

Progetto Finale Modulo 1

1. Obiettivo

L'obiettivo di questo esercizio è simulare, in ambiente di laboratorio virtuale, un'architettura client-server in cui un client Windows (precedentemente configurato 192.168.50.102) richiede una risorsa web all'hostname epicode.internal, servita da un server Kali Linux (precedentemente configurato 192.168.50.100) tramite protocollo HTTPS, con risoluzione del nome fornita dal servizio DNS del server.

Successivamente, la prova viene ripetuta utilizzando HTTP al posto di HTTPS.

Durante le prove, il traffico di rete è stato intercettato e analizzato con Wireshark, con l'obiettivo di evidenziare:

- MAC address sorgente e destinazione.
- Contenuto della richiesta HTTPS e HTTP.
- Differenze tra traffico cifrato (HTTPS) e non cifrato (HTTP).

Poiché il DNS non funziona correttamente a causa di un malfunzionamento noto, si intende anche dimostrare la mancata risoluzione dei nomi.

1.2 Ambiente di laboratorio

Sistema	Ruolo	IP
Kali Linux	Server	192.168.50.100
Windows	Client	192.168.50.102

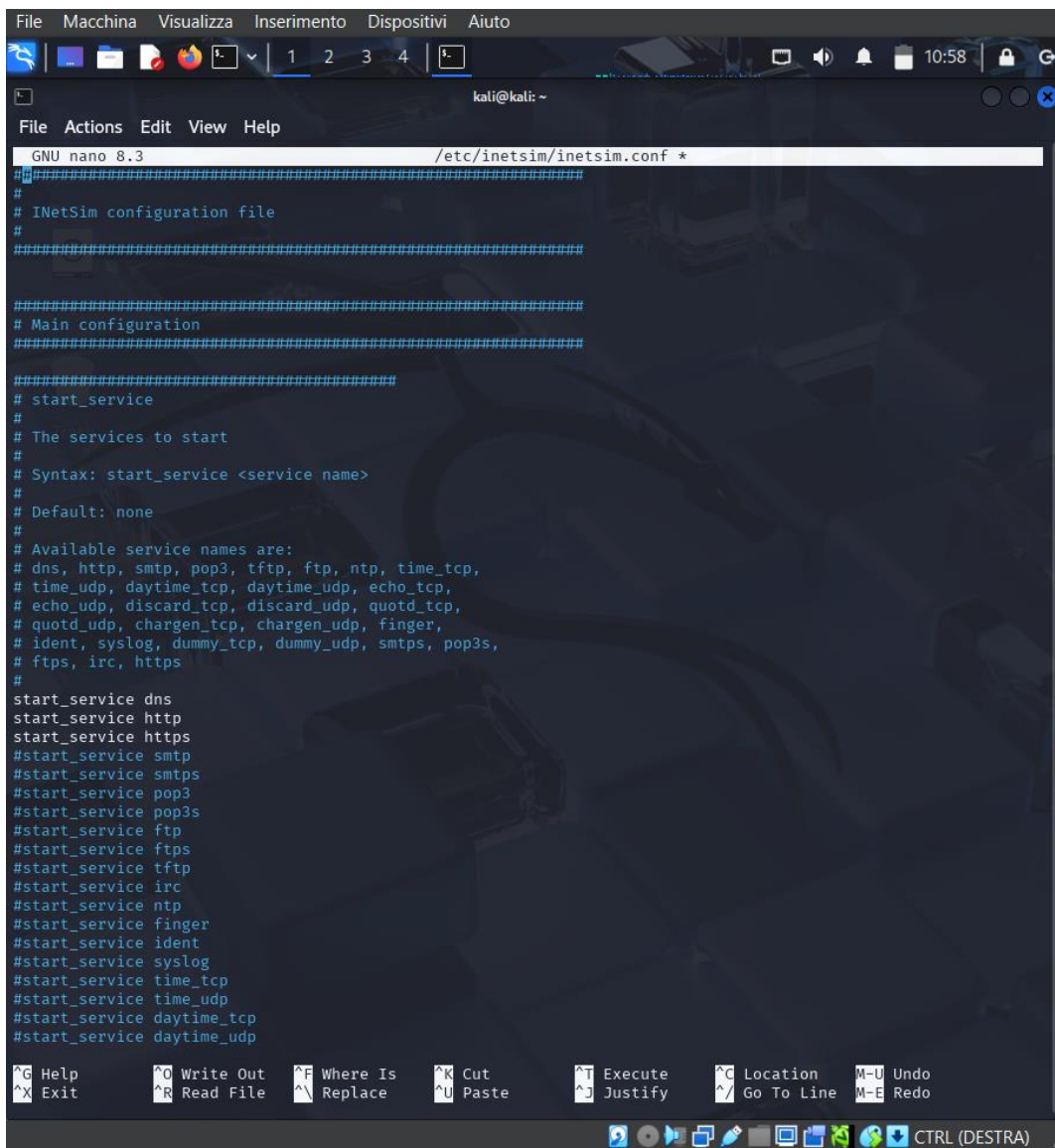
Servizi configurati su Kali Linux tramite INetSim:

- DNS
- HTTPS
- HTTP

2. Svolgimento

2.1 Configurazione HTTP, HTTPS, DNS

Sul server Kali Linux è configurato INetSim, uno strumento che simula vari servizi di rete. È stato modificato il file di configurazione con il comando "sudo /etc/inetsim/inetsim.conf" per abilitare i servizi richiesti (http, https, dns). Per abilitarli si è commentato con "#" tutte le altre voci tranne le tre che ci interessano.



The screenshot shows a Kali Linux terminal window with the nano text editor open, editing the file `/etc/inetsim/inetsim.conf`. The terminal title bar indicates the user is `kali@kali` and the time is 10:58. The nano editor's menu bar includes File, Actions, Edit, View, and Help. The file content is a configuration file for INetSim, starting with a header section followed by a main configuration section. The `start_service` section lists various services to be started, including `dns`, `http`, `https`, `smtp`, `smtps`, `pop3`, `pop3s`, `ftp`, `ftps`, `tftp`, `irc`, `ntp`, `finger`, `ident`, `syslog`, `time_tcp`, `time_udp`, `daytime_tcp`, and `daytime_udp`. The bottom of the terminal shows nano editor shortcuts and a system tray with icons for network, volume, and other utilities.

```
File  Macchina  Visualizza  Inserimento  Dispositivi  Aiuto
kali@kali: ~
File  Actions  Edit  View  Help
GNU nano 8.3 /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
#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

^G Help      ^O Write Out  ^F Where Is   ^K Cut        ^T Execute
^X Exit      ^R Read File  ^_ Replace    ^U Paste      ^J Justify
^C Location  ^-U Undo      ^M-E Redo
Go To Line

[Icons] CTRL (DESTRA)
```

Successivamente è stato configurato l'indirizzo IP a cui i servizi della macchina devono "legarsi" decommentando "#"

```
# service_bind_address
#
# IP address to bind services to
#
# Syntax: service_bind_address <IP address>
#
# Default: 127.0.0.1
#
service_bind_address 192.168.50.100
```

Per attivare il DNS (Domain Name System) è necessario impostare la porta su cui il servizio ascolta le richieste “dns_bind_port 53” e l’IP che restituisce di default nelle risposte “dns_default_ip 192.168.50.100”

```
#####  
# 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 192.168.50.100
```

Infine impostiamo il comando che serve per associare un nome host (epicode.internal) a un indirizzo IP preciso (192.168.50.100), salviamo e chiudiamo il config con ctrl X

```
#####  
# 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 192.168.50.100  
  
#####
```

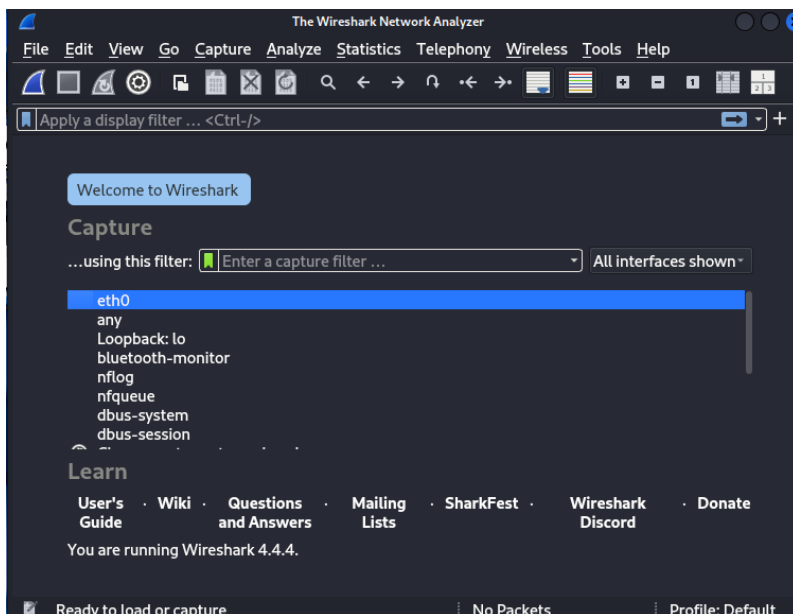
Avviamo INetSim con il comando “sudo inetsim” dal prompt Kali

```
(kali@kali)-[~]
$ sudo inetsim
INetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
Main logfile '/var/log/inetsim/main.log' does not exist. Trying to create it...
Main logfile '/var/log/inetsim/main.log' successfully created.
Sub logfile '/var/log/inetsim/service.log' does not exist. Trying to create it...
Sub logfile '/var/log/inetsim/service.log' successfully created.
Debug logfile '/var/log/inetsim/debug.log' does not exist. Trying to create it...
Debug logfile '/var/log/inetsim/debug.log' successfully created.
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 17282) ==
Session ID: 17282
Listening on: 192.168.50.100
Real Date/Time: 2025-07-19 11:42:10
Fake Date/Time: 2025-07-19 11:42:10 (Delta: 0 seconds)
Forking services ...
* dns_53_tcp_udp - started (PID 17284)
deprecatd method; prefer start_server() at /usr/share/perl5/INetSim/DNS.pm line 69.
Attempt to start Net::DNS::Nameserver in a subprocess at /usr/share/perl5/INetSim/DNS.pm line 69.
* http_80_tcp - started (PID 17285)
* https_443_tcp - started (PID 17286)
done.
Simulation running.
```

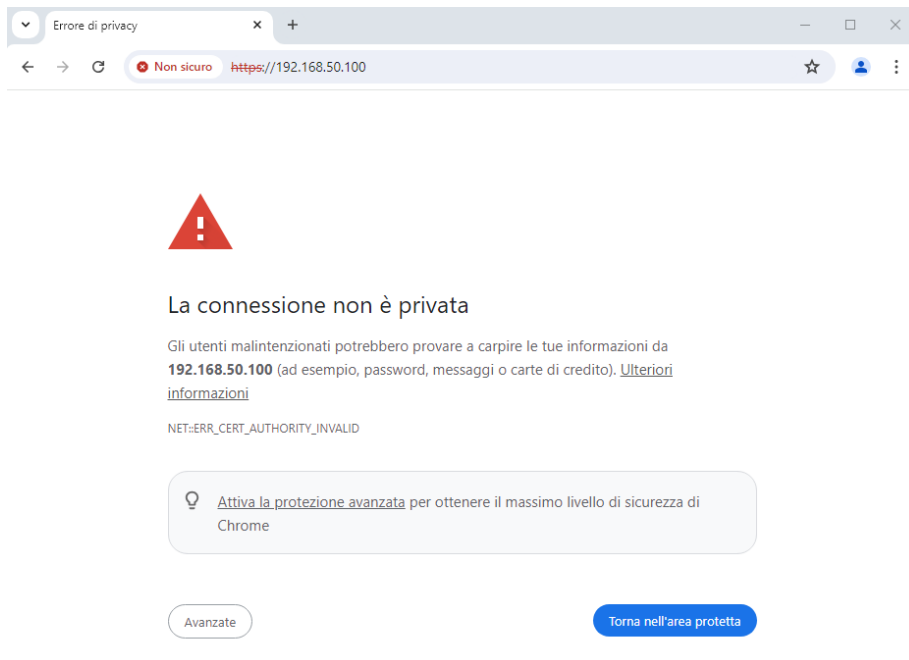
E' possibile notare che le porte 53 (DNS), 80 (HTTP) e 443 (HTTPS) siano in ascolto.

2.2 Ricerca dei pacchetti tramite Wireshark

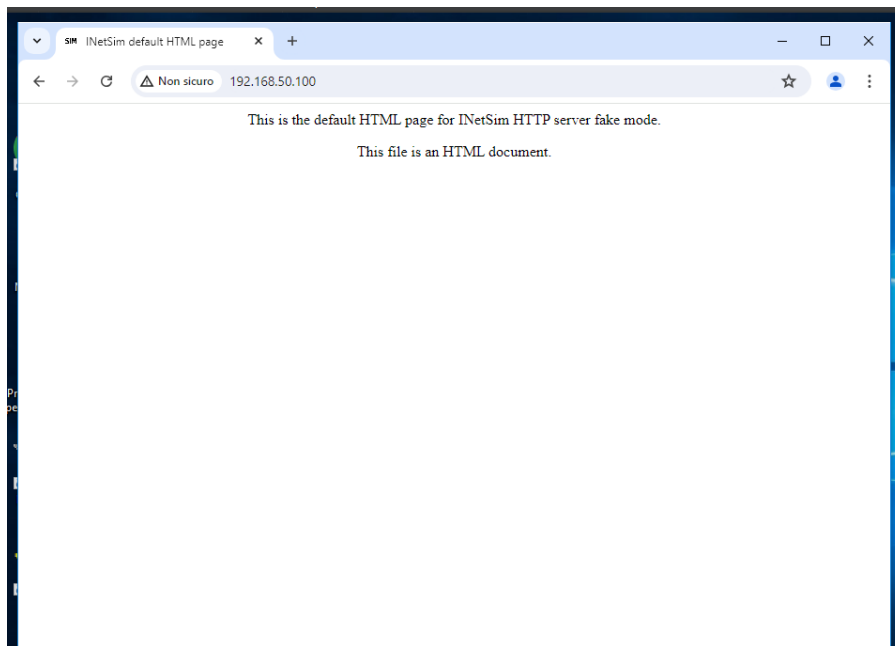
Su Kali Linux apro Wireshark, contemporaneamente al prompt con sudo inetsim, per intercettare il traffico, filtrando i pacchetti del client 192.168.50.102. Si avvia tramite l'opzione "eth0"



È stato avviato HTTPS (Figura 1) e HTTP (Figura 2) dal browser del client



(Figura 1)



(Figura 2)

Wireshark mostra la cattura dei pacchetti di rete effettuata. Il traffico proviene dal client Windows (IP 192.168.50.102) e viene inviato al server Kali Linux (IP 192.168.50.100)

The image shows a Wireshark network traffic capture. The top menu bar includes File, Macchina, Visualizza, Inserimento, Dispositivi, and Aiuto. The status bar at the top indicates 'Capturing from eth0'. The main window displays a list of captured packets with columns for No., Time, Source, Destination, Protocol, Length, and Info. The packets are filtered by 'Apply a display filter: <Ctrl-/>'. The packet list shows a sequence of TLS and HTTP packets. The packet details pane on the right shows the structure of the selected packet (No. 92), including Ethernet II, Internet Protocol Version 4, User Datagram Protocol, and NetBIOS Name Service. The packet bytes pane at the bottom shows the raw data in hexadecimal and ASCII.

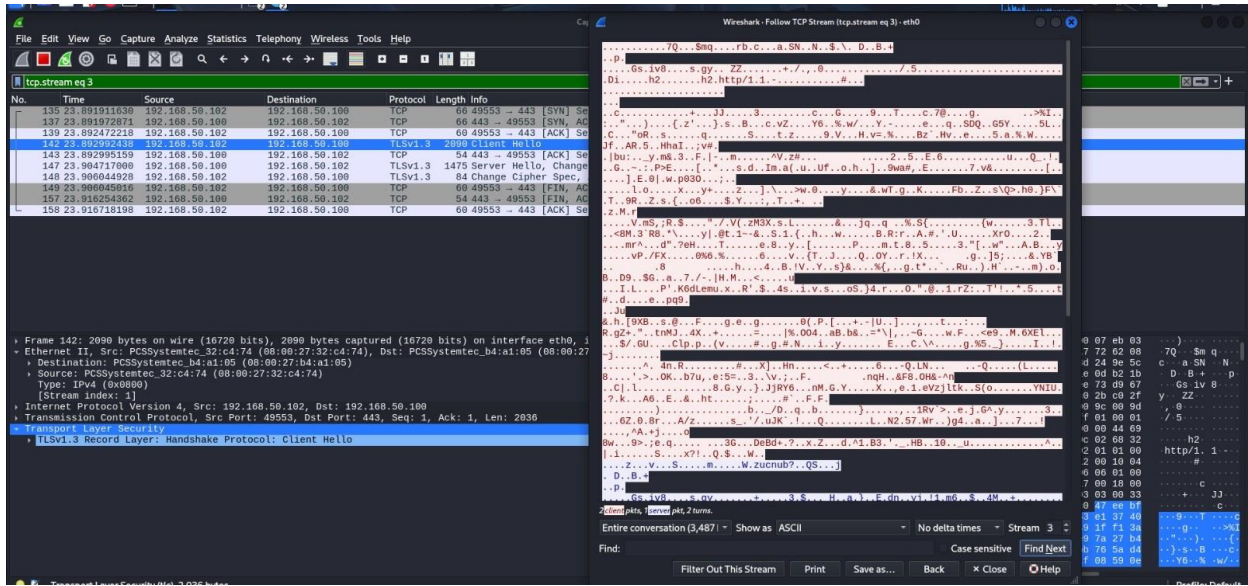
No.	Time	Source	Destination	Protocol	Length	Info
61	9.464780508	192.168.50.102	192.168.50.100	TLSv1.3	687	Application Data
62	9.464995586	192.168.50.100	192.168.50.102	TLSv1.3	389	Application Data
63	9.476493880	192.168.50.100	192.168.50.102	TLSv1.3	729	Application Data, Application Data, Application Data, Application Data
64	9.476911724	192.168.50.102	192.168.50.100	TCP	60	49479 → 443 [ACK] Seq=2782 Ack=2353 Win=65536 Len=0
65	9.477718196	192.168.50.102	192.168.50.100	TCP	60	49479 → 443 [FIN, ACK] Seq=2782 Ack=2353 Win=65536 Len=0
66	9.477731983	192.168.50.100	192.168.50.102	TCP	54	443 → 49479 [ACK] Seq=2353 Ack=2783 Win=64512 Len=0
67	13.753719934	192.168.50.102	192.168.50.100	TCP	66	49480 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM
68	13.753748610	192.168.50.100	192.168.50.102	TCP	66	80 → 49480 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=
69	13.754258479	192.168.50.102	192.168.50.100	TCP	66	49481 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM
70	13.754258688	192.168.50.102	192.168.50.100	TCP	60	49480 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
71	13.754278353	192.168.50.100	192.168.50.102	TCP	66	80 → 49481 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=
72	13.754660847	192.168.50.102	192.168.50.100	HTTP	530	GET / HTTP/1.1
73	13.754861027	192.168.50.102	192.168.50.100	TCP	60	49481 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
74	13.754871532	192.168.50.100	192.168.50.102	TCP	54	80 → 49480 [ACK] Seq=1 Ack=477 Win=64128 Len=0

Frame 1: 92 bytes on wire (736 bits), 92 bytes captured (736 bits) on interface eth0, id 0
Ethernet II, Src: PCSSystemtec_64:5e:39 (08:00:27:64:5e:39), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol Version 4, Src: 192.168.50.102, Dst: 192.168.50.255
User Datagram Protocol, Src Port: 137, Dst Port: 137
NetBIOS Name Service

eth0: <live capture in progress> Packets: 116 Profile: Default

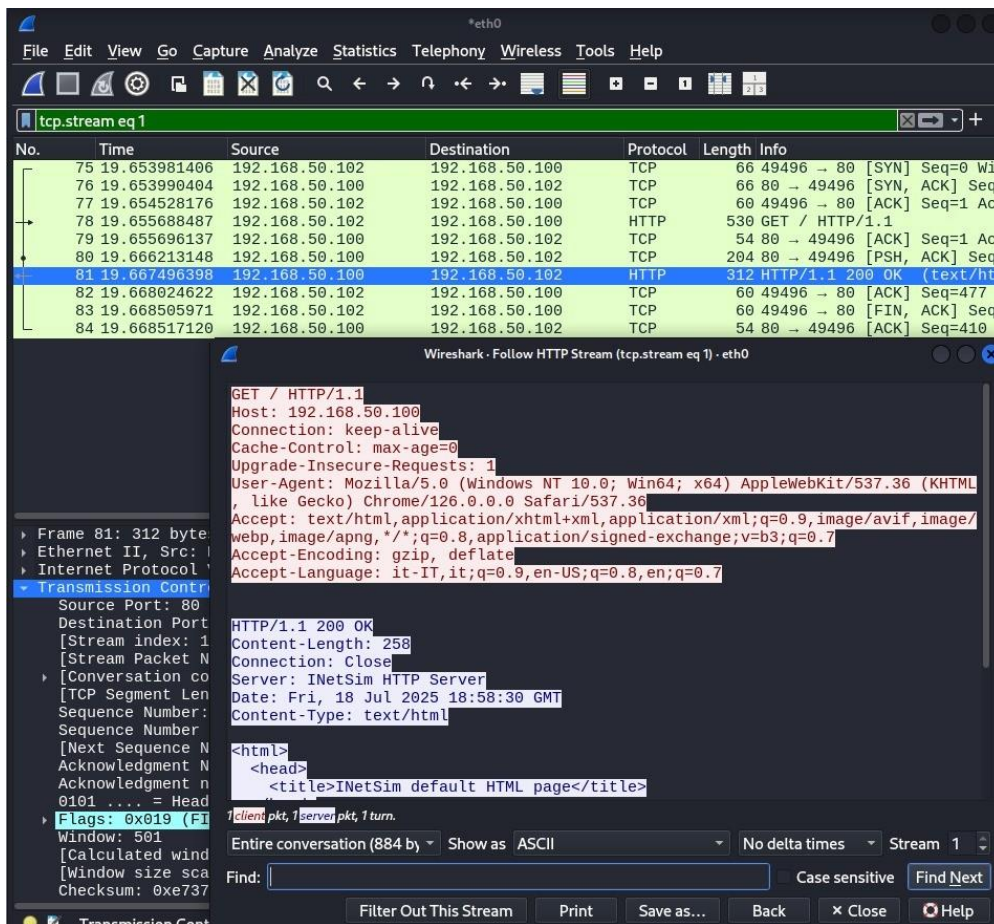
2.3 Analisi dei pacchetti

Nell'immagine seguente mostro la cattura di pacchetti in HTTPS:



- Pacchetti da 192.168.50.102 a 192.168.50.100 sulla porta 443.
- Protocollo: TLSv1.3.
- Contenuto della comunicazione cifrato, come tipico di HTTPS.

In questa seconda cattura si osserva il traffico HTTP:



- Pacchetti da 192.168.50.102 a 192.168.50.100 sulla porta 80.
- Protocollo: HTTP.
- Richiesta GET / HTTP/1.1 visibile in chiaro nel campo Info.

Nel dettaglio sono evidenziati anche i MAC:

- MAC address sorgente: 08:00:27:64:5e:39 (client).
- MAC address destinazione: 08:00:27:b4:a1:05 (server).
- Questo conferma l'identificazione degli host e la presenza di pacchetti broadcast tipici della rete locale.

2.4 Differenze tra HTTPS e HTTP

Quando si osservano le comunicazioni di rete in un laboratorio con Wireshark, è immediatamente evidente la differenza tra il traffico generato da una connessione HTTP e quello generato da una connessione HTTPS. Queste differenze sono sia tecniche che funzionali.

HyperText Transfer Protocol

HTTP è un protocollo di livello applicativo che consente la trasmissione di dati (pagine web, immagini, file) tra un client (browser) e un server web. Lavora sulla porta 80 (per default). Non utilizza alcuna forma di cifratura o autenticazione: i dati viaggiano in chiaro.

Questo significa che:

Tutto il contenuto della richiesta e della risposta (es. URL richiesto, intestazioni, cookies, dati POST) è leggibile da chiunque intercetti i pacchetti lungo il percorso. È vulnerabile ad attacchi.

Su Wireshark, i pacchetti HTTP appaiono con Protocol "HTTP" e il contenuto delle richieste e risposte è visibile nel pannello inferiore: è possibile leggere la richiesta GET / HTTP/1.1 o i dati inviati in un form.

HyperText Transfer Protocol Secure

HTTPS è una versione sicura di HTTP, che aggiunge un livello di cifratura usando il protocollo TLS (Transport Layer Security) o SSL (Secure Socket Layer). Lavora sulla porta **443** (per default). I dati vengono cifrati end-to-end tra client e server.

Questo garantisce:

Confidenzialità: un eventuale intercettore non può leggere i dati.

Integrità: eventuali modifiche ai dati in transito vengono rilevate.

Autenticità: il client può verificare l'identità del server tramite certificati digitali.

Su Wireshark, i pacchetti appaiono con Protocol: TLSv1.2 o TLSv1.3.

Il contenuto dei dati applicativi è **illeggibile**, perché cifrato.

3 Conclusioni

L'esercizio svolto ha permesso di simulare e analizzare un'architettura client-server in un ambiente di laboratorio virtuale, con un client Windows che richiede una risorsa web al server Kali Linux.

Tramite la configurazione dei servizi HTTP e HTTPS su Kali, e l'uso di Wireshark per l'intercettazione del traffico, è stato possibile evidenziare le differenze sostanziali tra le due tipologie di comunicazione:

- Nel caso della connessione HTTPS, i pacchetti catturati mostrano traffico cifrato, rendendo il contenuto della richiesta e della risposta non leggibile, a garanzia della riservatezza della comunicazione.
- Nel caso della connessione HTTP, i pacchetti trasportano i dati in chiaro, rendendo possibile l'ispezione diretta del contenuto della richiesta da parte di un eventuale intercettatore.

Inoltre, grazie a Wireshark è stato possibile identificare gli indirizzi MAC sorgente e destinazione dei pacchetti, verificando così il corretto instradamento del traffico all'interno della rete locale.

L'esercitazione ha quindi dimostrato in modo pratico l'importanza dell'uso di HTTPS per la protezione dei dati trasmessi su una rete e ha fornito un esempio concreto di analisi del traffico di rete in un ambiente controllato.