

CONSEGNA W4D4

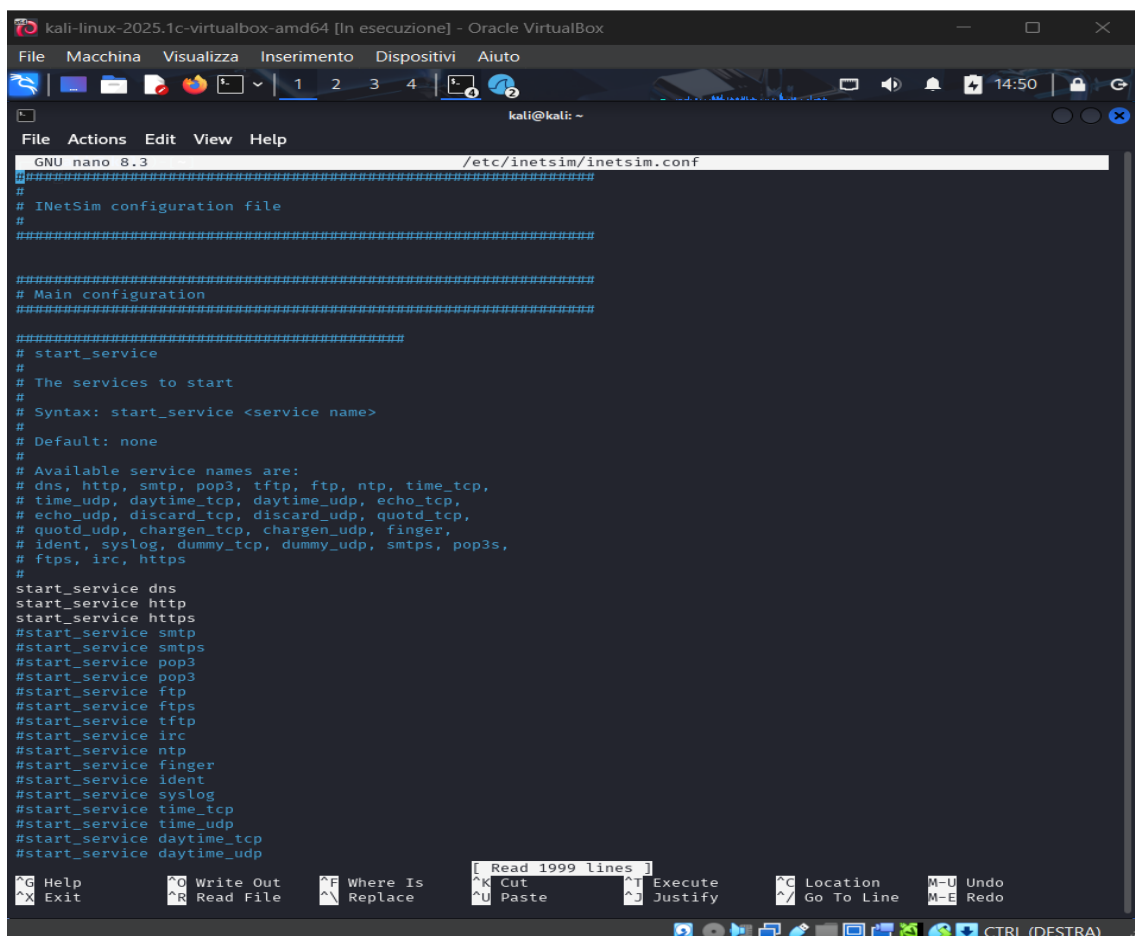
Nell'esercitazione in questione proviamo a simulare, nel nostro laboratorio virtuale, una comunicazione tra client windows che tramite web browser richiede una risorsa che risponde all'indirizzo di una kali.

1) CONFIGURAZIONE AMBIENTE DI LAVORO

Nel nostro caso abbiamo usato nell'ambiente sicuro creato in precedenza con 2 VM, una window con ip 192.168.50.102 e una Kali 192.168.50.100 con un server dns impostato su 192.168.50.1.

2) CONFIGURAZIONE DNS, HTTP, HTTPS

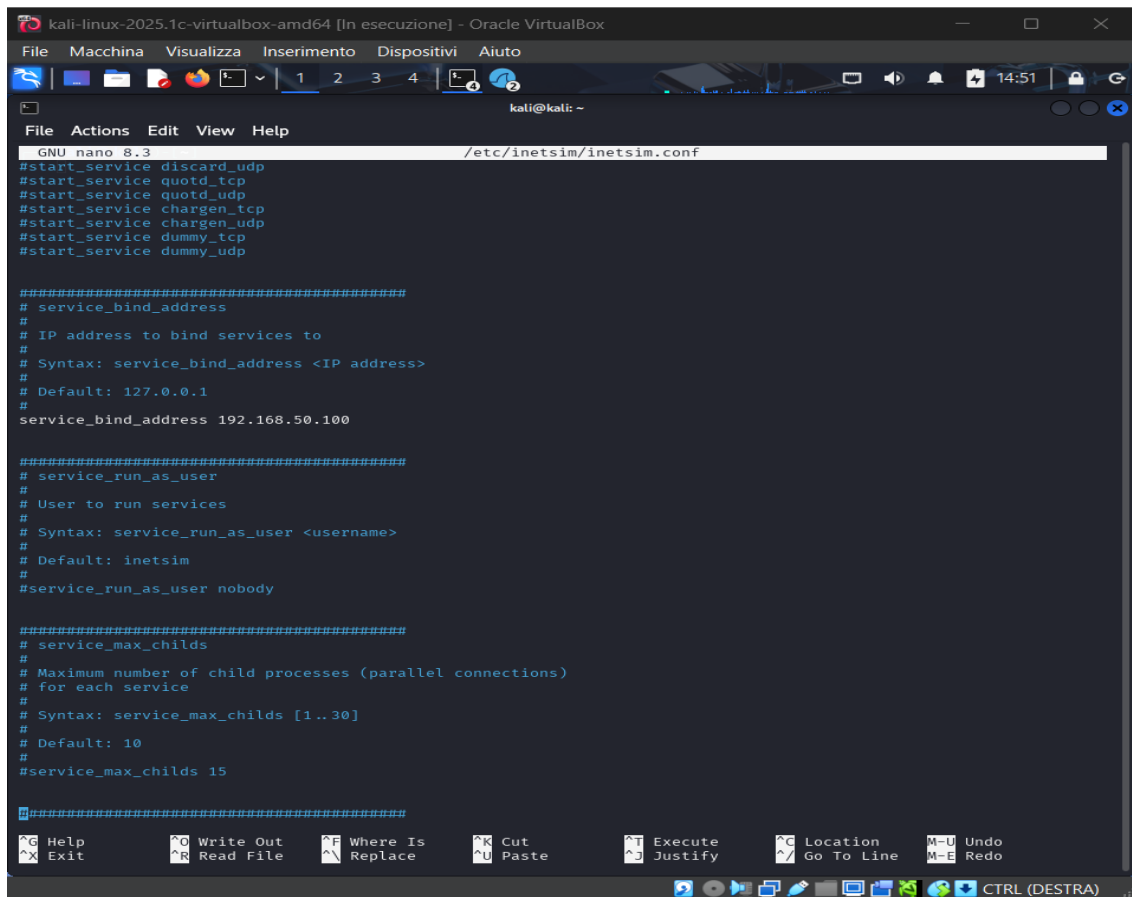
Successivamente ho lanciato su kali il comando “sudo nano /etc/inetsim/inetsim.conf” al fine di configurare i servizi da simulare. A tal proposito andiamo a commentare (mettere l’“#”) tutte le voci, tranne “start_service dns”, “start_service http” e “start_service https” come mostrato nella figura 1, per far sì che tali righe non venga ignorate da Wireshark in seguito.



```
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 pop3
#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
```

Figura 1

In seguito, andiamo, seguendo lo stesso ragionamento, a decommentare la voce “services_bind_address” e a modificarne l’ip di riferimento con l’ip 192.168.50.100 come nella figura 2. Questo servizio è un parametro di configurazione che indica a quale ip un servizio deve “legarsi” per accettare connessioni in ingresso. In pratica, dice al servizio su quale interfaccia di rete deve ascoltare.



```
kali-linux-2025.1c-virtualbox-amd64 [In esecuzione] - Oracle VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
kali@kali: ~
GNU nano 8.3 /etc/inetsim/inetsim.conf
#start_service discard_udp
#start_service quotd_tcp
#start_service quotd_udp
#start_service chargen_tcp
#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 192.168.50.100

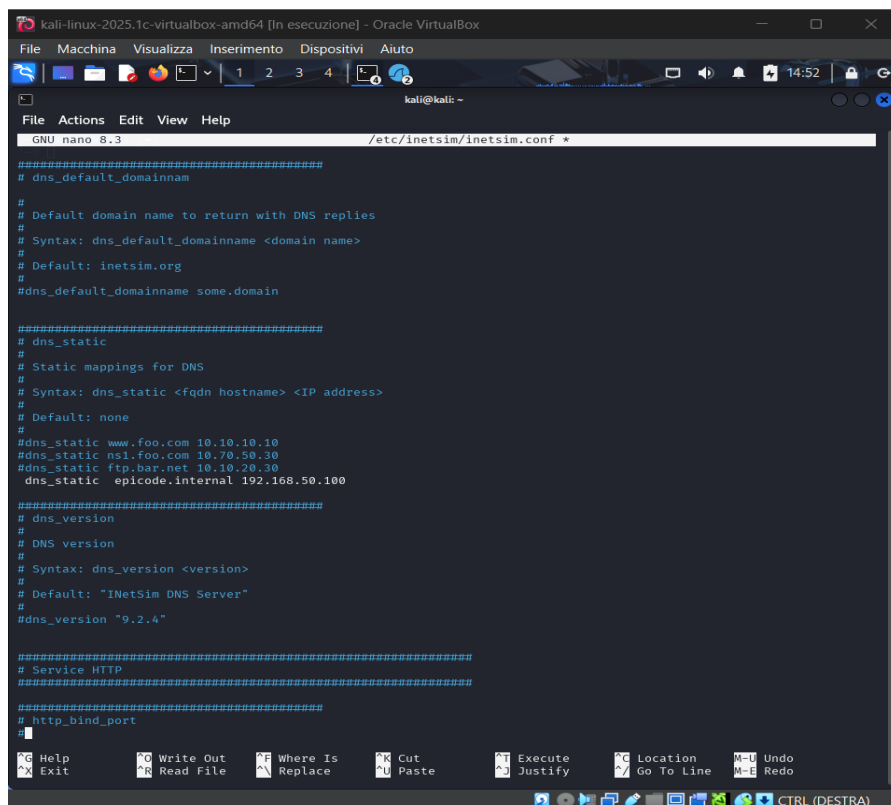
#####
# service_run_as_user
#
# User to run services
#
# Syntax: service_run_as_user <username>
#
# Default: inetsim
#
#service_run_as_user nobody

#####
# service_max_childs
#
# Maximum number of child processes (parallel connections)
# for each service
#
# Syntax: service_max_childs [1..30]
#
# Default: 10
#
#service_max_childs 15

#####
^G Help      ^O Write Out  ^F Where Is   ^K Cut        ^I Execute    ^C Location   M-U Undo
^X Exit      ^R Read File  ^A Replace    ^U Paste      ^T Justify    ^G Go To Line M-E Redo
CTRL (DESTRA)
```

Figura 2

Andiamo a modificare la voce Dns_static, aggiungendo una nuova voce senza commento, come nella figura 3, “dns_static epicode.internal 192.168.50.100”. Questa operazione ha lo scopo di definire una corrispondenza tra un nome di dominio e un indirizzo IP.



```
kali@kali: ~  
File Actions Edit View Help  
GNU nano 8.3 /etc/inetsim/inetsim.conf *  
#####  
# dns_default_domainname  
#  
# Default domain name to return with DNS replies  
#  
# Syntax: dns_default_domainname <domain name>  
#  
# Default: inetsim.org  
#  
#dns_default_domainname some.domain  
#####  
# 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  
#####  
# dns_version  
#  
# DNS version  
#  
# Syntax: dns_version <version>  
#  
# Default: "INetSim DNS Server"  
#  
#dns_version "9.2.4"  
#####  
# Service HTTP  
#####  
# http_bind_port  
#
```

Figura 3

3) ANALISI CON WIRESHARK

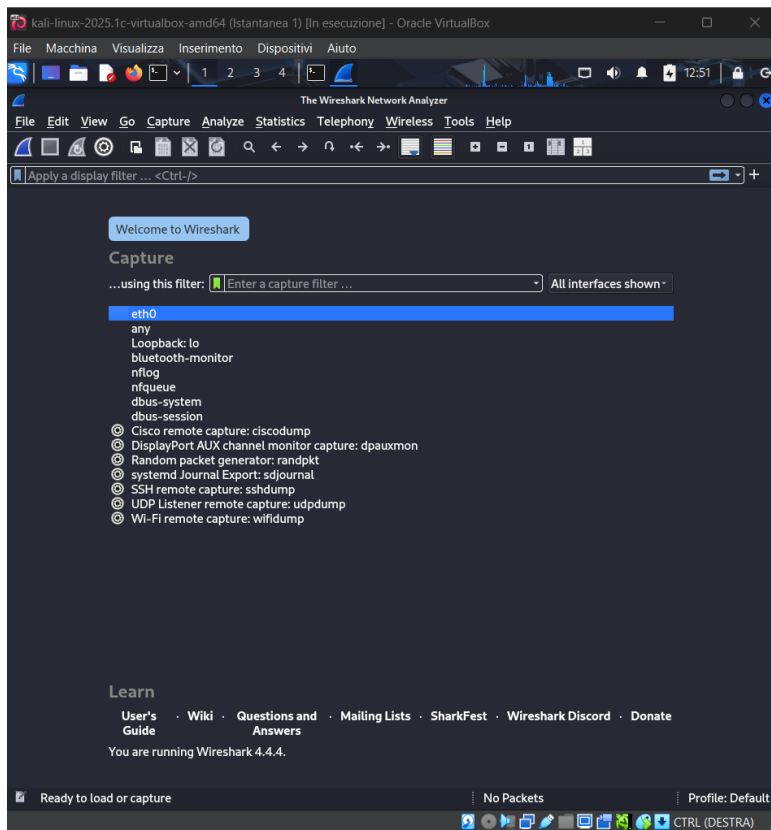
Dopodiché avviamo il comando “sudo inetsim” per avviare una simulazione di servizi internet in un’ambiente totalmente sicuro. In sede di call con il docente abbiamo appurato però la mancanza di alcuni pacchetti nell’Inetsim che ci precludevano la possibilità di accedere tramite altra macchina sul browser con la ricerca “epicode.internal” ma solo tramite l’utilizzo di IP. Alla luce di ciò nei prossimi screenshot vedremo le ricerche svolte tramite indirizzi ip. Nella figura seguente ho evidenziato la riga che ci mostra proprio quanto detto in precedenza. Quello evidenziato è un messaggio tecnico di log che documenta il tentativo di avvio del modulo DNS da parte di INetSim.

```
kali-linux-2025.1c-virtualbox-amd64 [In esecuzione] - Oracle VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
1 2 3 4
kali@kali: ~
File Actions Edit View Help
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 50011) ==
Session ID: 50011
Listening on: 127.0.0.1
Real Date/Time: 2025-07-18 14:41:56
Fake Date/Time: 2025-07-18 14:41:56 (Delta: 0 seconds)
Forking services ...
* dns_53_tcp_udp - started (PID 50013)
deprecated 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.
* https_443_tcp - started (PID 50014)
done.
Simulation running.
^C * https_443_tcp - stopped (PID 50014)
* https_443_tcp - stopped (PID 50014)
Simulation stopped.
== INetSim main process stopped (PID 50011) ==
.

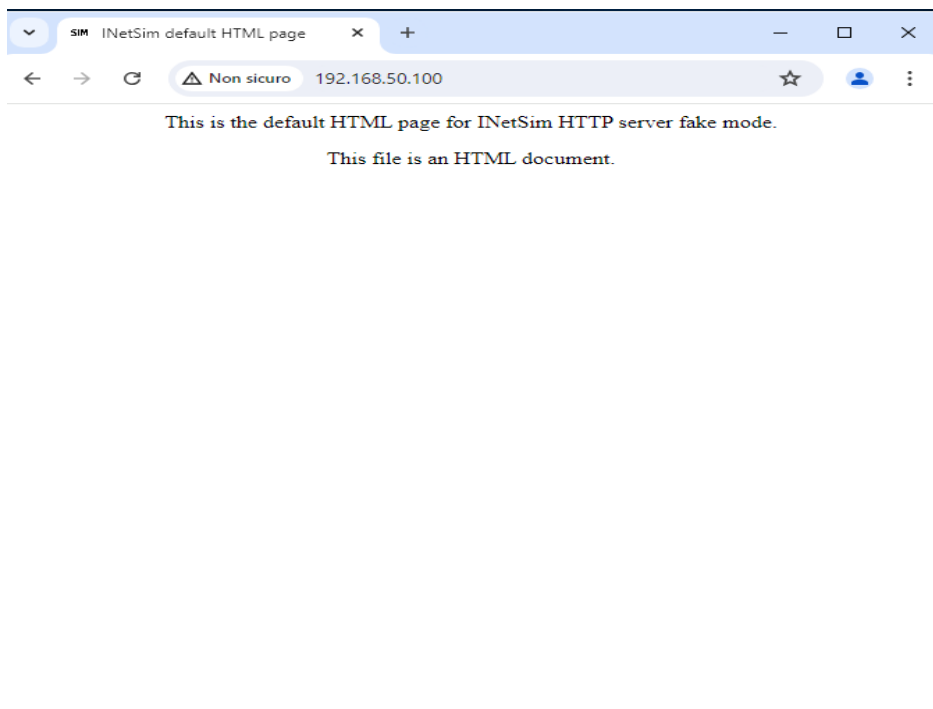
(kali@kali)-[~]
$ sudo inetsim
4INetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
Using log directory: /var/log/inetsim/
Using data directory: /var/lib/inetsim/
Using report directory: /var/log/inetsim/report/
Using configuration file: /etc/inetsim/inetsim.conf
Parsing configuration file.
Configuration file parsed successfully.
== INetSim main process started (PID 50310) ==
Session ID: 50310
Listening on: 192.168.50.100
Real Date/Time: 2025-07-18 14:42:33
Fake Date/Time: 2025-07-18 14:42:33 (Delta: 0 seconds)
Forking services ...
* dns_53_tcp_udp - started (PID 50312)
deprecated 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 50313)
* https_443_tcp - started (PID 50314)
done.
Simulation running.
```

Figura 4

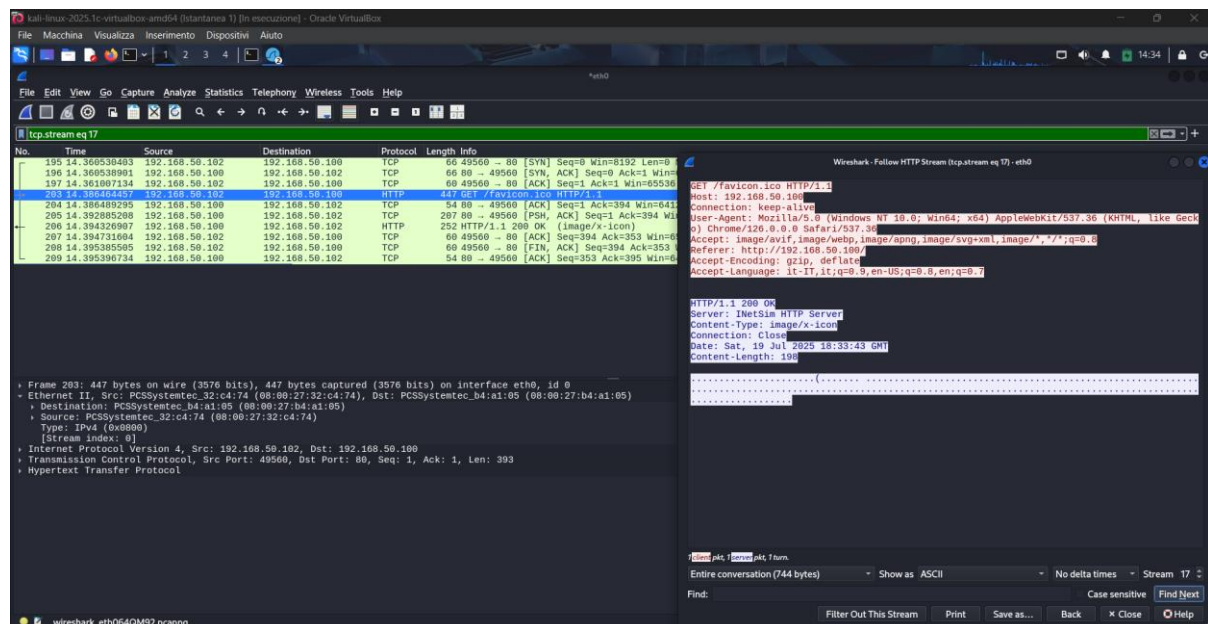
In seguito, avviamo Wireshark per analizzare il traffico in tempo reale. In questo modo riusciremo a catturare i pacchetti e analizzarne i protocolli utilizzati. Nell'esercitazione in questione noi imposteremo questo software in ascolto sulla linea "eth0"



Per poi accedere alla VM di windows, aprire il browser e cercare l'IP del dns che abbiamo configurato in precedenza, <http://192.168.50.100> e vedremo se ci sarà prima la risposta sul browser poi cosa ha intercettato wireshark.



In questo screenshot mostro come Wireshark ha catturato i pacchetti nel momento in cui ci siamo connessi al server 192.168.50.100 in http. Innanzi tutto, ho filtrato i risultati inserendo nella barra “tcp.port==80”, poi ho evidenziato le due righe di request e reply, e messo in evidenza nella parte inferiore della schermata la conversazione avvenuta tra le macchine, dove in rosso sono evidenziati i dati inviati dal client e in blu quelli inviati dal server.



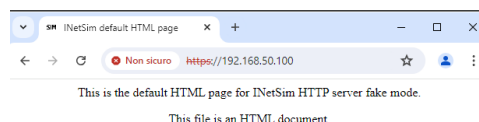
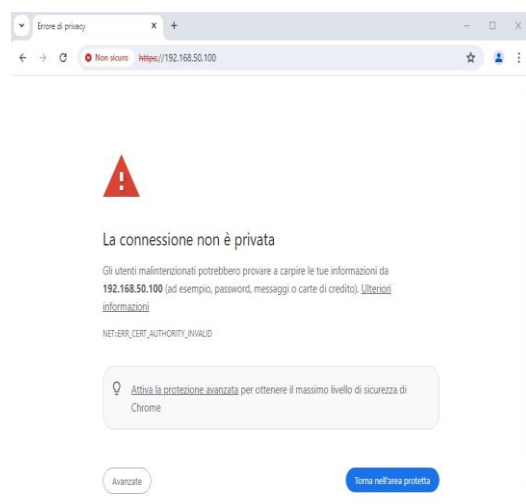
La parte di Ethernet II ci rivela:

Destination: 08:00:27:9f:f0:91 (MAC address del server Kali Linux 192.168.50.100)

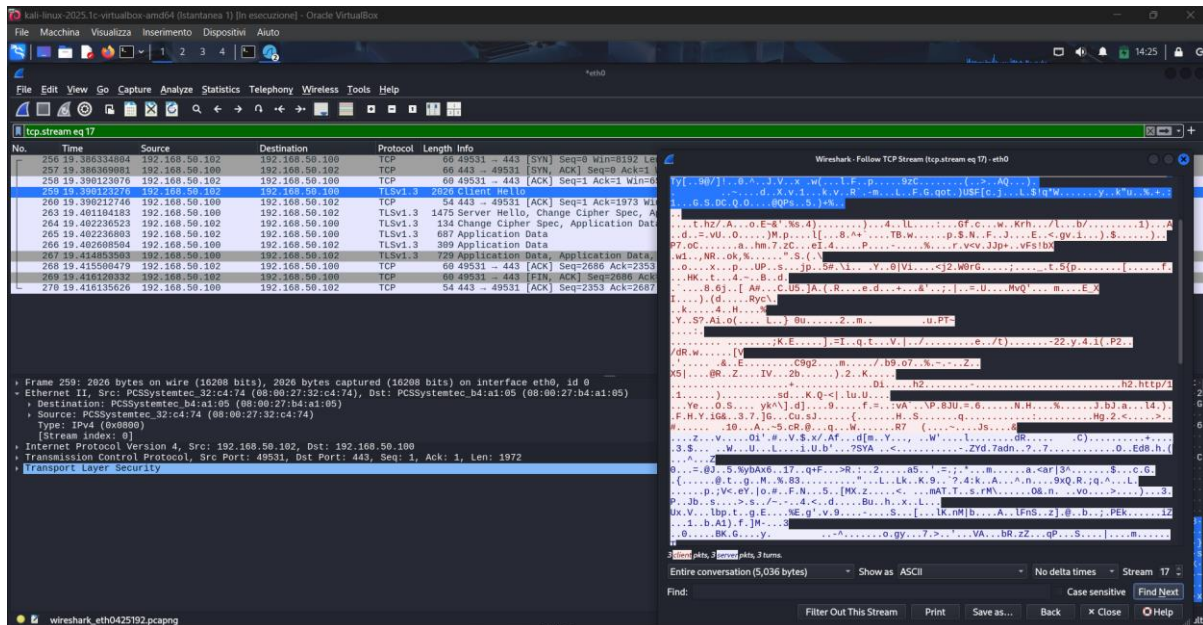
Source: 00:0c:29:d8:5e:6a (MAC address del client Windows 192.168.50.102)

Il pacchetto parte dal client Windows e arriva al server Kali.

Nella seconda parte di raccolta di dati su Wireshark andiamo ad analizzare invece lo scambio di pacchetti che avviene quando navighiamo in https. Come in precedenza andiamo a cercare sul browser della macchina Windows 192.168.50.100 in https. Nella schermata successiva mostro il risultato.



Wireshark, come in precedenza, ha catturato i pacchetti nel momento in cui ci siamo connessi al server 192.168.50.100 in https. Anche in questo caso per “pulizia” ho filtrato i risultati inserendo nella barra “tcp.port==443”. Anche in questo caso scegliendo un pacchetto e seguendolo troveremo la conversazione tra la macchina e il server, anche se in questo caso appare indecifrabile perché viene crittografato da TLS. Infatti, è importante sapere che quando andiamo a cercare i pacchetti in https, troveremo questi ultimi in formato TLS in quanto di per sé https è un http che viaggia in modo sicuro protetto da tls.



Nella sezione Ethernet II, possiamo leggere:

Destination: 08:00:27:9f:f0:91 (MAC del server Kali Linux 192.168.50.100)

Source: 00:0c:29:d8:5e:6a (MAC del client Windows 192.168.50.102)

Il pacchetto è stato inviato dal client Windows ed è diretto al server Kali.

Difatti questa esercitazione ci ha permesso di vedere le differenze che intercorrono tra i protocolli. Notiamo come http trasmette i dati in chiaro, quindi chiunque riesca ad intercettare il pacchetto in questione riesce a leggere e modificare le informazioni e usa la porta 80; mentre Https usa la porta 443 e usando, come detto in precedenza TLS, crittografa i dati trasmessi proteggendoli da intercettazioni e modifiche esterne.