

Compito di Fine Modulo W4D4

Corso: Cybersecurity Analyst – Epicode

Studiante: Daniele Taormina

Data: 16/10/2025

Introduzione

In questo compito di fine modulo viene realizzata una simulazione pratica che unisce tutte le competenze apprese finora nel corso. L'obiettivo è ricreare , in un ambiente di laboratorio virtuale (rete privata) , una configurazione **Client-Server** tra 2 macchine:

Kali Linux (Server) : 192.168.32.100

Windows (Client) :192.168.32.101

In questa simulazione vengono configurati i servizi fondamentali di rete:

DNS : risoluzione dei nomi di dominio, quindi tradurre un indirizzo **IP** (192.168.32.100) in nome (epicode.internal)

HTTP/HTTPS : protocolli per la visualizzazione e la comunicazione nel web.

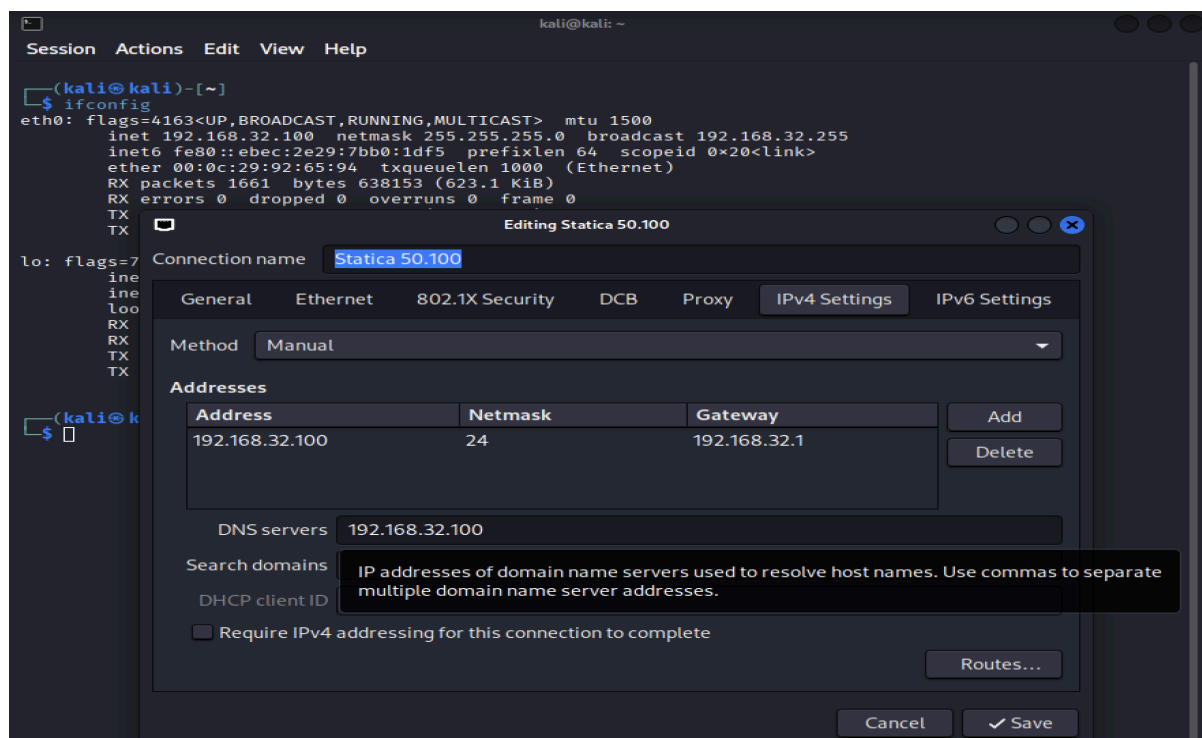
Infine con **Wireshark (packet sniffer)**, verranno osservati e analizzati il flusso dei dati e le relative differenze tra i protocolli **HTTP** (non cifrata) e **HTTPS** (cifrata) .

Obiettivi del compito

- Simulare una rete **client-server** in un ambiente di laboratorio virtuale isolato.
- Far comunicare un computer (**Windows**) con un server (**Kali Linux**) tramite **browser**.
- Richiedere una risorsa web interna chiamata *epicode.internal*.
- Osservare e analizzare la comunicazione con **Wireshark** (un **packet sniffer**).
- Identificare gli **indirizzi MAC** di chi invia e riceve i pacchetti.
- Visualizzare il contenuto della richiesta quando la connessione è **HTTPS**.
- Ripetere la prova con una connessione **HTTP** non sicura.
- Confrontare le differenze tra il traffico **HTTPS** e quello **HTTP**.
- Spiegare perché la cifratura dei dati è importante per la sicurezza delle comunicazioni.

Esecuzione

Per prima cosa vado a configurare gli indirizzi **IP** delle macchine in modo tale da creare una rete privata. Per far questo è necessario configurare la rete con un nuovo indirizzo **IP**. (192.168.32.100) necessario anche impostare il **DNS** sempre con lo stesso **IP**.



Per un ulteriore conferma faccio un double-check sull'interfaccia di rete, e controllo se l'indirizzo **IP** è effettivamente cambiato. A volte è necessario riavviare la scheda di rete o più semplicemente la macchina. Per questo utilizzo il comando **ifconfig**.

Stessa cosa vale per la macchina **Windows**.

Network & internet > Ethernet

Metered connection

Some apps might work differently to reduce data usage when you're connected to this network

Off 

[Set a data limit to help control data usage on this network](#)

IP assignment:

Manual

IPv4 address:

192.168.32.101

Edit

IPv4 mask:

255.255.255.0

IPv4 gateway:

192.168.32.1

DNS server assignment:

Manual

Edit

IPv4 DNS servers:

192.168.32.100 (Unencrypted)

Link speed (Receive/Transmit):

Copy

1000/1000 (Mbps)

Link-local IPv6 address:

fe80::b242:ca5f:d695:1148%5

IPv4 address:

192.168.32.101

IPv4 DNS servers:

192.168.32.100 (Unencrypted)

Manufacturer:

Intel Corporation

Description:

Intel(R) 82574L Gigabit Network Connection

Quindi, sempre accedendo alle impostazioni della scheda di rete modifico l'indirizzo **IP**, in questo caso (192.168.32.101). Anche qui è necessario configurare il **DNS IP** (192.168.32.100), Controllo anche qui dal terminale se la configurazione è avvenuta con successo.

Comando: **ipconfig /all**

```

Ethernet adapter Ethernet0:

    Connection-specific DNS Suffix  . : 
    Description . . . . . : Intel(R) 82574L Gigabit Network Connection
    Physical Address. . . . . : 00-0C-29-CF-24-10
    DHCP Enabled. . . . . : No
    Autoconfiguration Enabled . . . . : Yes
    Link-local IPv6 Address . . . . . : fe80::b242:ca5f:d695:1148%5(Preferred)
    IPv4 Address. . . . . : 192.168.32.101(Preferred)
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.32.1
    DHCPv6 IAID . . . . . : 100666409
    DHCPv6 Client DUID. . . . . : 00-01-00-01-30-65-3D-68-00-0C-29-CF-24-10
    DNS Servers . . . . . : 192.168.32.100
    NetBIOS over Tcpip. . . . . : Enabled

C:\Users\Dani>

```

Adesso che la configurazione delle macchine è terminata occorre configurare

inetsim, che verrà utilizzato per simulare un server **HTTP/HTTPS/DNS**.

```

kali
Session Actions Edit View Help
(kali@kali)-[~]
$ sudo nano /etc/inetsim/inetsim.conf

```

Con questo comando modifico il file di configurazione con privilegi **root**.

All'interno del file inetsim.conf trovo una lista di servizi, che per essere attivati devono essere "decommentati" togliendo il **#** iniziale. Per questo compito i servizi che occorrono sono i primi 3. **DNS, HTTP** e **HTTPS**. Gli altri servizi restano commentati con **#**, quindi inattivi.

```

#
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

```

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

Proseguendo all'interno del file trovo il **service_bind_address** che configuro con l'**IP** a cui i servizi faranno riferimento.

```
#####
# 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
#
dns_default_hostname epicode
#

#####
# dns_default_domainname
#
# Default domain name to return with DNS replies
#
# Syntax: dns_default_domainname <domain name>
#
# Default: inetsim.org
#
dns_default_domainname internal
```

Service **DNS** , questa è una parte molto cruciale perchè sarà questa la configurazione che permetterà la traduzione da **IP** a **Dominio**.

epicode.internal = 192.168.32.100

Esempio:

www.google.com = 8.8.8.8 o 8.8.4.4

Quindi attivando queste impostazioni concludiamo così la configurazione del **DNS**

```
dns_bind_port 53
dns_default_ip 192.168.32.100
dns_default_hostname epicode
dns_default_domainname internal
```

Salviamo il file aggiornato:

CTRL + o
Invio
CTRL + x

Digitando: `sudo inetsim` si verifica un errore, non di configurazione ma di incompatibilità.

```
Forking services...
* dns_53_tcp_udp - started (PID 43778)
Can't locate object method "main_loop" via package "Net::DNS::Nameserver" at /usr/share/perl5/INetSim/DNS.pm line 69.
```

Un'incompatibilità tra **inetsim** e la versione attuale del modulo **Perl Net::DNS**. Il metodo **main_loop**, usato per eseguire il server **DNS**, non esiste più nella nuova versione della libreria. Di conseguenza, il servizio **DNS** non riesce ad avviarsi correttamente. Per risolvere questo problema ho trovato 2 soluzioni (2a soluzione Pag.12) anche se in realtà avremmo potuto utilizzare un tool simile a inetsim, ho fatto un downgrade di **Perl Net::DNS** (v1.50) alla versione (1.37):

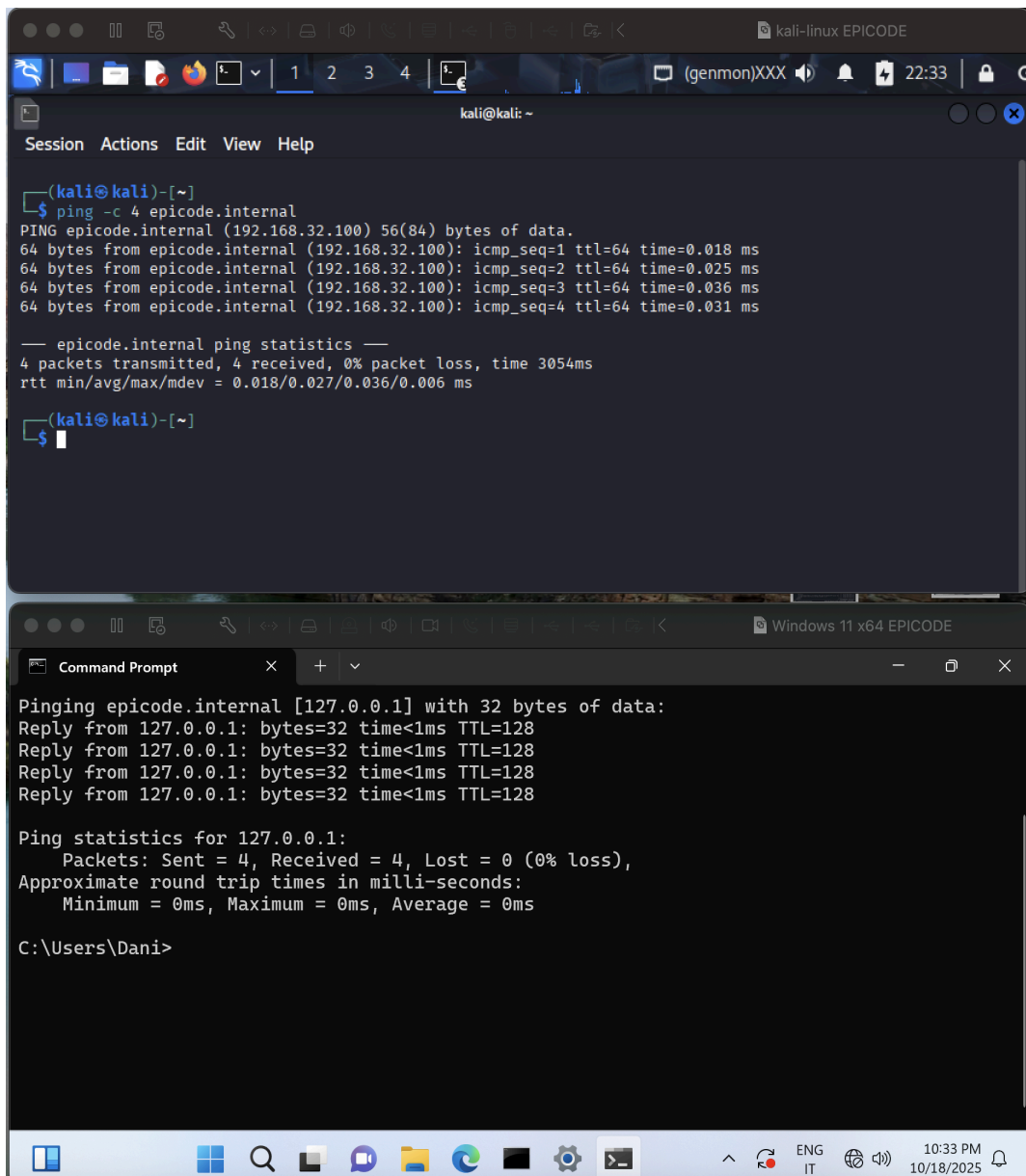
- `curl -O`
<https://cpan.metacpan.org/authors/id/N/NL/NLNETLABS/Net-DNS-1.37.tar.gz>
- `tar xzf Net-DNS-1.37.tar.gz`
- `cd Net-DNS-1.37`
- `perl Makefile.PL`
- `make`
- `make test`
- `sudo make install`

Così facendo il servizio DNS viene eseguito con successo insieme ad HTTP e HTTPS.

```
(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 2156) ==
Session ID:      2156
Listening on:    192.168.32.100
Real Date/Time:  2025-10-18 21:17:46
Fake Date/Time: 2025-10-18 21:17:46 (Delta: 0 seconds)
Forking services ...
* dns_53_tcp_udp - started (PID 2158)
* http_80_tcp    - started (PID 2159)
* https_443_tcp  - started (PID 2160)
done.
Simulation running.
```

È obbligatorio mantenere questo terminale attivo altrimenti i servizi cesseranno di funzionare.

Eseguo subito una prova di **ping** dimostrando così che oltre che alla connessione del **client** al **server**, anche il **dominio** epicode.internal viene riconosciuto e interpretato da 192.168.32.100.

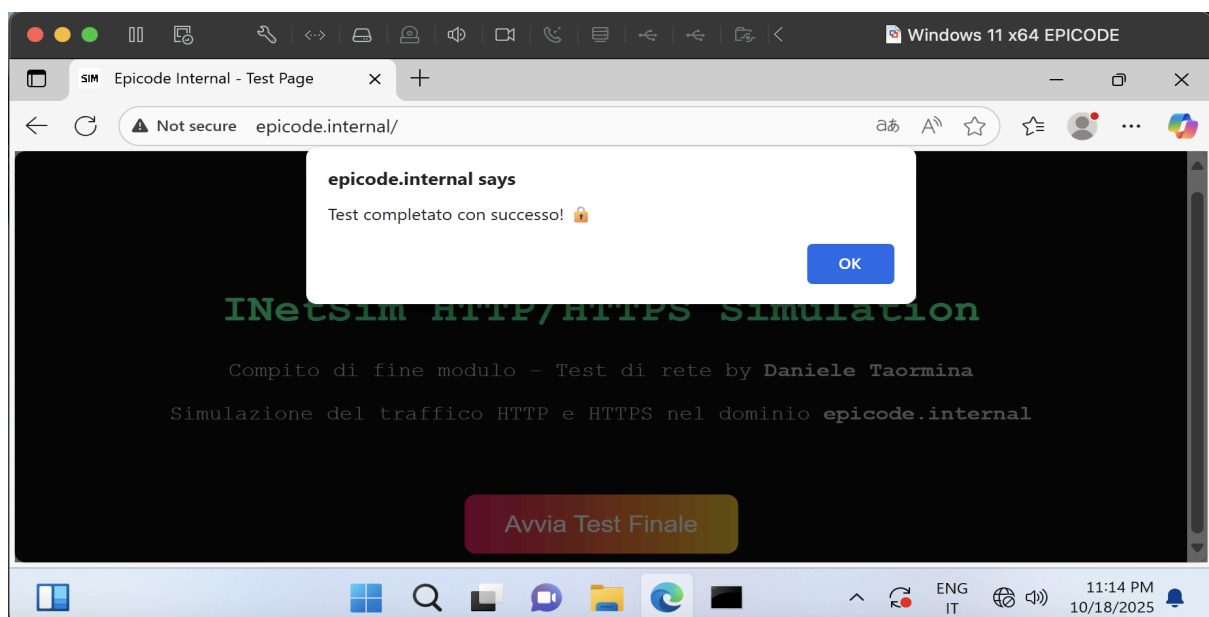
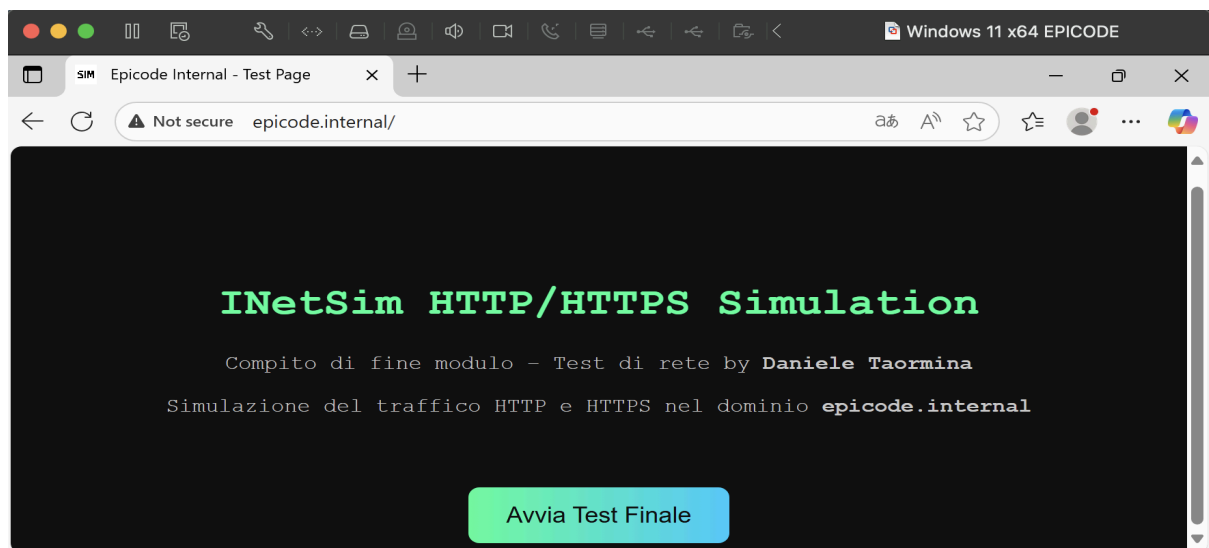
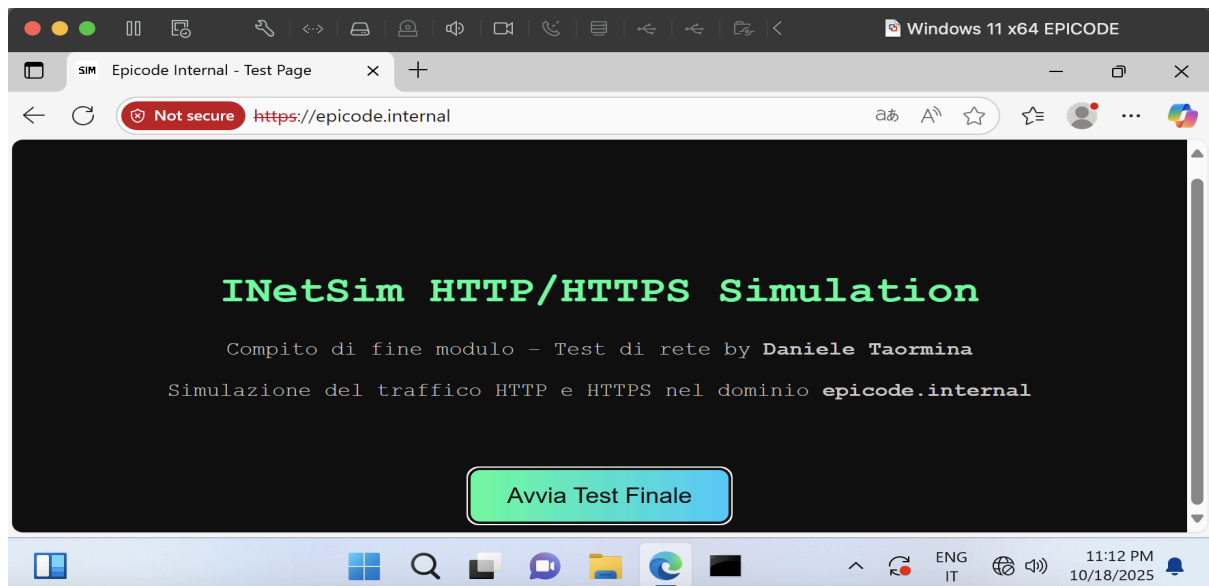


The image consists of two screenshots of terminal windows. The top screenshot is from a Kali Linux terminal, showing a ping command being executed against the domain 'epicode.internal'. The output shows four successful ping requests with varying response times and TTL values. The bottom screenshot is from a Windows 11 Command Prompt, showing a ping command being executed against the IP address '127.0.0.1'. The output shows four successful ping requests with a response time of less than 1ms and a TTL of 128.

```
kali@kali: ~  
$ ping -c 4 epicode.internal  
PING epicode.internal (192.168.32.100) 56(84) bytes of data:  
64 bytes from epicode.internal (192.168.32.100): icmp_seq=1 ttl=64 time=0.018 ms  
64 bytes from epicode.internal (192.168.32.100): icmp_seq=2 ttl=64 time=0.025 ms  
64 bytes from epicode.internal (192.168.32.100): icmp_seq=3 ttl=64 time=0.036 ms  
64 bytes from epicode.internal (192.168.32.100): icmp_seq=4 ttl=64 time=0.031 ms  
  
--- epicode.internal ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3054ms  
rtt min/avg/max/mdev = 0.018/0.027/0.036/0.006 ms  
  
kali@kali: ~  
$
```

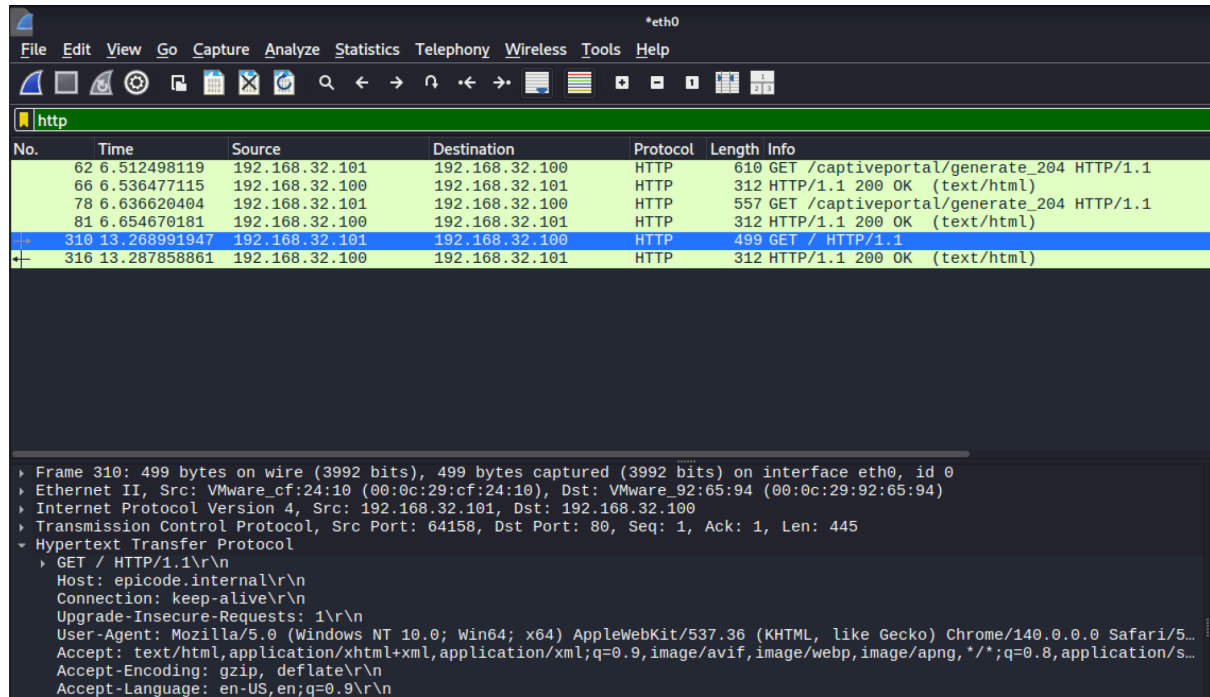
```
Windows 11 x64 EPICODE  
Command Prompt  
Pinging epicode.internal [127.0.0.1] with 32 bytes of data:  
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128  
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128  
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128  
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128  
  
Ping statistics for 127.0.0.1:  
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),  
    Approximate round trip times in milli-seconds:  
        Minimum = 0ms, Maximum = 0ms, Average = 0ms  
  
C:\Users\Dani>
```


Entrambe le pagine vengono visualizzate con successo.



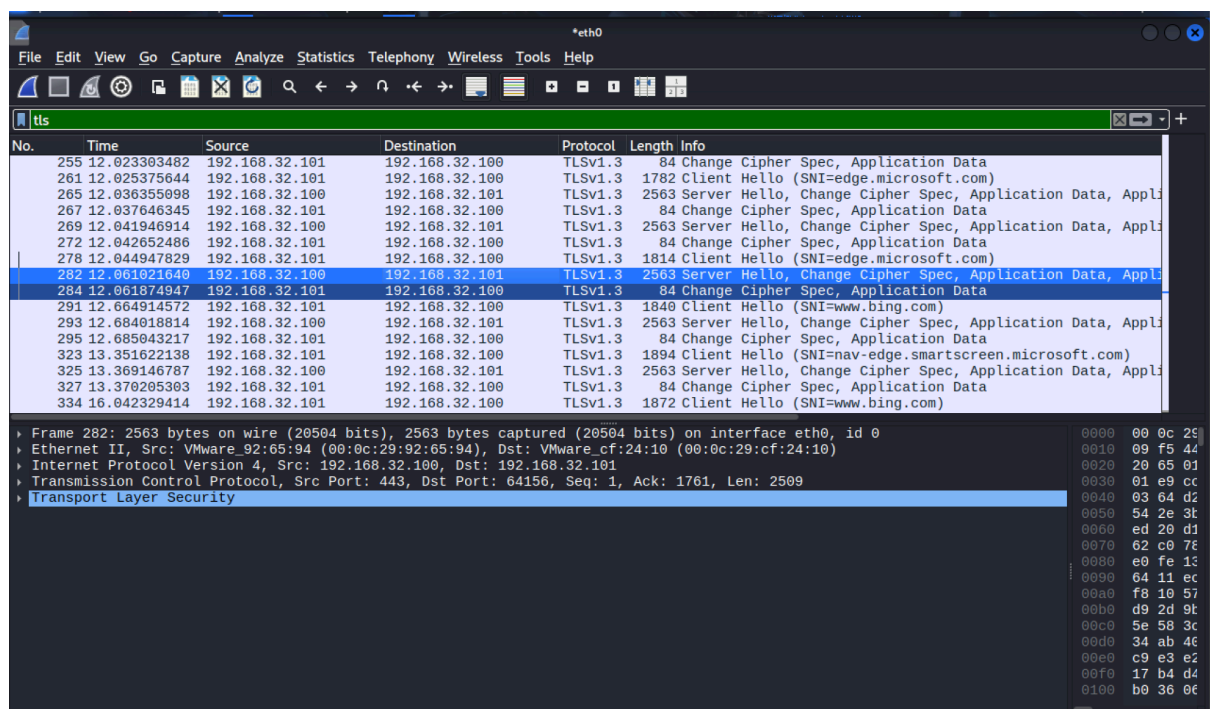
Analisi del traffico e dei pacchetti con **Wireshark**

In questa ultima parte del compito spiego le differenze dei dati che ho intercettato:



Analisi pacchetto HTTP (Wireshark)

- **Protocollo:** HTTP su TCP
- **Interfaccia di cattura:** eth0
- **MAC sorgente:** 00:0c:29:cf:24:10 (Kali)
- **MAC destinazione:** 00:0c:29:92:65:94 (Windows)
- **IP sorgente:** 192.168.32.101 (Kali)
- **IP destinazione:** 192.168.32.100 (Windows)
- **Porta sorgente:** 64158
- **Porta destinazione:** 80
- **Metodo HTTP:** GET / HTTP/1.1
- **Host richiesto:** epicode.internal
- **User-Agent:** Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 Chrome/140.0.0.0 Safari/537.36



Analisi pacchetto HTTPS (Wireshark)

- **Protocollo:** TLSv1.3 su TCP (HTTPS)
- **Interfaccia di cattura:** eth0
- **MAC sorgente:** 00:0c:29:92:65:94 (Kali)
- **MAC destinazione:** 00:0c:29:cf:24:10 (Windows)
- **IP sorgente:** 192.168.32.100 (Kali)
- **IP destinazione:** 192.168.32.101 (Windows)
- **Porta sorgente:** 443
- **Porta destinazione:** 64156
- **Versione TLS:** TLS 1.3
- **Lunghezza pacchetto:** 2563 byte
- Il contenuto dei dati appare come **“Application Data”** **cifrato**, non leggibile.
- Nessun metodo HTTP (GET/POST) è visibile perché la comunicazione è protetta dal canale TLS.
- HTTPS garantisce **confidenzialità, integrità e autenticazione** del server tramite certificato digitale.
- Le informazioni in chiaro (host, browser, header HTTP) non sono più visibili come nel traffico HTTP.

Conclusione

Nel traffico **HTTP** i dati viaggiano in chiaro e quindi tramite **Wireshark** è possibile leggere tutto il contenuto della comunicazione, l'host richiesto, l'user agent e gli altri header inviati dal browser. Questo dimostra che il protocollo **HTTP** non offre alcuna protezione sulla riservatezza dei dati, perchè chiunque riesce a intercettare il traffico e può visualizzare le informazioni trasmesse tra client e server.

Nel traffico **HTTPS** invece la comunicazione avviene attraverso un canale cifrato, che garantisce **confidenzialità, integrità e autenticazione (CIA)** del server tramite certificato digitale. In Wireshark non è più possibile visualizzare i contenuti delle richieste perché i dati sono racchiusi nei pacchetti "Application Data", completamente cifrati. Rimangono visibili solo gli indirizzi **MAC**, gli **IP** e le porte **TCP** necessarie alla trasmissione dei pacchetti, ma tutto quello che riguarda la parte applicativa non è più leggibile.

In sintesi il traffico **HTTP** risulta esposto e facilmente analizzabile, mentre il traffico **HTTPS** nasconde le informazioni sensibili e protegge la comunicazione contro intercettazioni o modifiche non autorizzate.

Sintesi finale

Ho utilizzato **INetSim** per simulare servizi Internet come **HTTP**, **HTTPS** e **DNS** in un ambiente isolato, e **Wireshark** per analizzare il traffico generato.

L'obiettivo era osservare la differenza tra una comunicazione non cifrata (**HTTP**) e una cifrata (**HTTPS**).

Dall'analisi è emerso che **HTTP** mostra tutte le informazioni in chiaro, mentre **HTTPS** protegge rendendo i dati illeggibili.

Il test dimostra in modo pratico l'importanza della cifratura e della sicurezza delle comunicazioni in rete.

Seconda Soluzione:

La seconda soluzione consisteva nel modificare in entrambe le macchine, il file hosts aggiungendo manualmente l'indirizzo IP e Dominio, forzando così la configurazione del DNS.

```
(kali@kali)-[~]
$ cd /etc

(kali@kali)-[/etc]
$ cat hosts
127.0.0.1    localhost
127.0.1.1    kali
::1         localhost ip6-localhost ip6-loopback
ff02::1     ip6-allnodes
ff02::2     ip6-allrouters
192.168.32.100 epicode.internal

(kali@kali)-[/etc]
$
```

```
Windows 11 x64 EPICODE
Command Prompt

C:\Windows\System32\drivers\etc>
C:\Windows\System32\drivers\etc>dir
Volume in drive C has no label.
Volume Serial Number is 1E29-18BE

Directory of C:\Windows\System32\drivers\etc

10/18/2025  04:07 PM    <DIR>          .
10/09/2025  10:40 PM    <DIR>          ..
10/18/2025  03:58 PM             858 hosts
10/18/2025  09:17 PM             857 hosts.txt
05/07/2022  07:22 AM          3,683 lmhosts.sam
05/07/2022  07:22 AM             407 networks
05/07/2022  07:22 AM          1,358 protocol
05/07/2022  07:22 AM          17,635 services
               6 File(s)          24,798 bytes
               2 Dir(s)  43,117,658,112 bytes free

C:\Windows\System32\drivers\etc>type hosts.txt
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#       102.54.94.97       rhino.acme.com       # source server
#       38.25.63.10       x.acme.com           # x client host
#
# localhost name resolution is handled within DNS itself.
#       127.0.0.1         localhost
#       ::1               localhost
#
192.168.32.100 epicode.internal
C:\Windows\System32\drivers\etc>
```

Glossario tecnico

DNS (Domain Name System)

Sistema che converte i nomi di dominio nei relativi indirizzi **IP**.

GET

Metodo **HTTP** usato per richiedere risorse a un server.

HTTP (HyperText Transfer Protocol)

Protocollo web che trasferisce dati in chiaro tra client e server.

HTTPS (HyperText Transfer Protocol Secure)

Versione cifrata di HTTP che utilizza **TLS** per proteggere le comunicazioni.

INetSim (Internet Services Simulation Suite)

Software che simula servizi Internet (HTTP, HTTPS, DNS) in ambiente isolato.

IP (Internet Protocol Address)

Indirizzo logico che identifica un dispositivo all'interno di una rete.

MAC (Media Access Control Address)

Indirizzo fisico univoco della scheda di rete di un dispositivo.

TCP (Transmission Control Protocol)

Protocollo di trasporto che assicura la consegna corretta dei dati.

TLS (Transport Layer Security)

Protocollo che cifra e protegge i dati scambiati su rete.

Wireshark

Strumento che cattura e analizza il traffico di rete in tempo reale.