           libtag1v5
  firebird3.0-common-doc libglvnd-core-dev libtag1v5-vanilla
  libbfio1             libglvnd-dev       libtagc0
  libc++1-19            libgtksourceview-3.0-1 libunwind-19
  libc++abi1-19         libgtksourceview-3.0-common libwebrtc-audio-processing1
  libcapstone4          libgtksourceviewmm-3.0-0v5 libx265-209
  libconfig++9v5        libgumbo2          openjdk-23-jre
  libconfig9            libjxl0.9          openjdk-23-jre-headless
  libdirectfb-1.7-7t64  libmbcrypto7t64    python3-appdirs
  libegl-dev            libmsgraph-0-1     ruby3.1
  libflac12t64          libpaper1          ruby3.1-dev
  libfmt9               libqt5sensor5    ruby3.1-doc
  libgl1-mesa-dev       libqt5webkit5   Use 'sudo apt autoremove' to remove them.

Installing:
  dnsmasq

Suggested packages:
  resolvconf

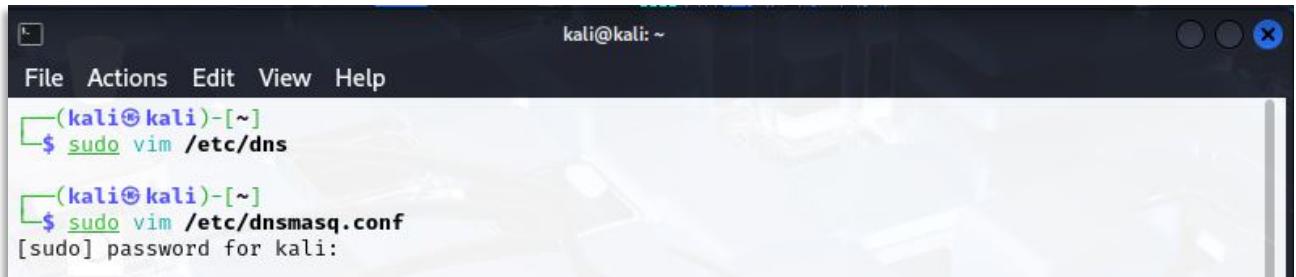
Summary:
  Upgrading: 0, Installing: 1, Removing: 0, Not Upgrading: 476
  Download size: 69.2 kB
  Space needed: 159 kB / 61.7 GB available

Get:1 http://http.kali.org/kali kali-rolling/main amd64 dnsmasq all 2.91~test9-1 [69.2 kB]
```

Figura 7 Installazione dnsmasq da repository ufficiale

Una volta ottenuto il pacchetto, è importantissima la configurazione del servizio tramite due step cruciali, indispensabili per il corretto svolgimento della prova. Per procedere, scrivere su terminale il comando:

- `sudo vim /etc/dnsmasq.conf`

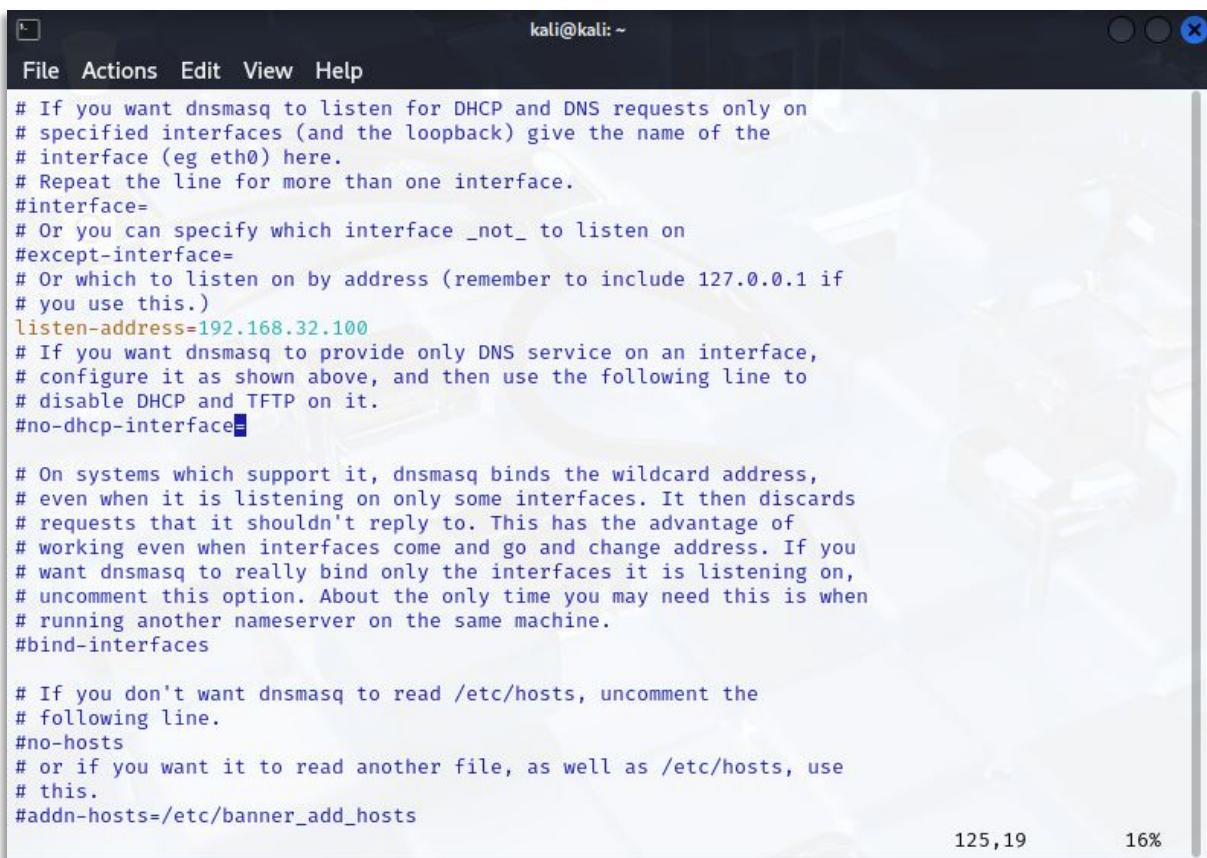


A screenshot of a terminal window titled "kali@kali: ~". The window shows a command history:
1. `(kali㉿kali)-[~]`
2. `$ sudo vim /etc/dns`
3. `(kali㉿kali)-[~]`
4. `$ sudo vim /etc/dnsmasq.conf`
A password prompt "[sudo] password for kali:" is visible at the bottom.

1.6 CONFIGURAZIONE DNSMASQ

Dal comando inviato precedentemente si aprirà il file di configurazione in cui bisognerà modificare dei parametri:

- **FASE 1 DNSMASQ** = impostare l'IP Address che ci interessa inserendo alla riga 121 il comando `listen-address=192.168.32.100`. In questo modo si mette in ascolto il servizio sull'indirizzo IP assegnato all'interfaccia `eth0` (come mostrato nella fig. 1).



A screenshot of a terminal window titled "kali@kali: ~" showing the contents of the `/etc/dnsmasq.conf` file in Vim. The file contains several comments and configuration options. The `listen-address=192.168.32.100` line is highlighted in blue, indicating it has been modified. Other lines include:
If you want dnsmasq to listen for DHCP and DNS requests only on
specified interfaces (and the loopback) give the name of the
interface (eg eth0) here.
Repeat the line for more than one interface.
`#interface=`
Or you can specify which interface _not_ to listen on
`#except-interface=`
Or which to listen on by address (remember to include 127.0.0.1 if
you use this.)
`listen-address=192.168.32.100`
If you want dnsmasq to provide only DNS service on an interface,
configure it as shown above, and then use the following line to
disable DHCP and TFTP on it.
`#no-dhcp-interface=`

On systems which support it, dnsmasq binds the wildcard address,
even when it is listening on only some interfaces. It then discards
requests that it shouldn't reply to. This has the advantage of
working even when interfaces come and go and change address. If you
want dnsmasq to really bind only the interfaces it is listening on,
uncomment this option. About the only time you may need this is when
running another nameserver on the same machine.
`#bind-interfaces`

If you don't want dnsmasq to read /etc/hosts, uncomment the
following line.
`#no-hosts`
or if you want it to read another file, as well as /etc/hosts, use
this.
`#addn-hosts=/etc/banner_add_hosts`

- **FASE 2 DNSMASQ** = Inserire il dominio tramite il comando:
- `address=/epicode.internal/192.168.32.100`. Con la seguente configurazione si assegna questo indirizzo a “**epicode.internal**”.

```
# from /etc/hosts or DHCP only.
#local=/localnet/
#
# Add domains which you want to force to an IP address here.
# The example below send any host in double-click.net to a local
# web-server.
#address=/double-click.net/127.0.0.1
address=/epicode.internal/192.168.32.100
# --address (and --server) work with IPv6 addresses too.
#address=/www.thekelleys.org.uk/fe80::20d:60ff:fe36:f83
#
# Add the IPs of all queries to yahoo.com, google.com, and their
# subdomains to the vpn and search ipsets:
#ipset=/yahoo.com/google.com/vpn,search
#
# Add the IPs of all queries to yahoo.com, google.com, and their
# subdomains to netfilters sets, which is equivalent to
# 'nft add element ip test vpn { ... }; nft add element ip test search { ... }'
#nftset=/yahoo.com/google.com/ip#test#vpn,ip#test#search
#
# Use netfilters sets for both IPv4 and IPv6:
# This adds all addresses in *.yahoo.com to vpn4 and vpn6 for IPv4 and IPv6 addresses.
#nftset=/yahoo.com/4#ip#test#vpn4
#nftset=/yahoo.com/6#ip#test#vpn6
#
# You can control how dnsmasq talks to a server: this forces
# queries to 10.1.2.3 to be routed via eth1
# server=10.1.2.3@eth1
#
# and this sets the source (ie local) address used to talk to
-- INSERT --
80,41 10%
```

Dopo aver effettuato questi brevi passaggi, è necessario abilitare e avviare il servizio **dnsmasq** con il seguente comando:

- `sudo systemctl enable --now dnsmasq`

Successivamente per controllare lo stato running del servizio bisogna inviare:

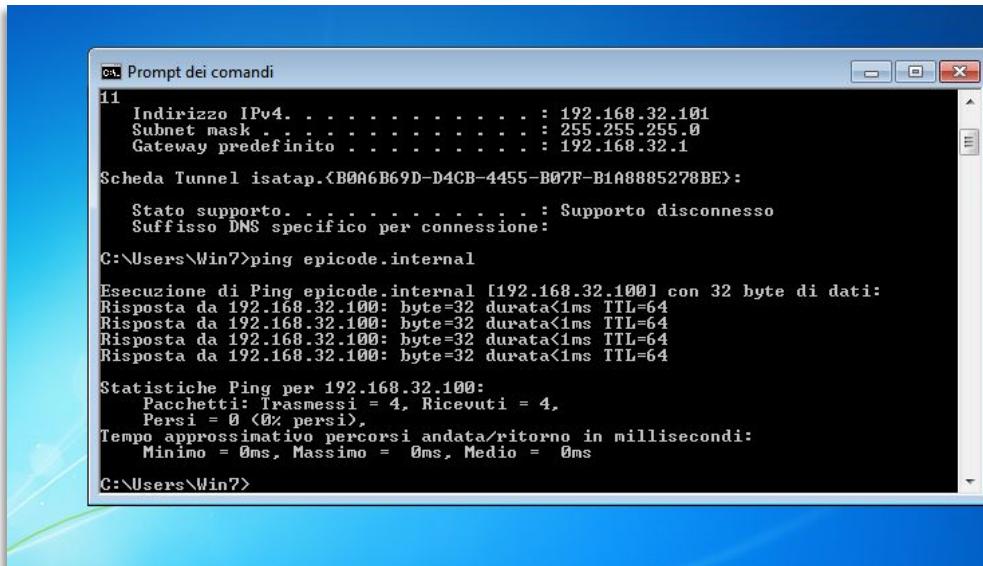
- `sudo systemctl status dnsmasq`

```
(kali㉿kali)-[~]
$ sudo systemctl status dnsmasq
● dnsmasq.service - A lightweight DHCP and caching DNS server
   Loaded: loaded (/usr/lib/systemd/system/dnsmasq.service; enabled; preset: disabled)
   Active: active (running) since Sat 2025-03-22 17:21:02 EDT; 27s ago
     Docs: man:dnsmasq(8)
   Process: 17882 ExecStartPre=/usr/share/dnsmasq/systemd-helper checkconfig (code=exited, st>
   Process: 17889 ExecStart=/usr/share/dnsmasq/systemd-helper exec (code=exited, status=0/SUC>
   Process: 17895 ExecStartPost=/usr/share/dnsmasq/systemd-helper start-resolvconf (code=exit>
 Main PID: 17894 (dnsmasq)
   Docs: man:dnsmasq(8)
 Tasks: 1 (limit: 9379)
  Memory: 2.3M (peak: 4.1M)
    CPU: 31ms
   CGroup: /system.slice/dnsmasq.service
           └─17894 /usr/sbin/dnsmasq -x /run/dnsmasq/dnsmasq.pid -u dnsmasq -7 /etc/dnsmasq.2

Mar 22 17:21:02 kali systemd[1]: Starting dnsmasq.service - dnsmasq - A lightweight DHCP and c
Mar 22 17:21:02 kali dnsmasq[17894]: started, version 2.91test9 cachesize 150
Mar 22 17:21:02 kali dnsmasq[17894]: compile time options: IPv6 GNU-getopt DBus no-UBus i18n I
Mar 22 17:21:02 kali dnsmasq[17894]: reading /etc/resolv.conf
Mar 22 17:21:02 kali dnsmasq[17894]: using nameserver 1.1.1.1#53
Mar 22 17:21:02 kali dnsmasq[17894]: using nameserver 8.8.8.8#53
Mar 22 17:21:02 kali dnsmasq[17894]: read /etc/hosts - 7 names
Mar 22 17:21:02 kali systemd[1]: Started dnsmasq.service - dnsmasq - A lightweight DHCP and ca
```

Lo step successivo prevede un controllo per il corretto funzionamento del server DNS monitorando dal **cmd Windows** il **resolve di episode.internal** digitando:

- Ping episode.internal



```

cmd Prompt dei comandi
11
Indirizzo IPv4 . . . . . : 192.168.32.101
Subnet mask . . . . . : 255.255.255.0
Gateway predefinito . . . . . : 192.168.32.1

Scheda Tunnel isatap.{B0A6B69D-D4CB-4455-B07F-B1A8885278BE}:
    Stato supporto . . . . . : Supporto disconnesso
    Suffixo DNS specifico per connessione: 

C:\Users\Win7>ping episode.internal

Esecuzione di Ping episode.internal [192.168.32.100] con 32 byte di dati:
Risposta da 192.168.32.100: byte=32 durata<1ms TTL=64

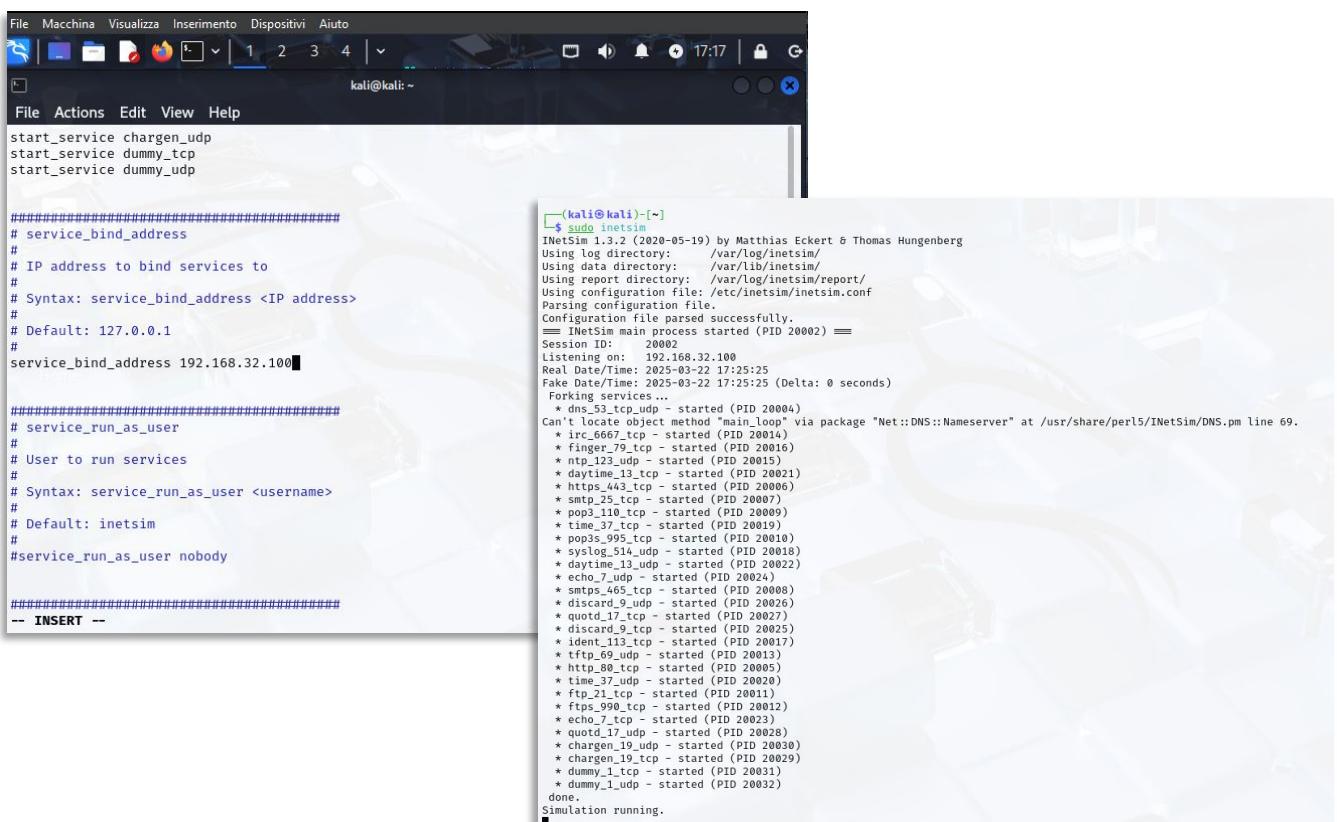
Statistiche Ping per 192.168.32.100:
    Pacchetti: Trasmessi = 4, Ricevuti = 4,
    Persi = 0 <0% persi>
    Tempo approssimativo percorsi andata/ritorno in millisecondi:
        Minimo = 0ms, Massimo = 0ms, Medio = 0ms
C:\Users\Win7>

```

Figura 8 Ping di episode.internal su terminale Windows 7

1.7 CONFIGURAZIONE INETSIM

A questo punto è apparato che il server DNS funziona nel modo giusto, quindi si può procedere con la configurazione del servizio **INetSim**. Come da figura sarà necessario modificare il campo **“service_bind_address”** inserendo l’IP di nostro interesse, ovvero **192.168.32.100**. In questo modo tutti i servizi che effettueranno l’avvio, risulteranno in ascolto sull’IP impostato. Dopo ciò, avviare INetSim tramite comando **sudo inetsim**.



```

File Macchina Visualizza Inserimento Dispositivi Aiuto
File Actions Edit View Help
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 192.168.32.100

#####
# service_run_as_user
# User to run services
# Syntax: service_run_as_user <username>
# Default: inetsim
# #service_run_as_user nobody

#####
-- INSERT --

```

```

(kali㉿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 20002) ==
Session ID: 20002
Listening on: 192.168.32.100
Real Date/Time: 2025-03-22 17:25:25
Fake Date/Time: 2025-03-22 17:25:25 (Delta: 0 seconds)
Forking services...
* dns_53_tcp_udp - started (PID 20004)
Can't locate method "main_loop" via package "Net::DNS::Nameserver" at /usr/share/perl5/INetSim/DNS.pm line 69.
* http_6661_tcp - started (PID 20005)
* finger_70_tcp - started (PID 20016)
* ntp_123_udp - started (PID 20015)
* daytime_13_udp - started (PID 20021)
* https_443_tcp - started (PID 20006)
* smtp_25_tcp - started (PID 20007)
* pop3_110_tcp - started (PID 20009)
* time_37_tcp - started (PID 20019)
* pop3s_995_tcp - started (PID 20010)
* syslog_514_udp - started (PID 20018)
* daytime_13_udp - started (PID 20022)
* echo_7_udp - started (PID 20024)
* smtps_465_tcp - started (PID 20008)
* discard_9_udp - started (PID 20026)
* whois_47_udp - started (PID 20017)
* discard_9_tcp - started (PID 20025)
* ident_113_tcp - started (PID 20017)
* tftp_69_udp - started (PID 20013)
* http_80_tcp - started (PID 20005)
* time_37_udp - started (PID 20020)
* ftp_21_tcp - started (PID 20011)
* ftps_990_tcp - started (PID 20012)
* echo_7_tcp - started (PID 20023)
* quotd_17_udp - started (PID 20028)
* chargen_19_udp - started (PID 20030)
* chargen_19_tcp - started (PID 20029)
* dummy_1_tcp - started (PID 20031)
* dummy_1_udp - started (PID 20032)
done.
Simulation running.

```

1.8 RICERCA WEB BROWSER

A questo punto, si può procedere con la ricerca su Browser di “epicode.internal” sia con **HTTP** che **HTTPS**. Noteremo subito una differenza sostanziale:

- HTTP ricercherà semplicemente ciò che abbiamo richiesto
- HTTPS tenterà di bloccarci la navigazione segnalando la **mancanza di un certificato valido**.

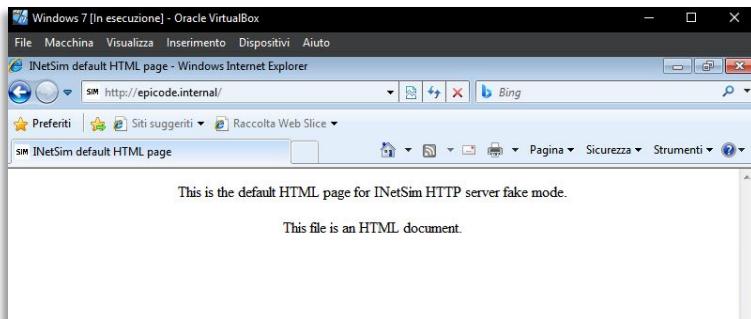


Figura 10 Ricerca di epicode.internal con HTTP

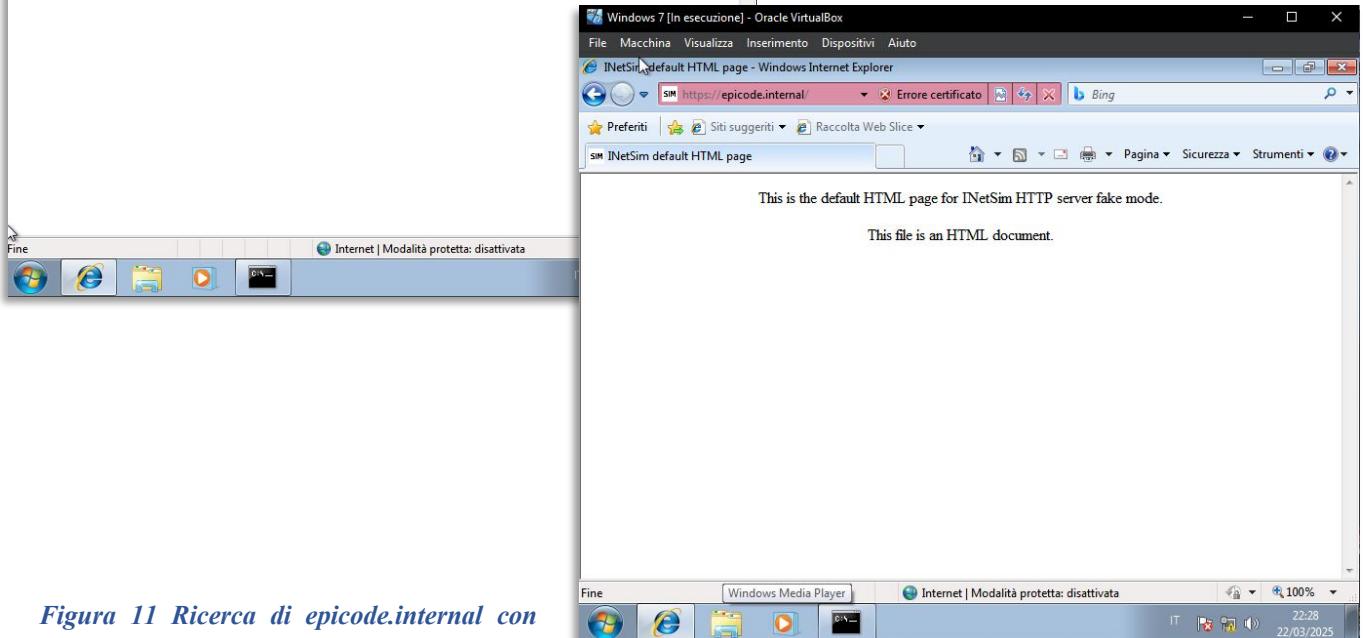


Figura 11 Ricerca di epicode.internal con HTTPS

Da queste due schermate si può già notare come HTTPS rilevi il nostro dominio come potenzialmente pericoloso a causa della convalida del certificato in quanto **self-signed**, ovvero “**auto-firmato**”.

1.9 SNIFFING CON WIRESHARK

L'ultima fase del progetto prevede di effettuare uno **sniffing** dei pacchetti utilizzando l'applicazione **WireShark**, rilevando anche i **MAC Address** di **sorgente** e **destinatario**. Nella figura 12 è riportato uno sniffing HTTP, mentre nella fig. 13 in HTTPS. Le differenze sono notevoli in quanto HTTPS presenta un controllo maggiore sui pacchetti denotato dalla presenza del protocollo **TLS**.

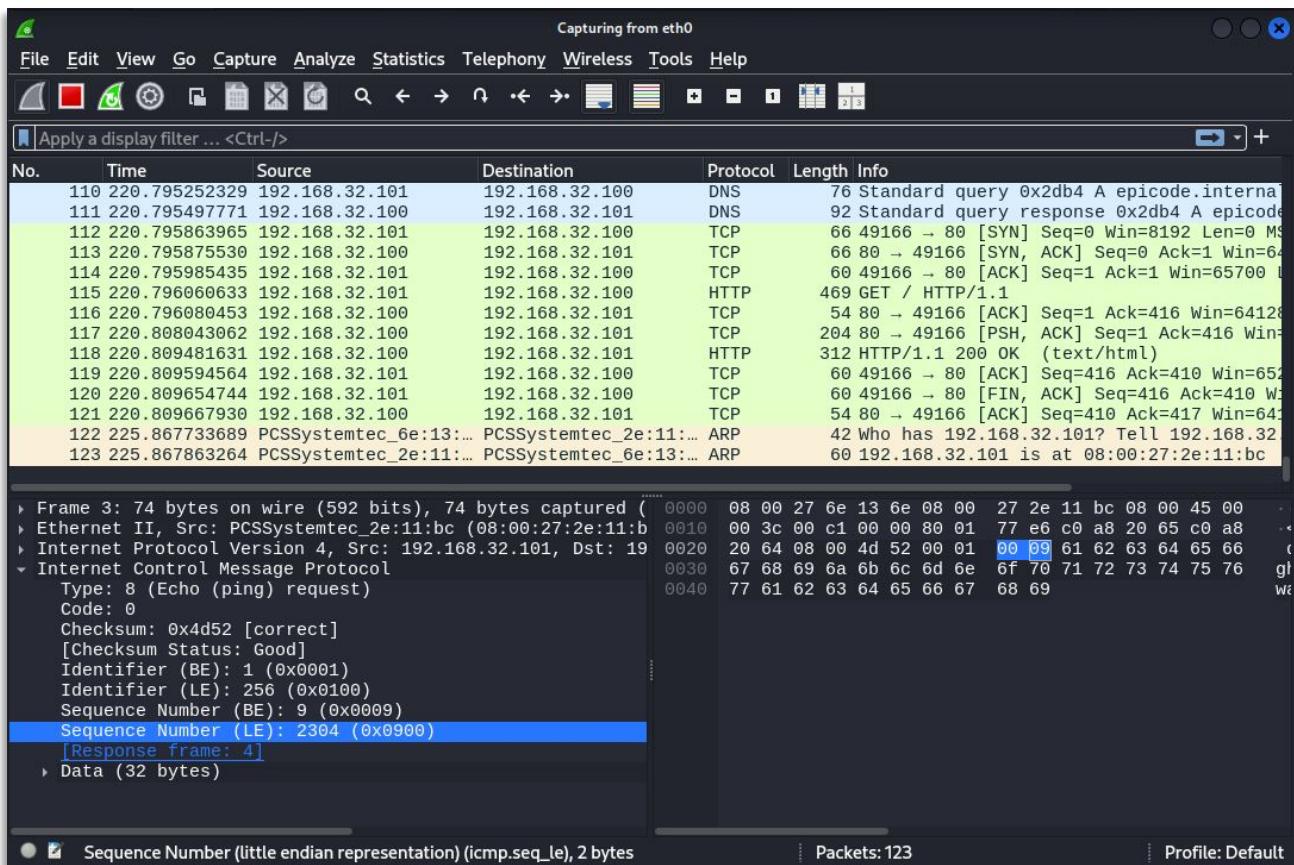


Figura 12 Sniffing HTTP

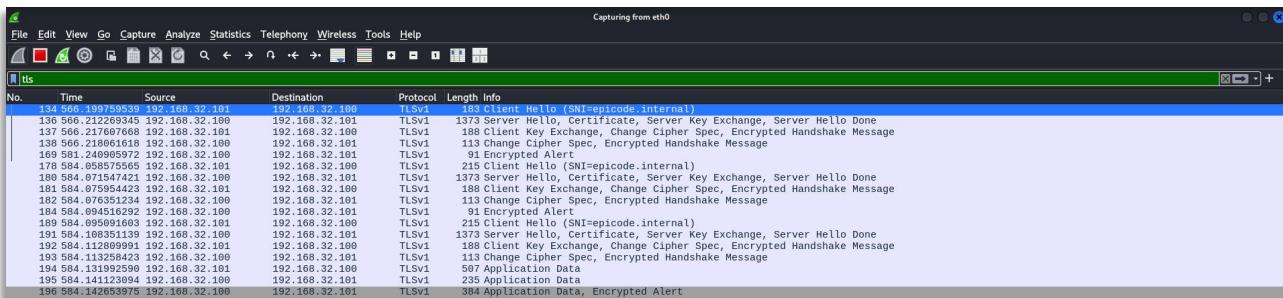


Figura 13 Sniffing HTTPS

I MAC Address possiamo facilmente vederli aprendo un qualsiasi pacchetto in HTTP/HTTPS. In questo caso il MAC Address sorgente è **08:00:27:6e:13:6e**, mentre il destinatario **08:00:27:2e:11:bc** in sniffing HTTPS, mentre per HTTP i MAC Address saranno i medesimi ma invertiti, quindi il sorgente **08:00:27:2e:11:bc** e il destinatario **08:00:27:6e:13:6e**.

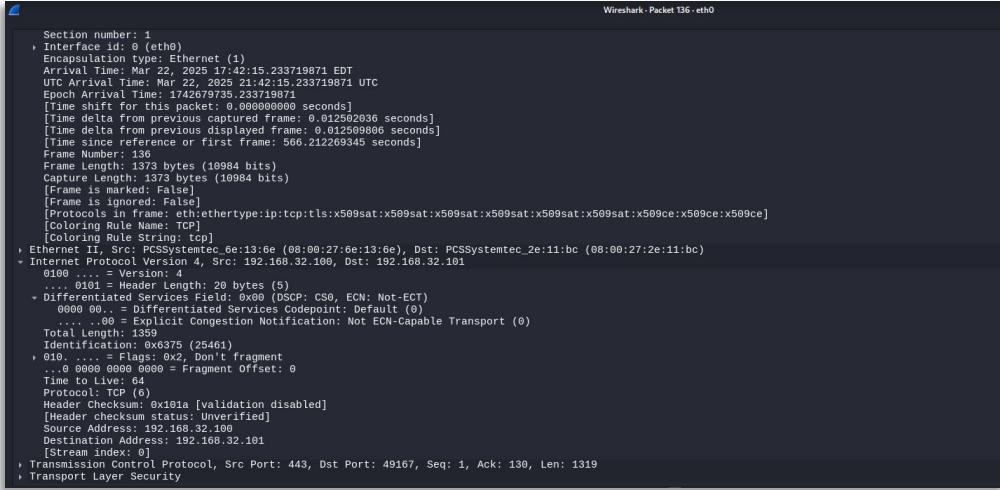


Figura 14 Apertura pacchetto in sniffing HTTPS

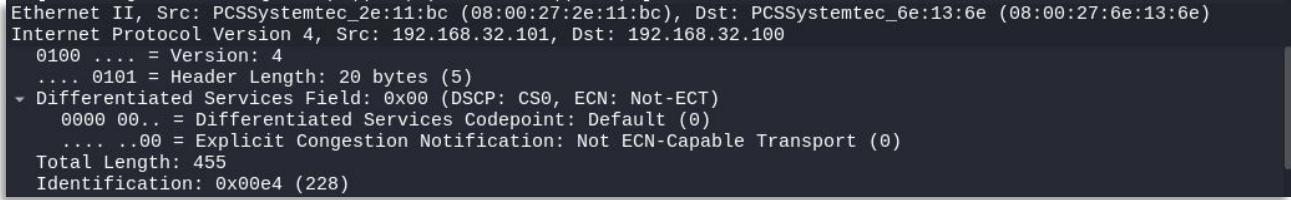


Figura 15 Apertura pacchetto in sniffing HTTP