

Progetto di fine Modulo M1

Antonio Piredda

20.10.2025

INTRODUZIONE

In questo progetto ci viene chiesto di simulare un'architettura client/server nella quale siano attivi sia il servizio HTTPS (in seguito solo HTTP) e il servizio DNS. Si richiede inoltre di intercettare il traffico per evidenziare la differenza tra il traffico HTTPS e HTTP.

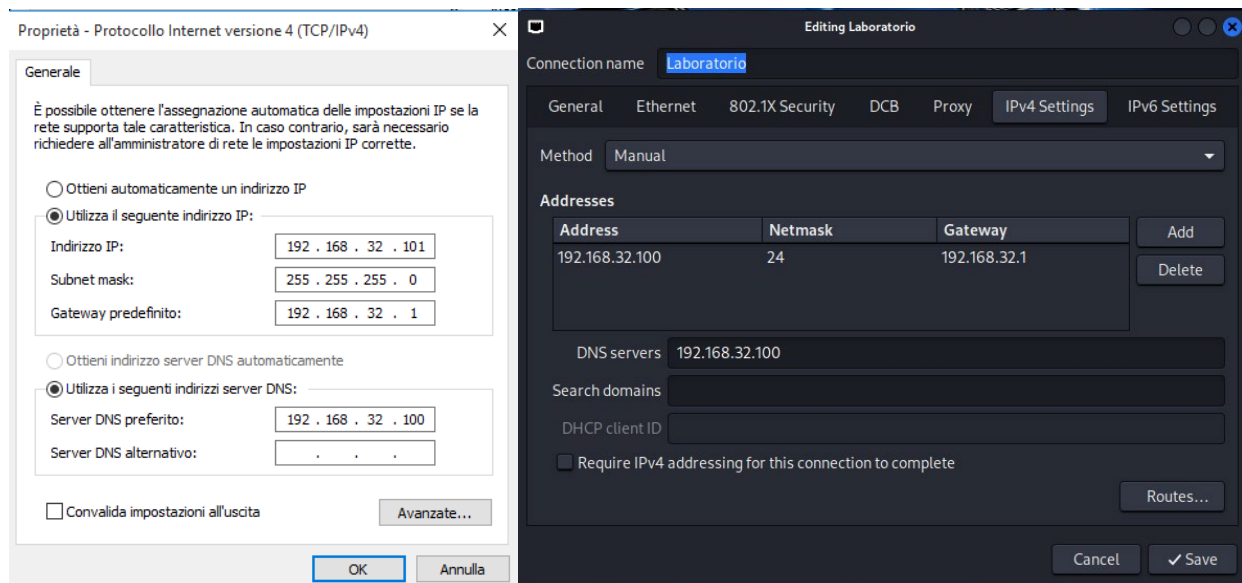
PROCEDIMENTO

Si procede ad impostare gli indirizzi assegnati alle macchine virtuali:

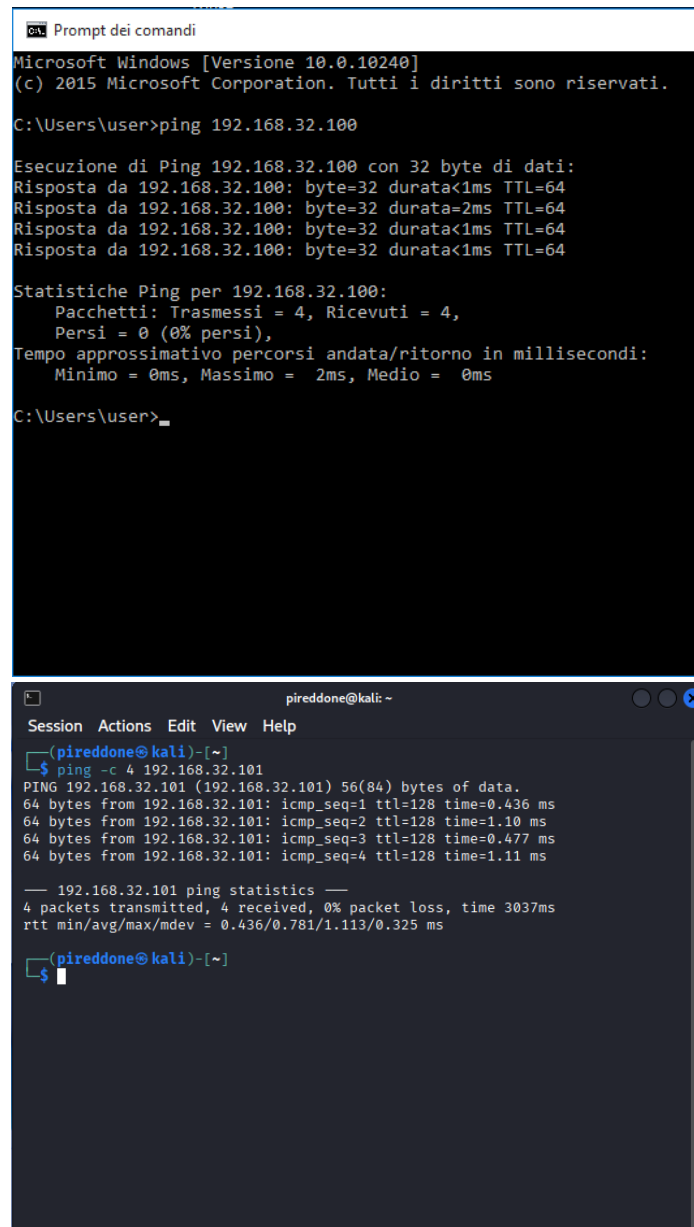
- Kali 192.168.32.100

- Win10 192.168.32.101

Ho proceduto ad assegnare l'indirizzo della MV Kali (192.168.32.100) come indirizzo DNS per entrambe le macchine.



Una volta assegnati gli indirizzi alle macchine virtuali eseguo un controllo di PING per verificare che le macchine comunichino tra loro. Nella prima immagine il ping da Win10 a Kali, nella seconda immagine il ping da Kali a Win10.



The image contains two screenshots of terminal windows. The top window is a Windows 10 Command Prompt titled 'Prompt dei comandi'. It shows a successful ping from a Windows 10 machine to 192.168.32.100. The output includes details about the ping execution (32 bytes, <1ms TTL=64) and statistics (4 packets transmitted, 4 received, 0% loss). The bottom window is a Kali Linux terminal titled 'pireddone@kali: ~'. It shows a successful ping from the Kali machine to 192.168.32.101. The output includes details about the ping execution (56(84) bytes, times around 1ms) and statistics (4 packets transmitted, 4 received, 0% loss).

```
Prompt dei comandi
Microsoft Windows [Versione 10.0.10240]
(c) 2015 Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\user>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
Risposta da 192.168.32.100: byte=32 durata=2ms 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 = 2ms, Medio = 0ms

C:\Users\user>
```

```
pireddone@kali: ~
Session Actions Edit View Help

(pireddone@kali)-[~]
$ ping -c 4 192.168.32.101
PING 192.168.32.101 (192.168.32.101) 56(84) bytes of data.
64 bytes from 192.168.32.101: icmp_seq=1 ttl=128 time=0.436 ms
64 bytes from 192.168.32.101: icmp_seq=2 ttl=128 time=1.10 ms
64 bytes from 192.168.32.101: icmp_seq=3 ttl=128 time=0.477 ms
64 bytes from 192.168.32.101: icmp_seq=4 ttl=128 time=1.11 ms

— 192.168.32.101 ping statistics —
4 packets transmitted, 4 received, 0% packet loss, time 3037ms
rtt min/avg/max/mdev = 0.436/0.781/1.113/0.325 ms

(pireddone@kali)-[~]
$
```

Una volta eseguiti questi controlli di comunicazione ho proceduto alla configurazione dei servizi HTTPS e DNS, incontrando qualche difficoltà nella configurazione del DNS, poiché riscontravo dei problemi con la versione 1.50 di Perl già installata.

Questo problema di versione creava una incompatibilità tra Perl e Inetsim che non faceva funzionare il DNS. Ho eseguito una ricerca ed ho scoperto che per un utilizzo senza problemi e senza conflitti tra Perl e Inetsim avrei dovuto scaricare la versione 1.37.

Di seguito il comando eseguito (che ho ritrovato su un forum) per l'installazione della versione 1.37:

```
curl -O https://cpan.metacpan.org/authors/id/N/NL/NLNETLABS/Net-DNS-1.37.tar.gz
```

```
tar xzf Net-DNS-1.37.tar.gz
```

```
cd Net-DNS-1.37
```

```
perl Makefile.PL
```

```
make
```

```
make test
```

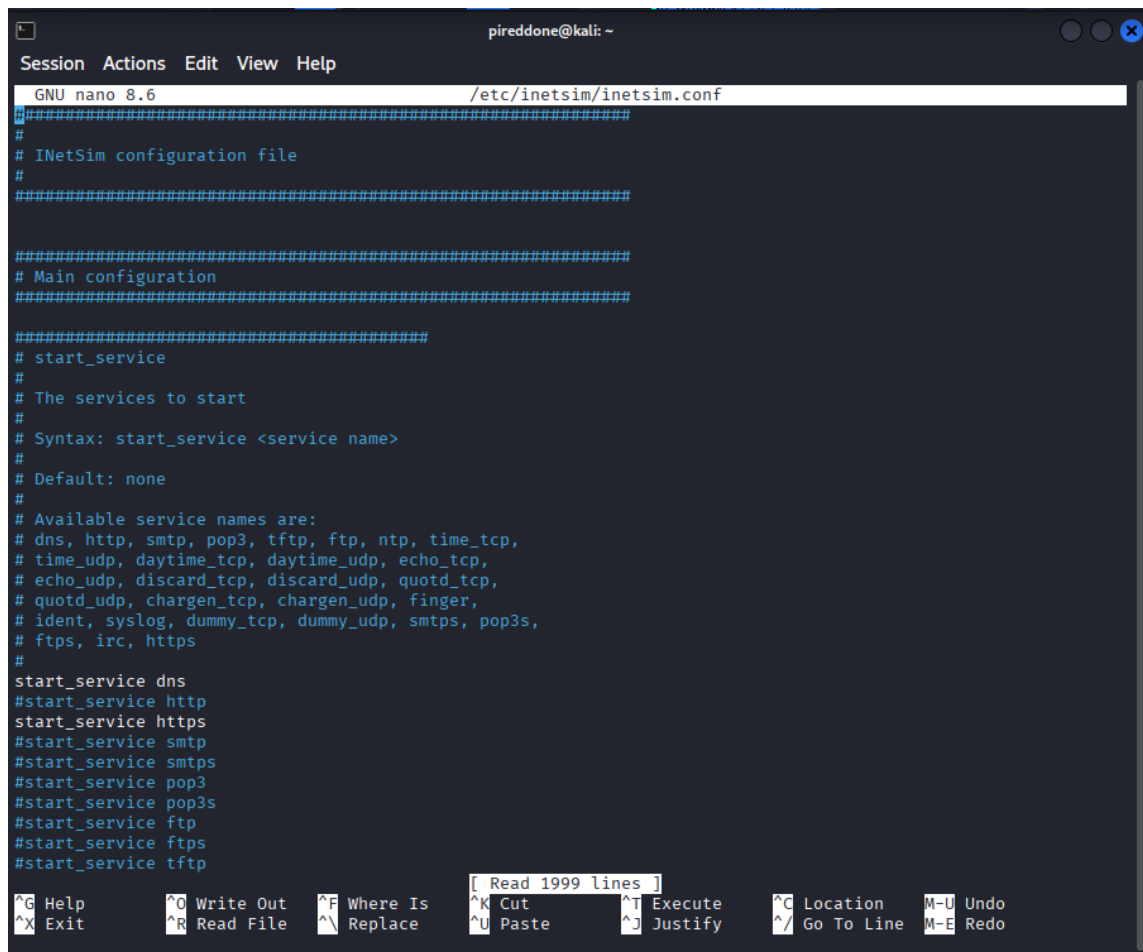
```
sudo make install
```

Una volta eseguita questa modifica, i servizi hanno funzionato a dovere, di seguito illustro le fasi della configurazione dei servizi.

Abilitazione di HTTPS e DNS

(accendiamo al file di configurazione con `sudo nano /etc/inetsim/inetsim.conf`)

In questa schermata andiamo ad eliminare il “#” che va a commentare le righe **start_service dns** e **start_service https** in modo che queste vengano lette ed eseguite.



```
GNU nano 8.6 /etc/inetsim/inetsim.conf
#####
#
# InetSim configuration file
#
#####

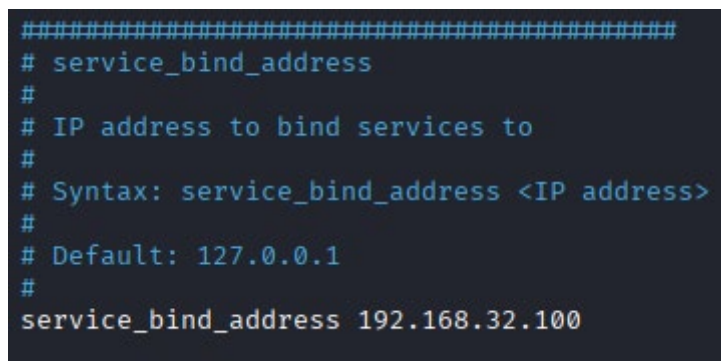
#####
# Main configuration
#####

#####
# 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, 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

[ Read 1999 lines ]
^G Help      ^O Write Out  ^F 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
```

Abilitazione e modifica dei parametri DNS

In service bind address ho impostato come IP di ascolto l'indirizzo della macchina Kali, poiché altrimenti Inetsim avrebbe impostato di default 127.0.0.1 rendendo impossibile lo svolgimento del progetto.



```
#####
# 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
```

In questa schermata ho proceduto ad abilitare la porta 53 in ascolto, ho impostato come indirizzo di default del DNS l'indirizzo 192.168.32.100 ed ho assegnato il DNS Hostname "epicode".

```
Session Actions Edit View Help
GNU nano 8.6 /etc/inetsim/inetsim.conf
# Service DNS
#####
# dns_bind_port
#
# Port number to bind DNS service to
# Syntax: dns_bind_port <port number>
# Default: 53
#
dns_bind_port 53

#####
# 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

#####
# dns_default_hostname
#
# Default hostname to return with DNS replies
# Syntax: dns_default_hostname <hostname>
# Default: www
#
dns_default_hostname epicode
```

Ho proceduto in seguito ad associare l'indirizzo 192.168.32.100 a "epicode.internal" utilizzando la funzione "dns_static" di inetsim.

```
#####
# 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 epicode.internal 192.168.32.100
```


Nella parte finale della configurazione ho proceduto ad abilitare (rimuovendo il #) la porta 80 per il servizio HTTP e la porta 443 per il servizio HTTPS.

Avvio del servizio INETSIM

```
(pireddone@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 28051) ==  
Session ID: 28051  
Listening on: 192.168.32.100  
Real Date/Time: 2025-10-20 14:59:45  
Fake Date/Time: 2025-10-20 14:59:45 (Delta: 0 seconds)  
Forking services ...  
* dns_53_tcp_udp - started (PID 28061)  
* https_443_tcp - started (PID 28062)  
done.  
Simulation running.  
█
```

Verifica del funzionamento - nslookup

Per verificare il corretto funzionamento del servizio DNS ho proceduto con un nslookup sia su Win10 che su Kali, con esito positivo.

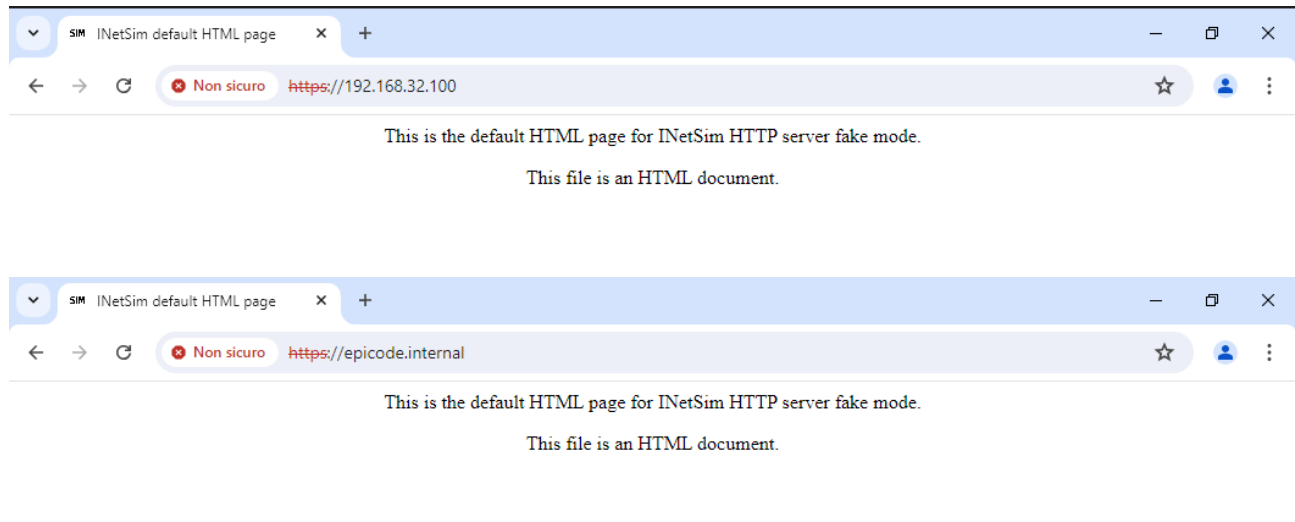
 Prompt dei comandi

```
Microsoft Windows [Versione 10.0.10240]  
(c) 2015 Microsoft Corporation. Tutti i diritti sono riservati.  
  
C:\Users\user>nslookup epicode.internal  
Server: epicode.internal  
Address: 192.168.32.100  
  
Nome: epicode.internal  
Address: 192.168.32.100  
  
C:\Users\user>
```

```
(pireddone@kali)-[~]  
$ nslookup epicode.internal 192.168.32.100  
Server: 192.168.32.100  
Address: 192.168.32.100#53  
  
Name: epicode.internal  
Address: 192.168.32.100  
  
(pireddone@kali)-[~]  
$ █
```

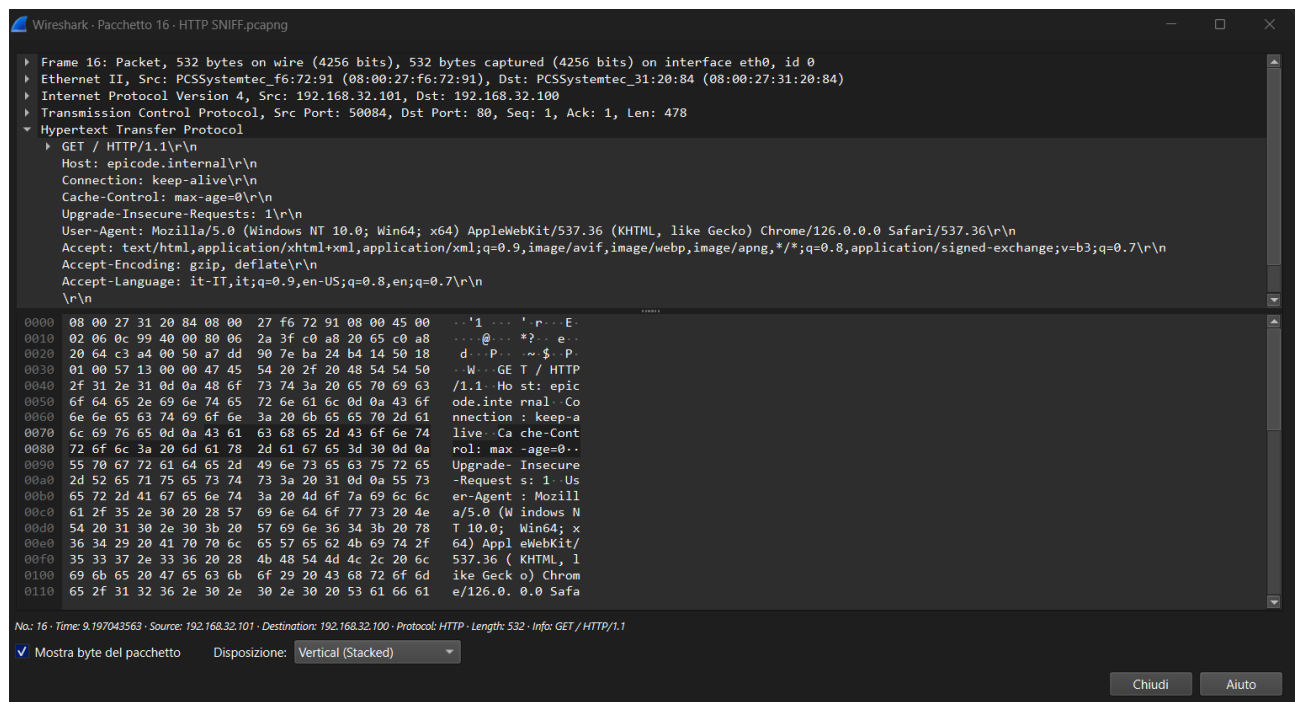
Verifica del funzionamento – Chrome

Per verificare che da Win10 riusciamo a raggiungere il servizio HTTPS attivo, ho provato ad utilizzare Chrome per raggiungere sia il 192.168.32.100 che “epicode.internal”, in entrambe i casi con esito positivo, raggiungendo questa schermata di seguito.

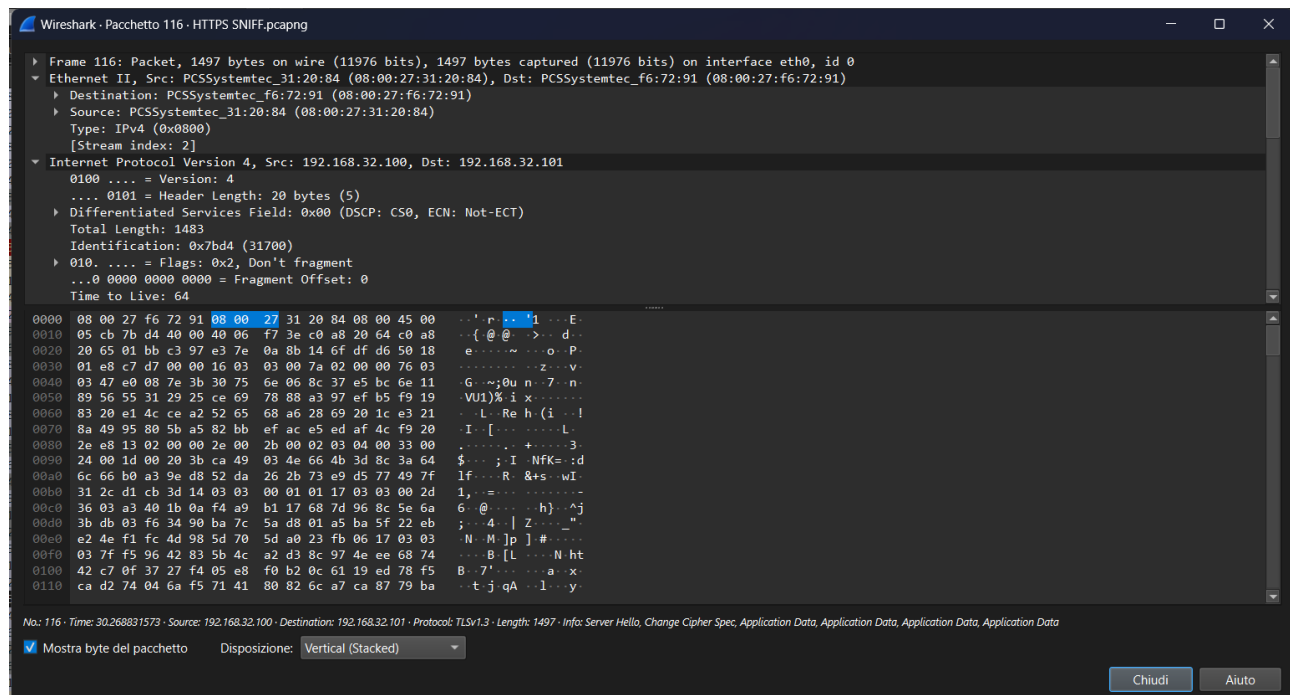


Analisi di Wireshark e conclusione

Nella schermata di Wireshark, con il servizio **HTTP** attivo su Inetsim, tutto rimane in chiaro, poiché si nota il metodo (GET), l’URL richiesto, l’header e il contenuto.



Nella schermata di Wireshark, con il servizio **HTTPS** attivo su Inetsim invece tutto è cifrato con TLS, il che dimostra l'efficacia del servizio.



In entrambe i casi, gli indirizzi MAC invece restano visibili, essendo al livello 2 della pila ISO/OSI.