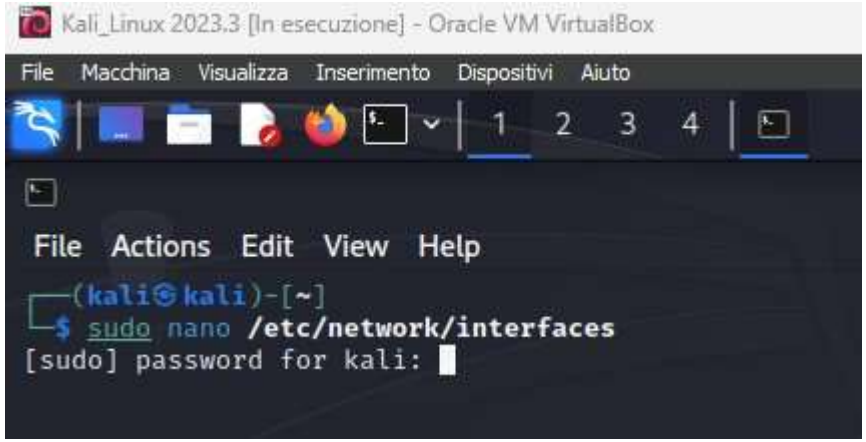
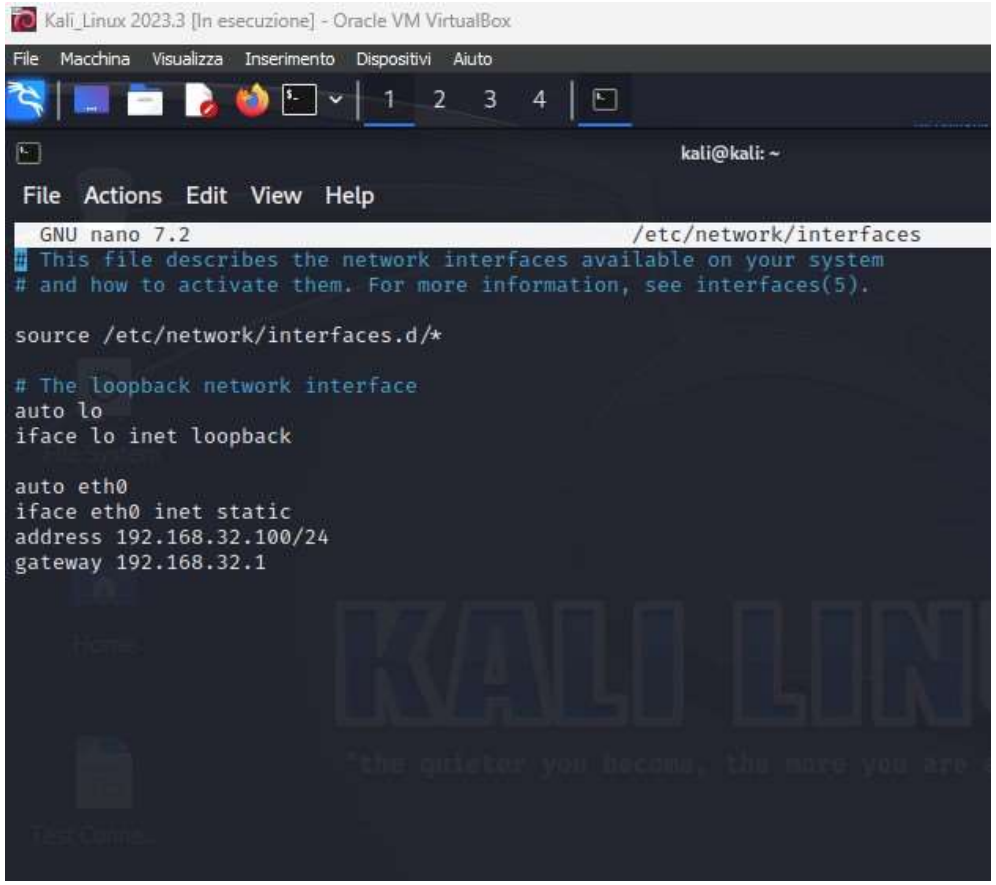
ESERCITAZIONE WEEK 4 DAY 5

[1] CONFIGURAZIONE MACCHINE WINDOWS 7 E KALI-LINUX.

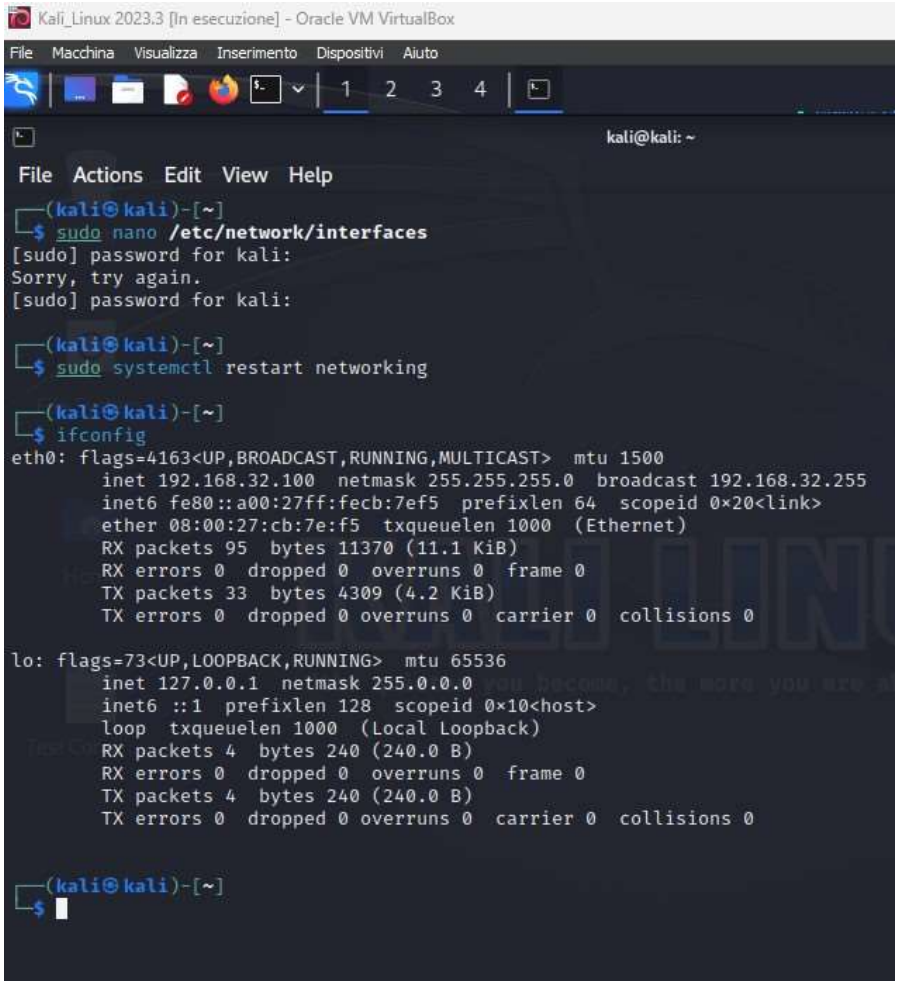
Parto configurando l'indirizzo IP 192.168.32.100/24 sulla macchina Kali-Linux, come Gateway 192.168.32.1. Avvio la macchina virtuale di Kali-Linux ed entro nel Command Line. Digito il comando **sudo nano /etc/network/interfaces** che mi permette di accedere al file contenente la configurazione della scheda di rete presente. In **sudo** perché senza questo comando che mi permette di accedere come SuperUser non potrei salvare le eventuali modifiche apportate nel file.



Una volta dentro, modifico la scheda di rete **eth0**, essere la scheda principale di rete presente in Kali-Linux, impostando l'indirizzo IP prestabilito e la sua relativa subnet: 192.168.32.100/24 (/24 sta per 255.255.255.0 scritto per esteso). Come indirizzo di Gateway l'indirizzo IP 192.168.32.1. Dopo di che termino salvo il file interfaces.



Per confermare che l'indirizzo IP configurato sia corretto, una volta chiuso il file e salvato correttamente, avvio un comando di riavvio della rete, ovvero **sudo systemctl restart networking.** Poi lancio il comando **ifconfig** che mi permette di vedere la configurazione di ogni singola scheda di rete presente nella macchina, fornendomi IP, MAC, Netmask, broadcast etc… Con questo verifico che sia correttamente configurata.

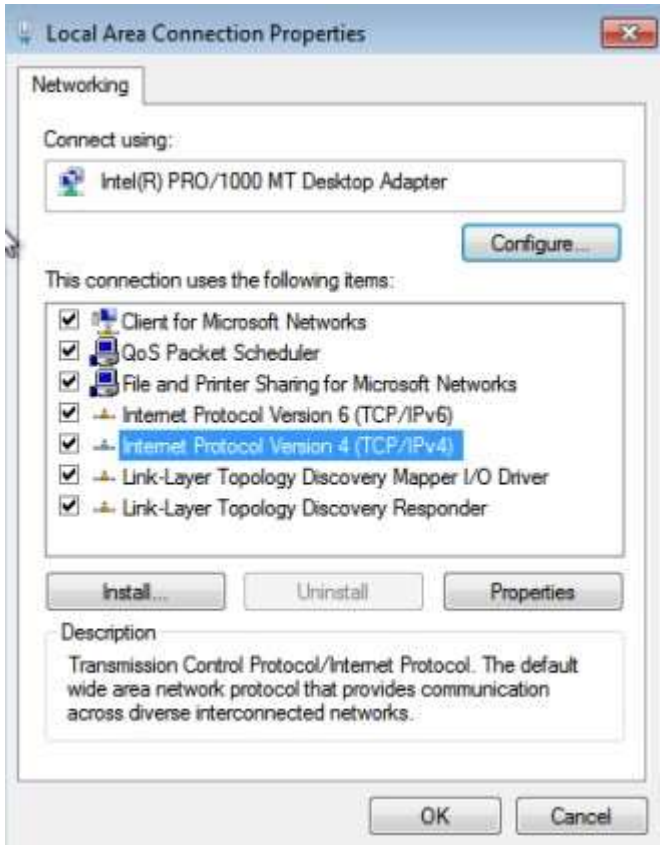


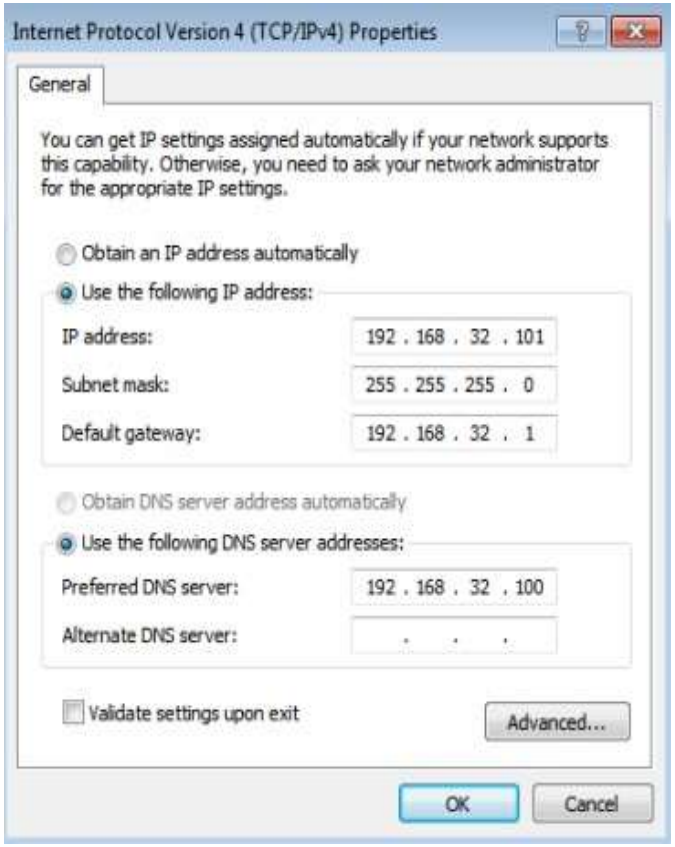
Un volta ultimata questa configurazione, salvo il file di inetsim.conf e riavvio il servizio di rete.

Da qui passo alla configurazione della scheda di rete di Windows 7 impostando come ip il **192.168.32.101** con subnet **/24** (per esteso 255.255.255.0), come il gateway **192.168.32.1** e come indirizzo DNS il **192.168.32.100**.

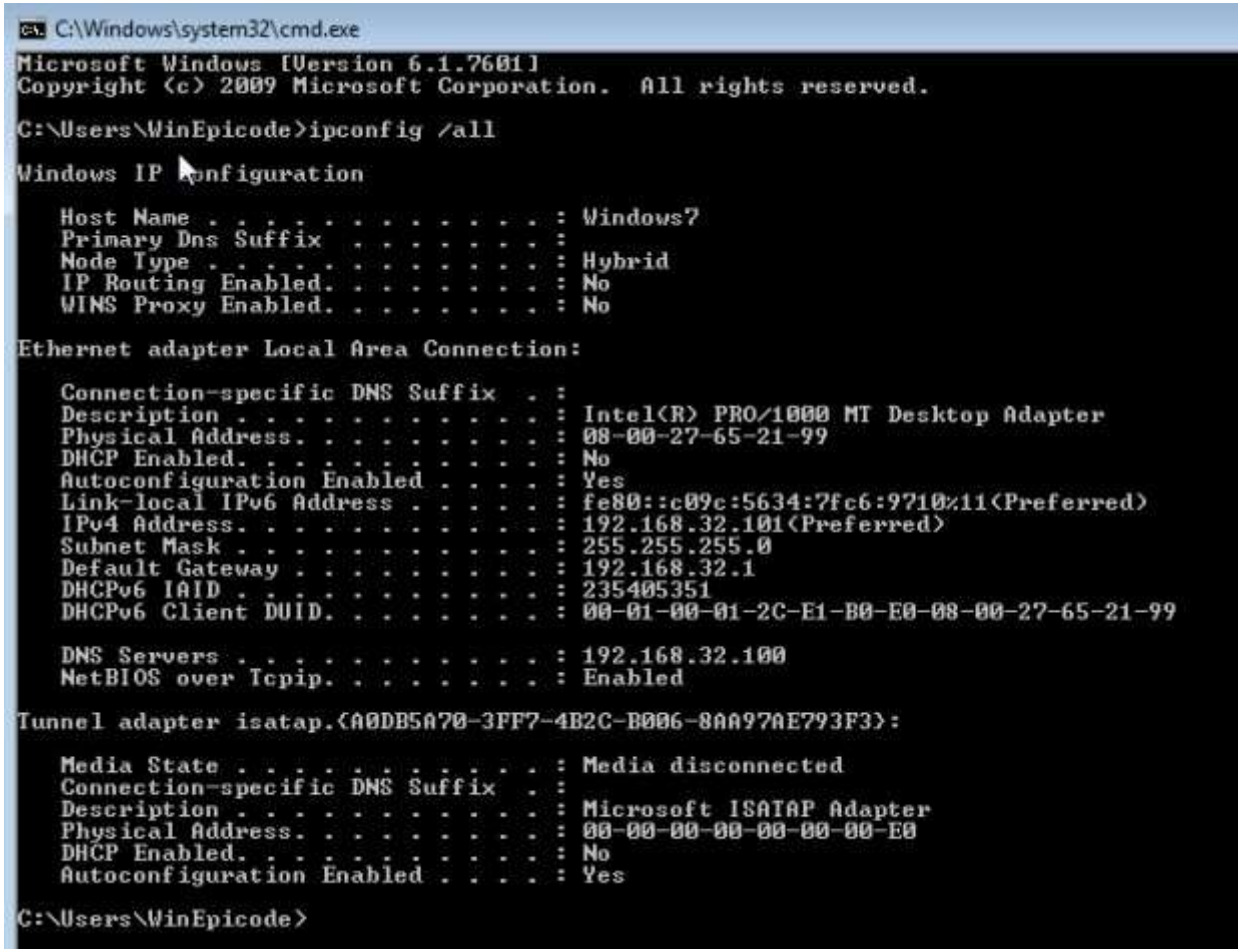
Per accedere alla configurazione della scheda in Windows, accedo su Controll Panel – Network and internet – Network and Sharing Center – Change adapter settings e seleziono l'unica scheda disponibile con tasto destro e "Properties"

Seleziono Internet protocol Version 4 e ne cambio gli indirizzi come indicato e premo "Apply".





Verifico poi tramite il Prompt dei comandi e il comando **ipconfig /all** se le configurazioni sono state applicate con successo.

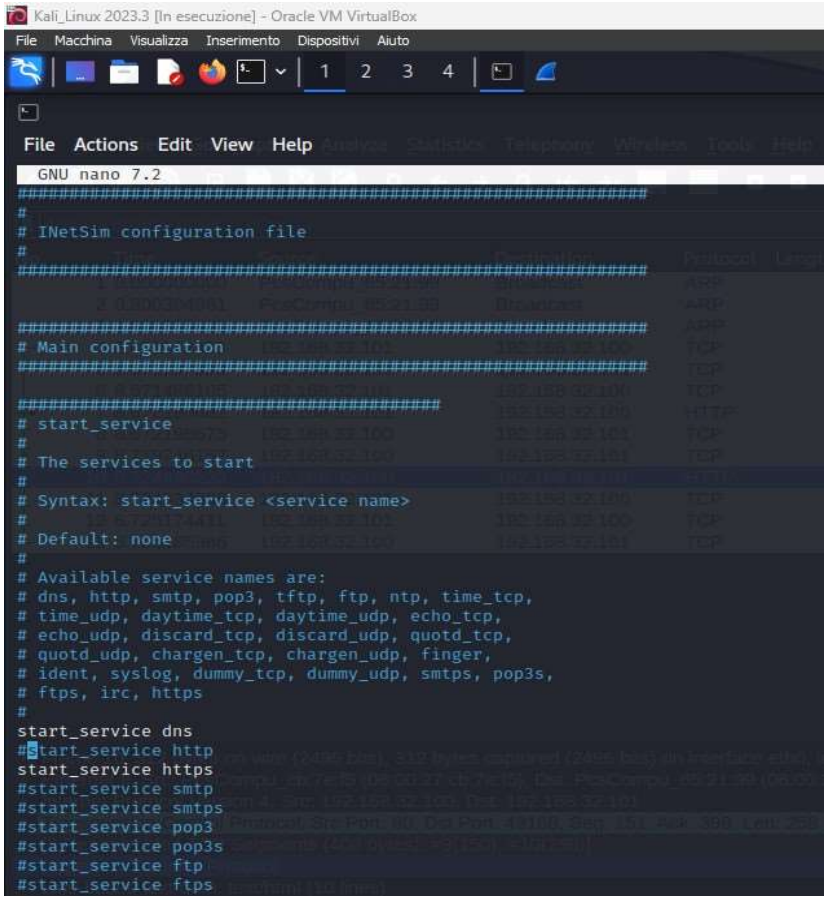


[2] SIMULAZIONE DI LABORATORIO VIRTUALE DI UN'ARCHITETTURA CLIENT-SERVER

Una volta configurate le due macchine correttamente, passo alla richiesta successiva. Windows 7 richiede tramite web browser una risorsa all'hostname epicode.internal che risponde all'indirizzo 192.168.32.100, ovvero Kali-linux.

Partendo da quest'ultima, configuro su i servizi di rete richiesti dalla traccia, ovvero attivare il **servizio HTTPS** server e **servizio DNS.** Per abilitare questi servizi sulla macchina Kali-Linux, mi avvalgo di una funzione chiamata **InetSim.** Questa mi permette di simulare i servizi che un Server generico può offrire alla rete e configurandone li stessi. Per accedervi, entro nella cartella in cui è contenuto il file **inetsim.conf**, ovvero **cd /etc/inetsim/.** Una volta dentro, digito il comando **sudo nano inetsim.conf** per accedere al file di configurazione.

Così facendo, accedo alla configurazione del file che mi permetterà di simulare i servizi di rete richiesti. Per farlo, mi basti rimuovere il simbolo **#** nelle voci di **start_service https** e **start_service dns.** Così facendo sono configurati come attivi.



Scorrendo giù nella pagina si può accedere alle configurazioni di ogni servizio presente nel file.

Accedendo al **service_bind_address** lo abilitiamo e indichiamo come indirizzo il 0.0.0.0, ovvero abilitando i servizi su tutte le schede.

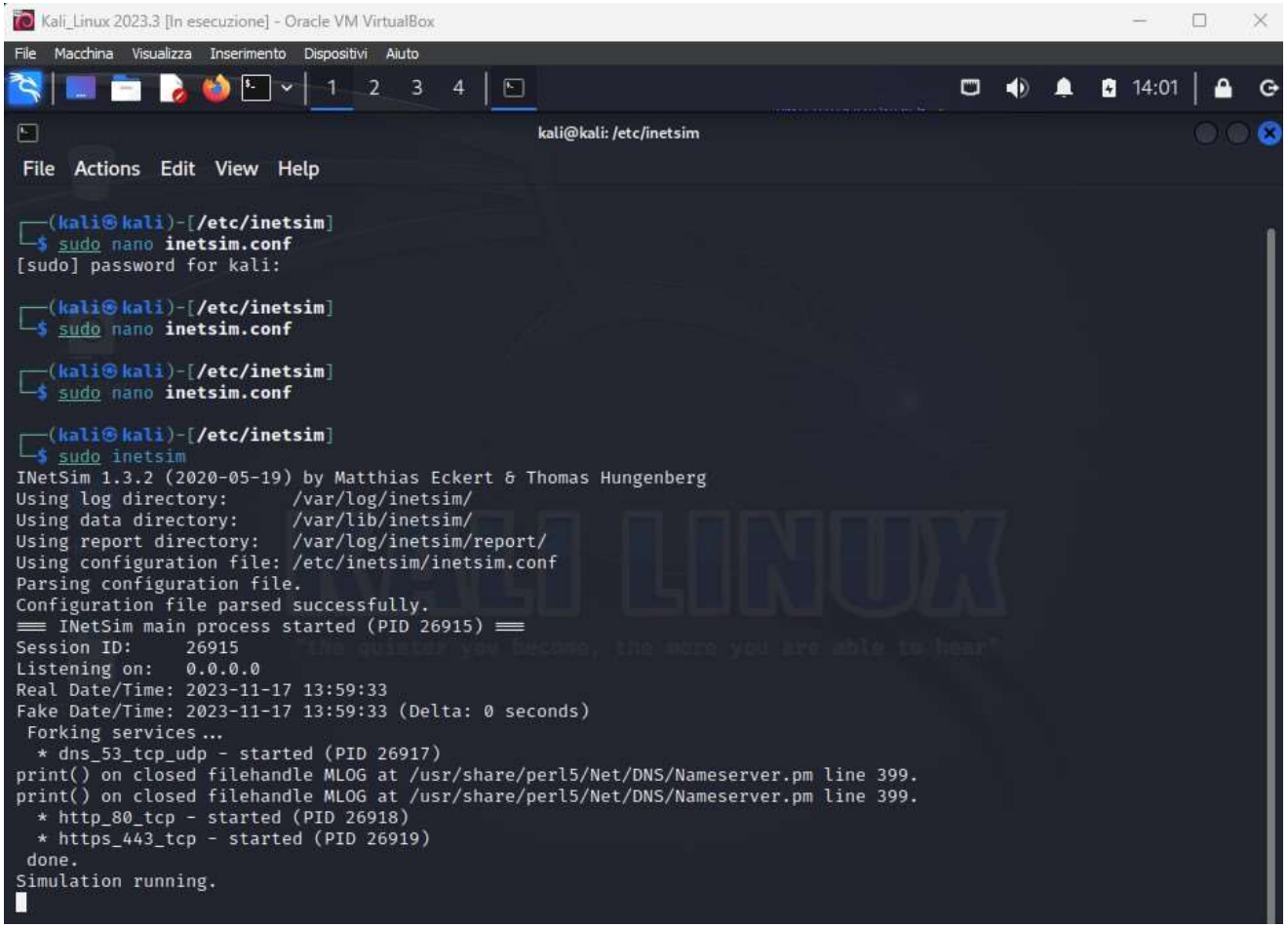
```
#start_service dummy_udp

##############################################
# service_bind_address
#
# IP address to bind services to
#
# Syntax: service_bind_address <IP address>
#
# Default: 127.0.0.1
#
service_bind_address 0.0.0.0

##############################################
# service_run_as_user
#
# User to run services
#
# Syntax: service_run_as_user <username>
```

Scorrendo ancora più in basso, accediamo alla configurazione del **servizio dns**. Imposto come nome del sito(di tipo Record A) il nome “epicode.internal” come indirizzo lo stesso della macchina Kali-linux, in questo caso 192.168.32.100

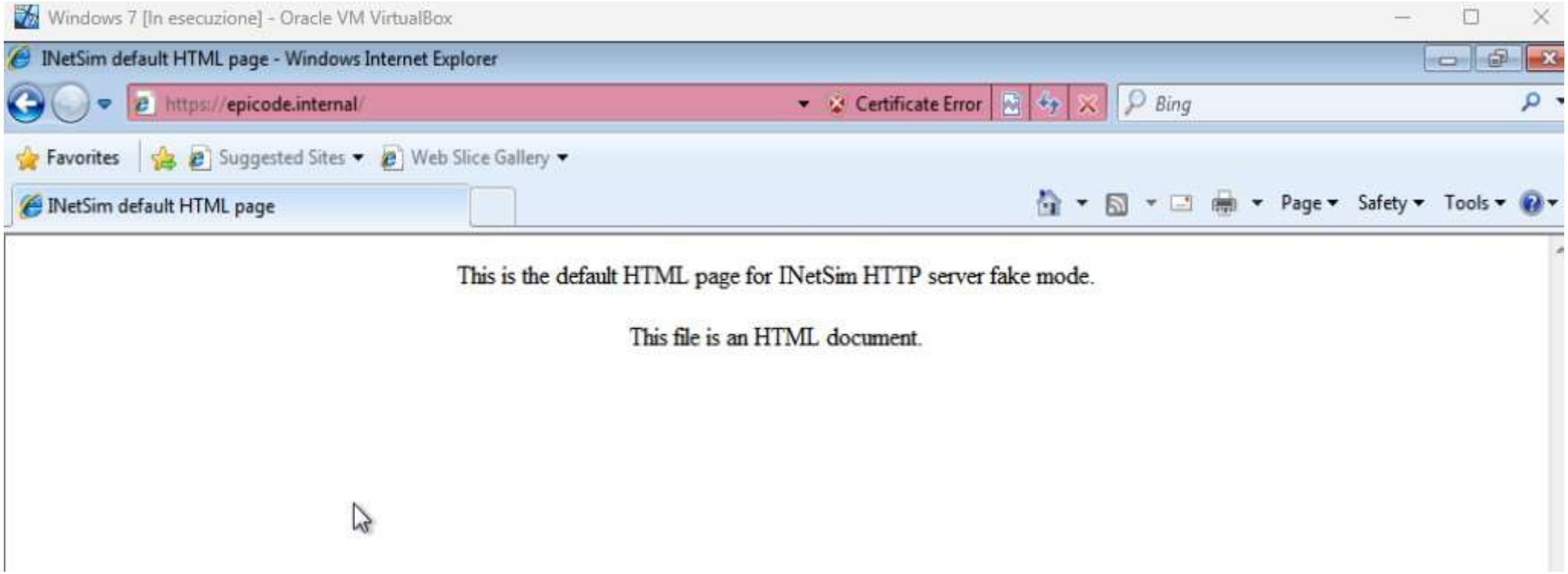
```
##############################################
# dns_static
#
# Static mappings for DNS
#
# Syntax: dns_static <fqdn hostname> <IP address>
#
# Default: none
#
#dns_static www.foo.com 10.10.10.10
#dns_static ns1.foo.com 10.70.50.30
#dns_static ftp.bar.net 10.10.20.30
dns_static epicode.internal 192.168.32.100

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

Come vero Server DNS, esso ha il compito di associare il nome che viene ricercato nel browser web all'indirizzo ip della macchina che contiene la risorsa e di registrarlo nel suo database interno. In sostanza permette instradare il client sul server giusto, anche se lui cerca solo nome senza conoscere il reale indirizzo IP del Server. Ovviamente il pc client deve conoscere l'IP del Server DNS. La cosa si risolve impostando sulla scheda di rete di Windows 7 l'IP del server DNS. Ciò permette alla macchina client di effettuare una richiesta a al DNS di avere l'ip address associato a quel nome.

Una volta configurato il file **inetsim.conf** correttamente, lo salvo, chiudo e riavvio i servizi di rete. A processo ultimato, uso il comando **sudo inetsim** per avviare il servizi configurati in precedenza. Con l'ultima voce "Simulation running" e senza altre segnalazioni di anomalie, il simulatore è avviato correttamente.



Ritorno in windows 7, avvio il web browser e ricerco **https://epicode.internal** per verificare l'avvenuta connessione.

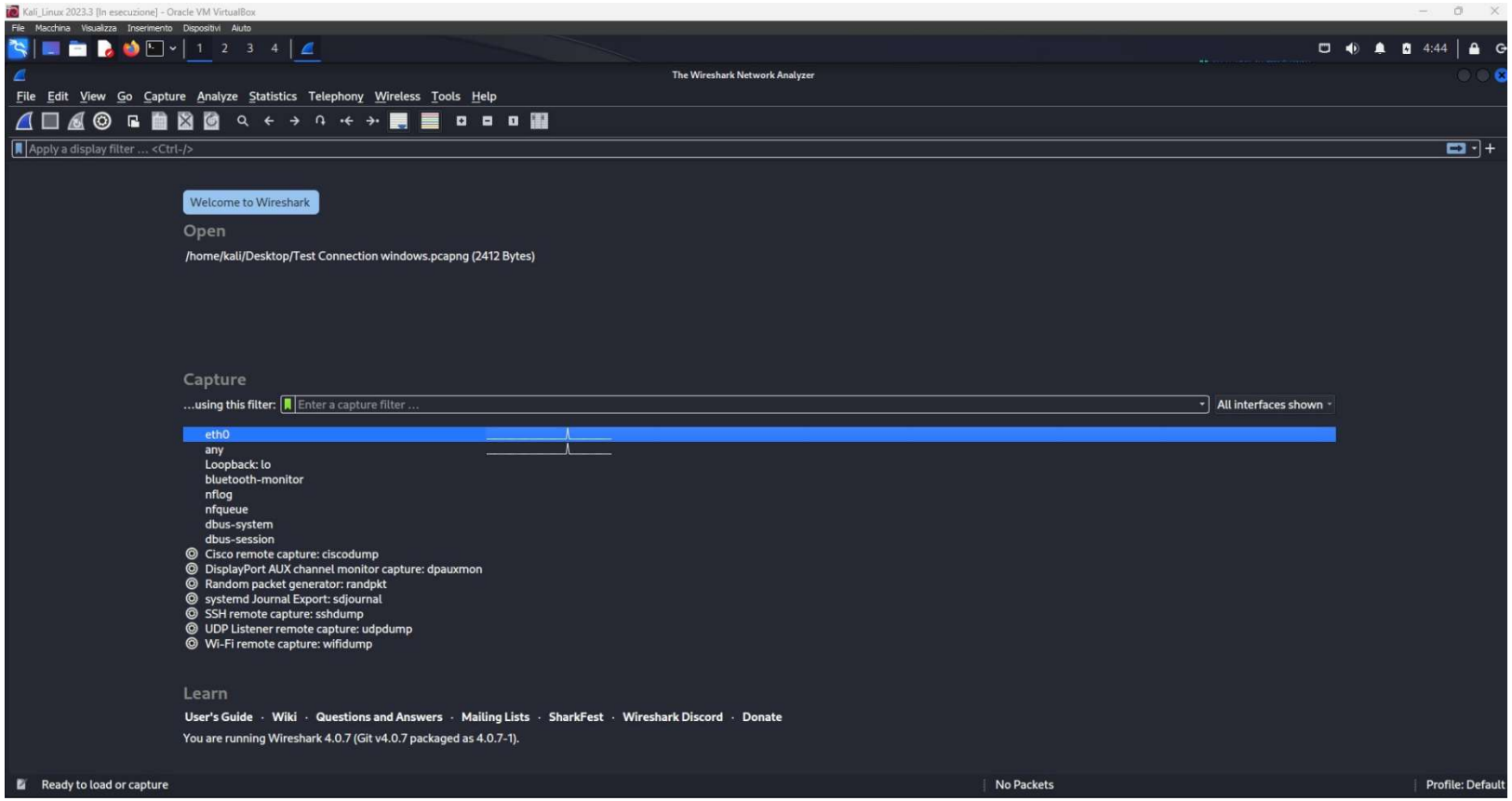


Riporto inoltre la richiesta che ha effettuato Windows 7 al server DNS (impostato su Kali-linux come da configurazione precedente) che chiede a chi corrisponde epicode.internal e il servizio risponde con l'associazione. Pacchetto intercettato con Wireshark.

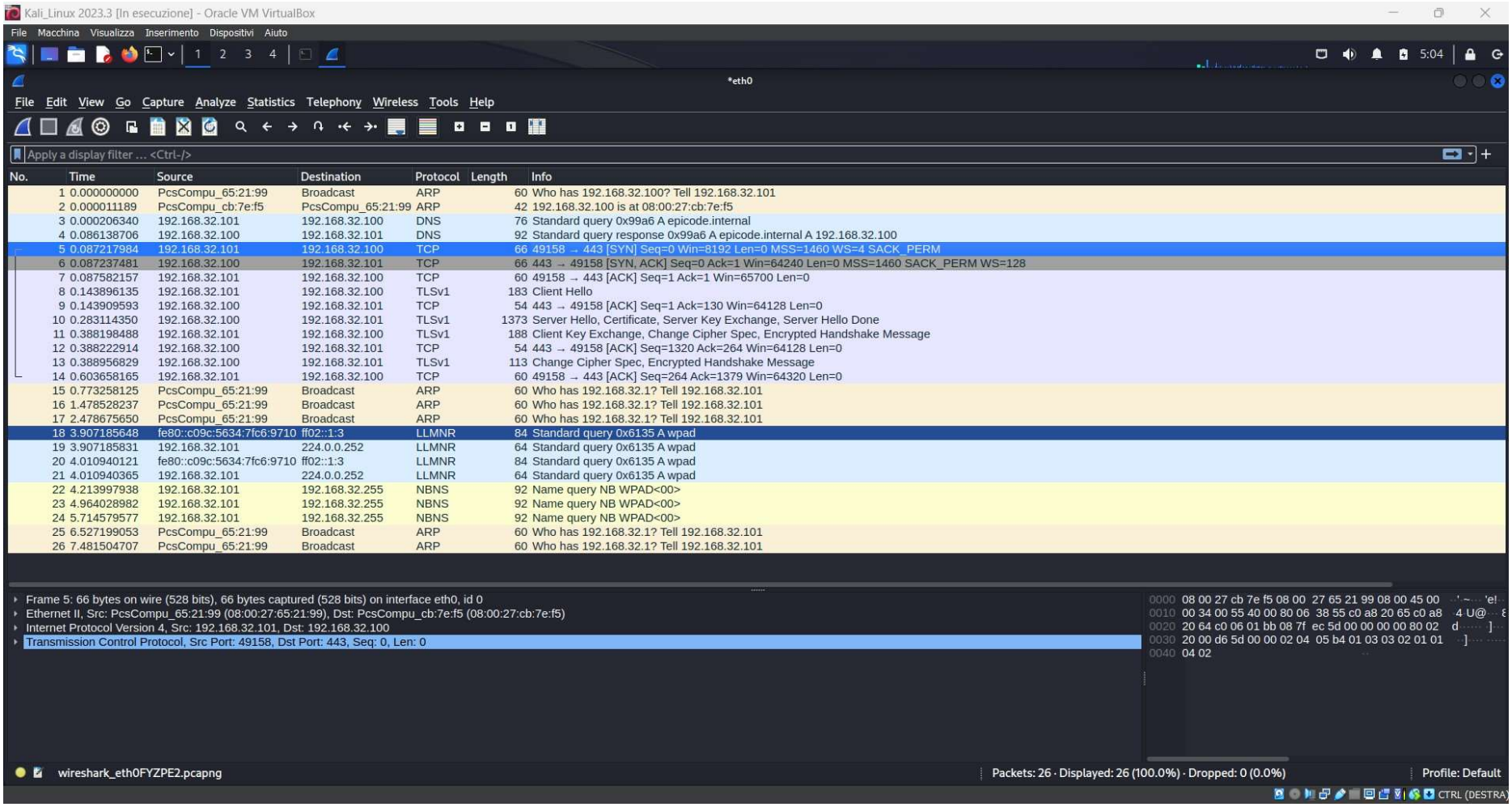
3	0.000206340	192.168.32.101	192.168.32.100	DNS	76	Standard query 0x99a6 A epicode.internal
4	0.086138706	192.168.32.100	192.168.32.101	DNS	92	Standard query response 0x99a6 A epicode.internal A 192.168.32.100

[3] INTERCATTAZIONE PACCHETTI TRAMITE WIRESHARK

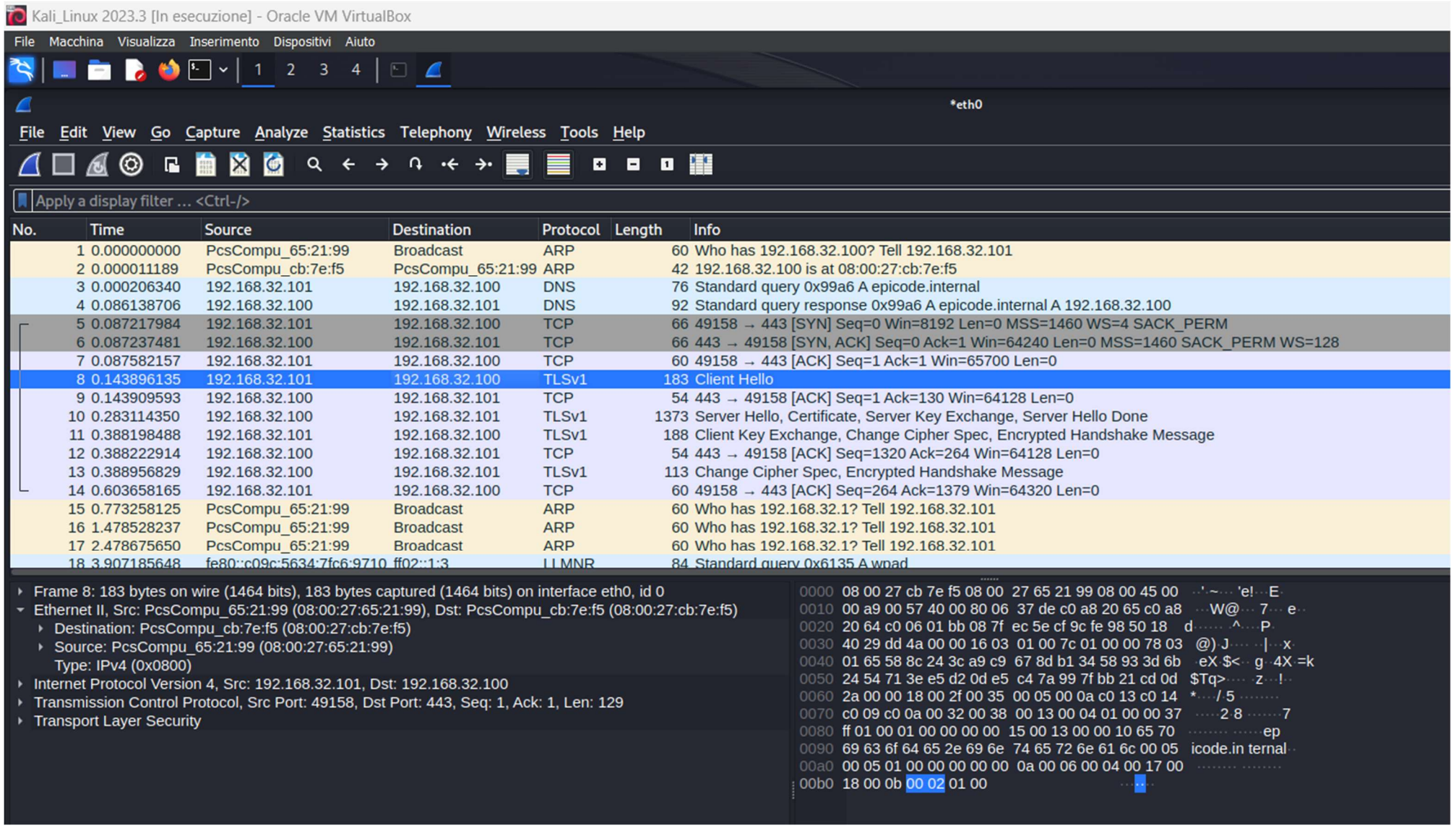
Wireshark è un applicativo che permette di eseguire una analisi sul traffico delle varie schede di rete presenti in una macchina. Come da traccia, tramite l'uso di Wireshark, intercetto tutti i pacchetti che transitano fra il pc Kali-linux e Windows7 quando quest'ultimo prova ad effettuare un accesso alla risorsa tramite il suo web browser con protocollo HTTPS. Di seguito avvio Wireshark su Kali-linux.



Seleziono la voce **eth0,** accedo alla schermata successiva che avvia automaticamente l'ascolto e intercettazione dei pacchetti su quella scheda di rete. Faccio presente che, come configurato in precedenza, la scheda **eth0** ha come **IP 192.168.32.100/24** e che il servizio di simulazione **InetSim** è avviato. Da qui torno in Windows 7 ed effettuo nuovamente la ricerca sul web browser (ovvero **https://epicode.internal).** Il risultato è quanto segue.

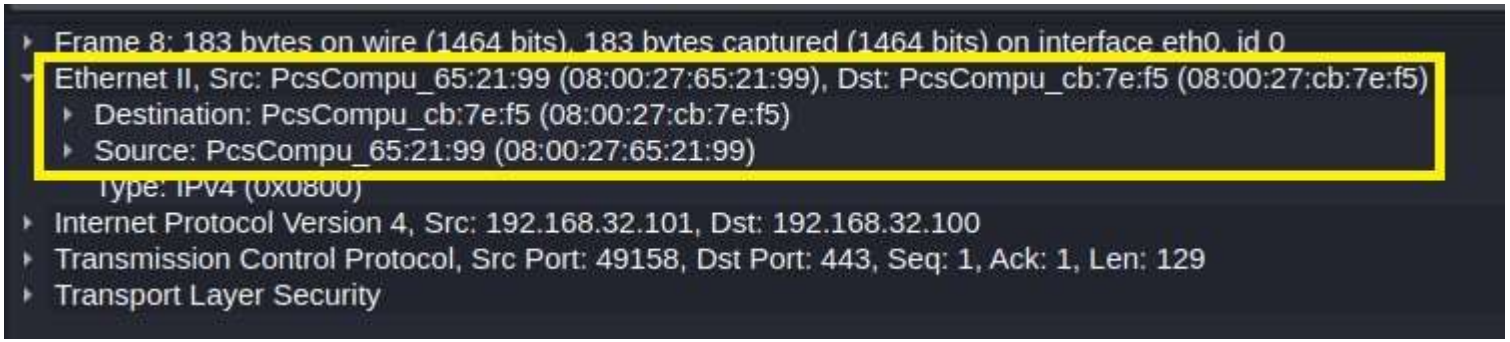


Seleziono uno dei pacchetti intercettati da **WireShark**.



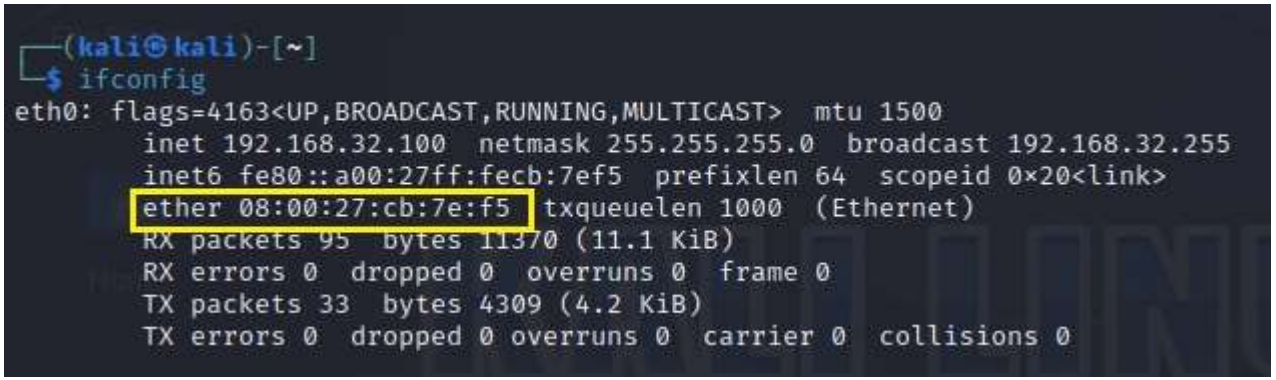
Come protocollo **HTTPS** applica, i pacchetti in transito e le varie richieste e risposte che avvengono da entrambe le macchine sono criptate. Perciò non mi sarà possibile identificare il tipo di richiesta che sta effettuando in questo caso il client, ma posso semplicemente “dedurlo”, senza avere una certezza.

Selezionando il pacchetto con “Info” di “Client Hello”, che si presenta dopo la three-way-handshake (ovvero il metodo con cui i due dispositivi si mettono in comunicazione concordando la cifratura) vediamo i vari dati che transitano. Quello che evidenzio è il Source MAC Address e il Destination MAC Address di questo pacchetto. In questo caso il pc Windows7 fa la richiesta di una risorsa a Kali-Linux

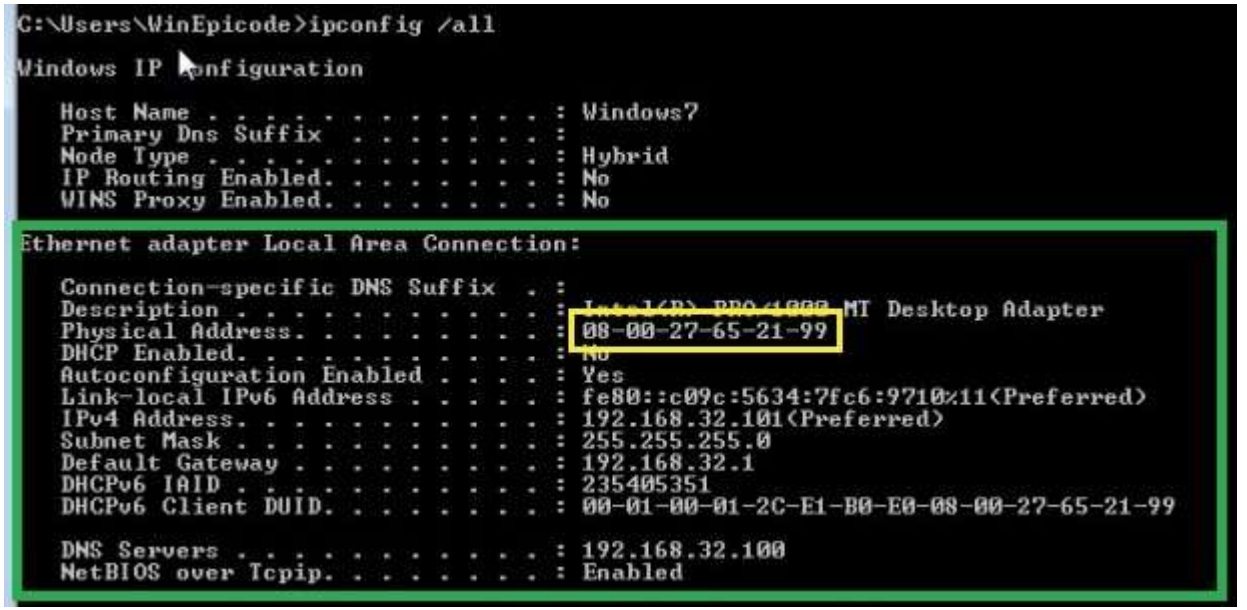


Per prova indico di seguito riporto le schemare di **ifconfig** (Kali-Linux) e **ipconfig**(Windows 7) che mostrano i MAC Address delle due macchine, indicando quale è il Source e quale il Destination.

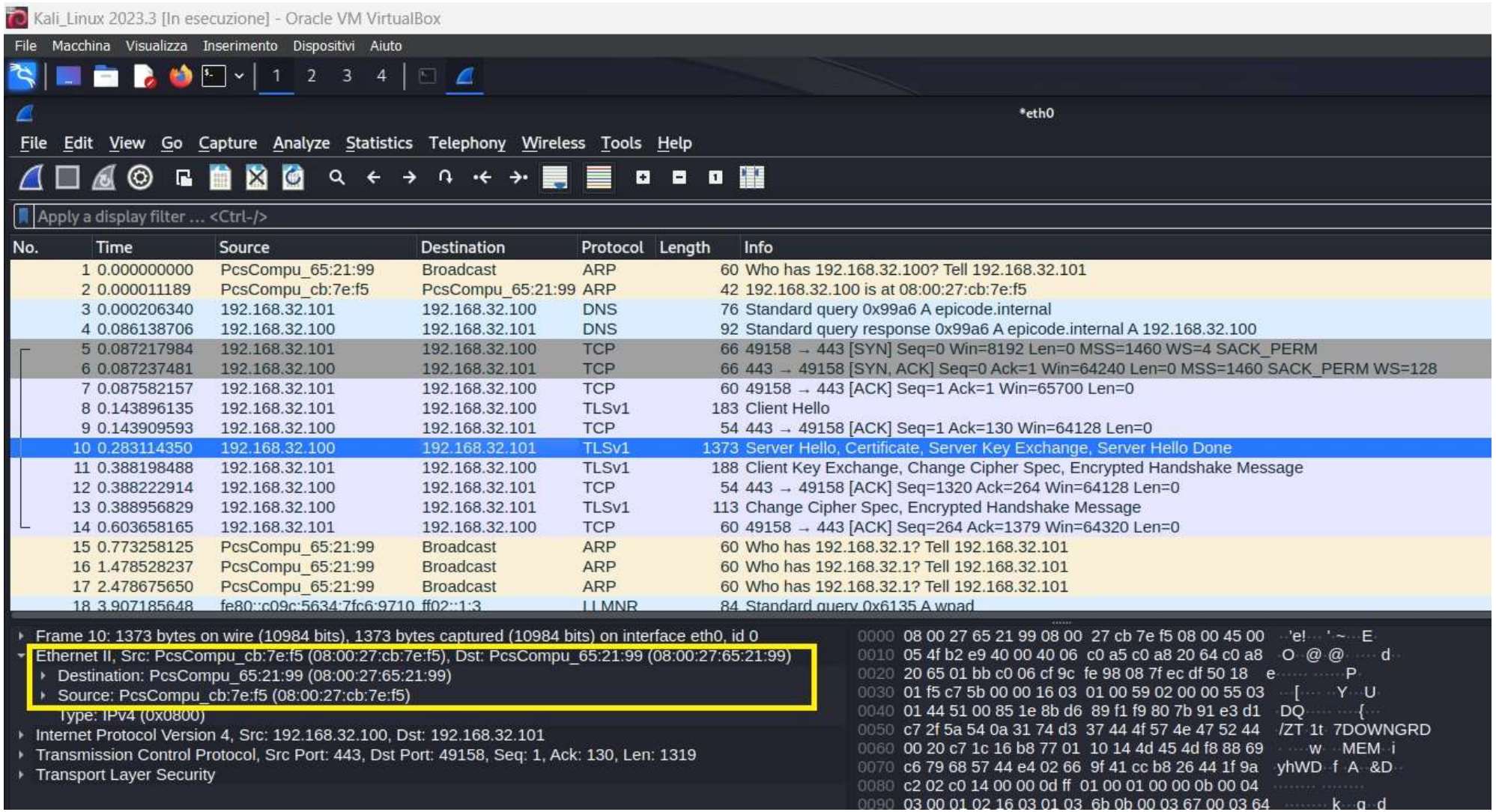
Questa di destra si tratta dell'IP di Kali-Linux, nell'esempio citato prima si tratta del Destination MAC (Dst) e ne evidenzio il l'indirizzo, essere scritto su **ether.**



Quest'altra invece è il prompt dei comandi di Windows 7 che, sempre nel caso precedente, si tratta della macchina Source che fa la richiesta della risorsa a Kali-linux. Evidenzio tale indirizzo in giallo che su Windows 7 corrisponde a **physical address**.



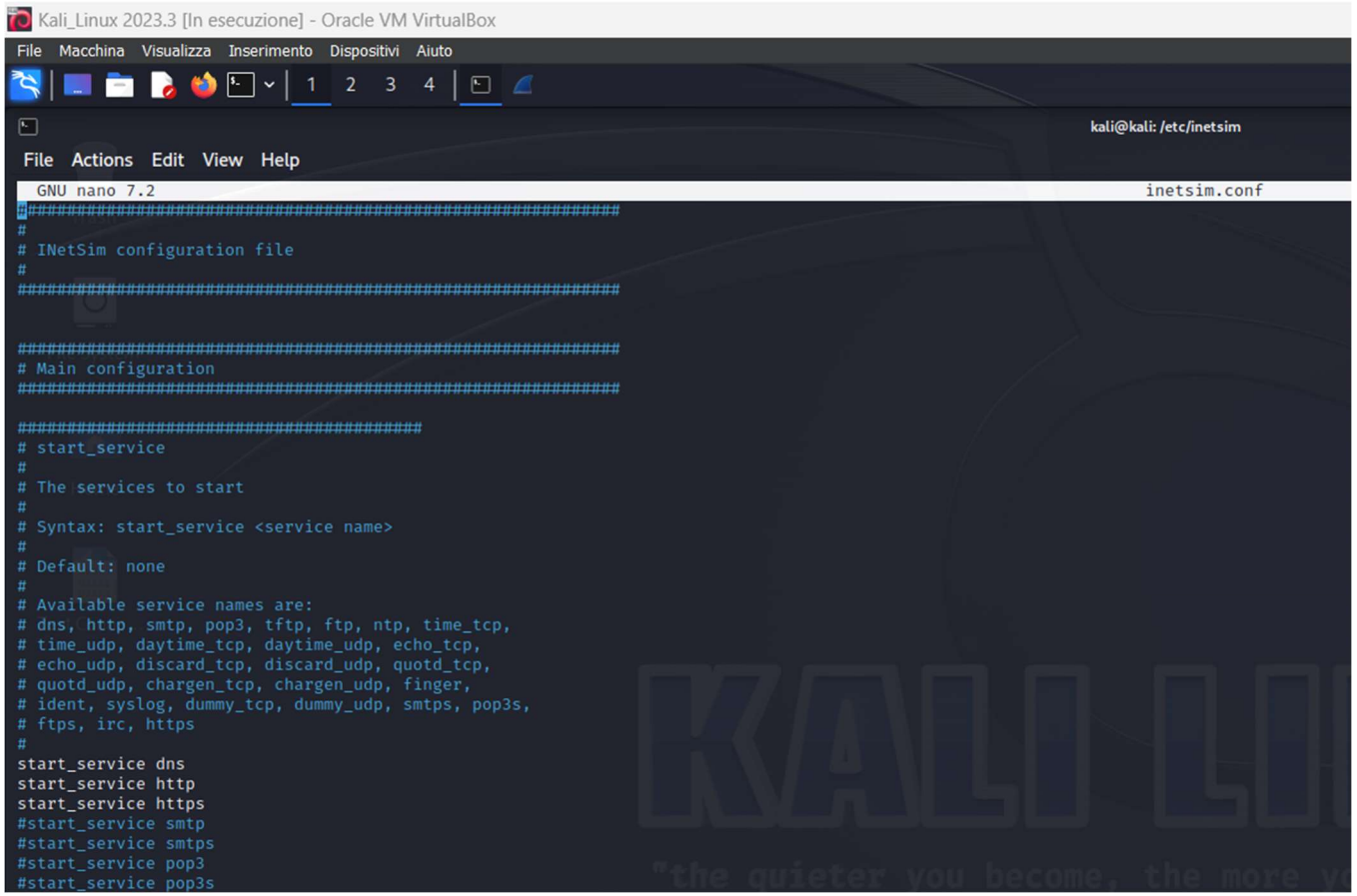
I **MAC Address**, sia **Source(Src)** che **Destination(Dst),** cambieranno posizione quando il Server Kali-linux effettuerà la risposta al cliente nel medesimo protocollo. Di seguito riporto una schermata del pacchetto successivo ma con l'inversione prima citata. Di fatto il Server risponde alla richiesta "Cliente Hello" con "Server Hello" e sempre in modo **cifrato**, mostrando che però i MAC Address s'invertono.



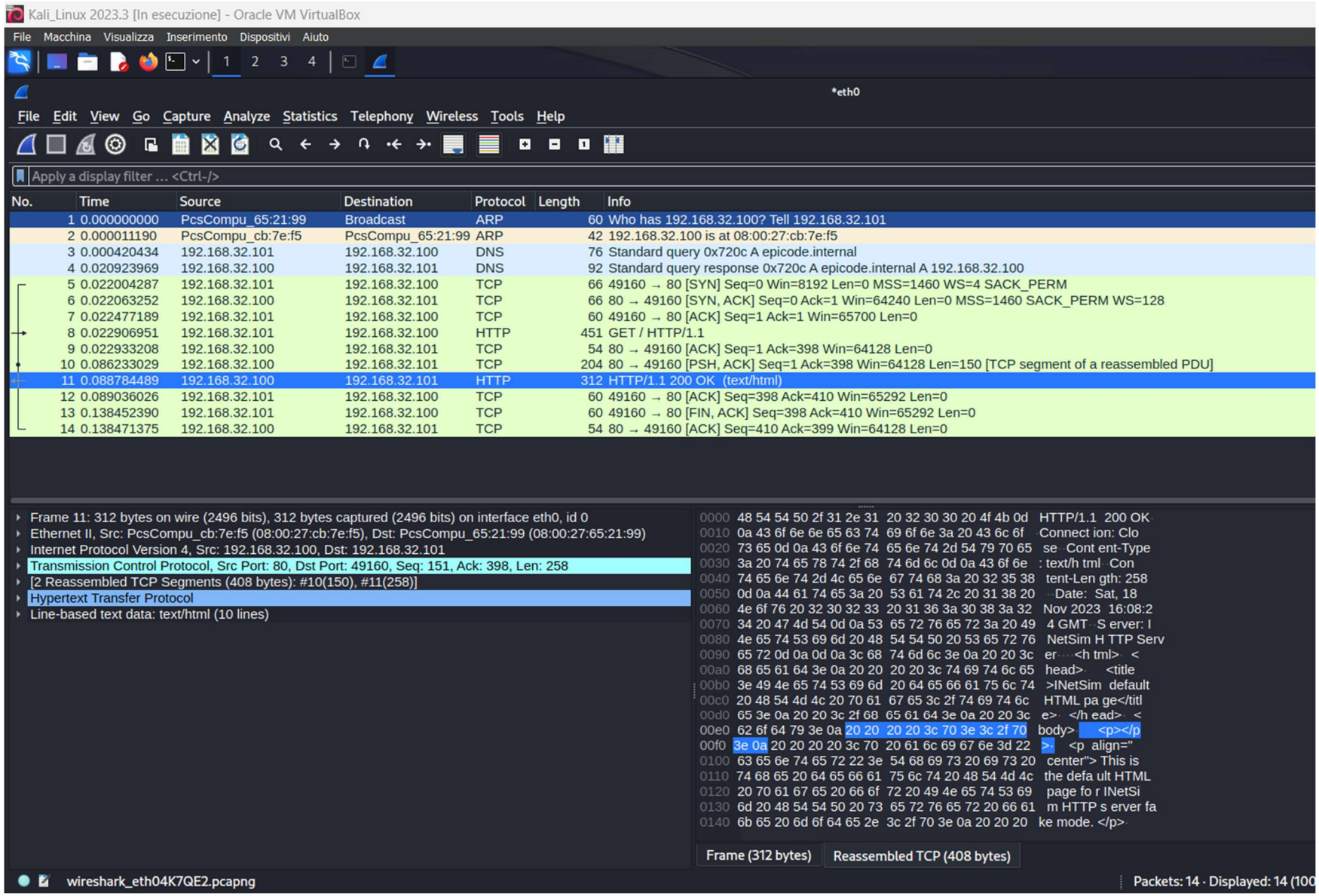
DIFFERENZE FRA PROTOCOLLO HTTP E HTTPS

Nella parte precedente ho visto che tipologia di **frame** navigano durante una comunicazione fra Client e Server con il protocollo **HTTPS**, di livello 7 Applicazione. Ho effettuato nuovamente la procedura del passo precedente, ma sta volta utilizzando il protocollo **HTTP.**

Prima di procedere però, disattivo momentaneamente InetSim su Kali-linux, riconfiguro il file **inetsim.conf** e rimuovo il **#** sul servizi **start_service https** come da schemata. Salvo la configurazione precedente e rieseguo il comando **sudo inetsim** per riattivare il simulatore.

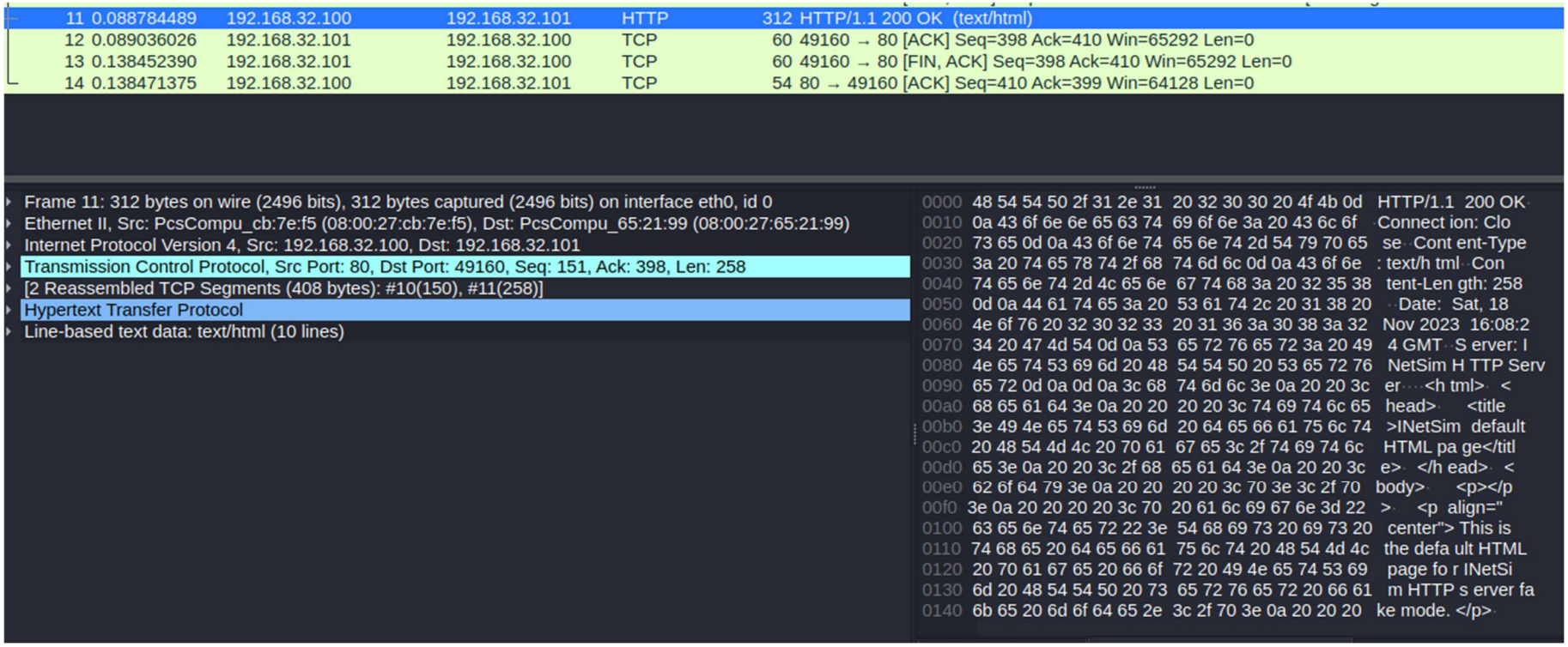


Da qui torno su Windows e sul Web browser effettuo la ricerca di **http://epicode.internal.** Ne intercetto i pacchetti con **Wireshark** come in precedenza.

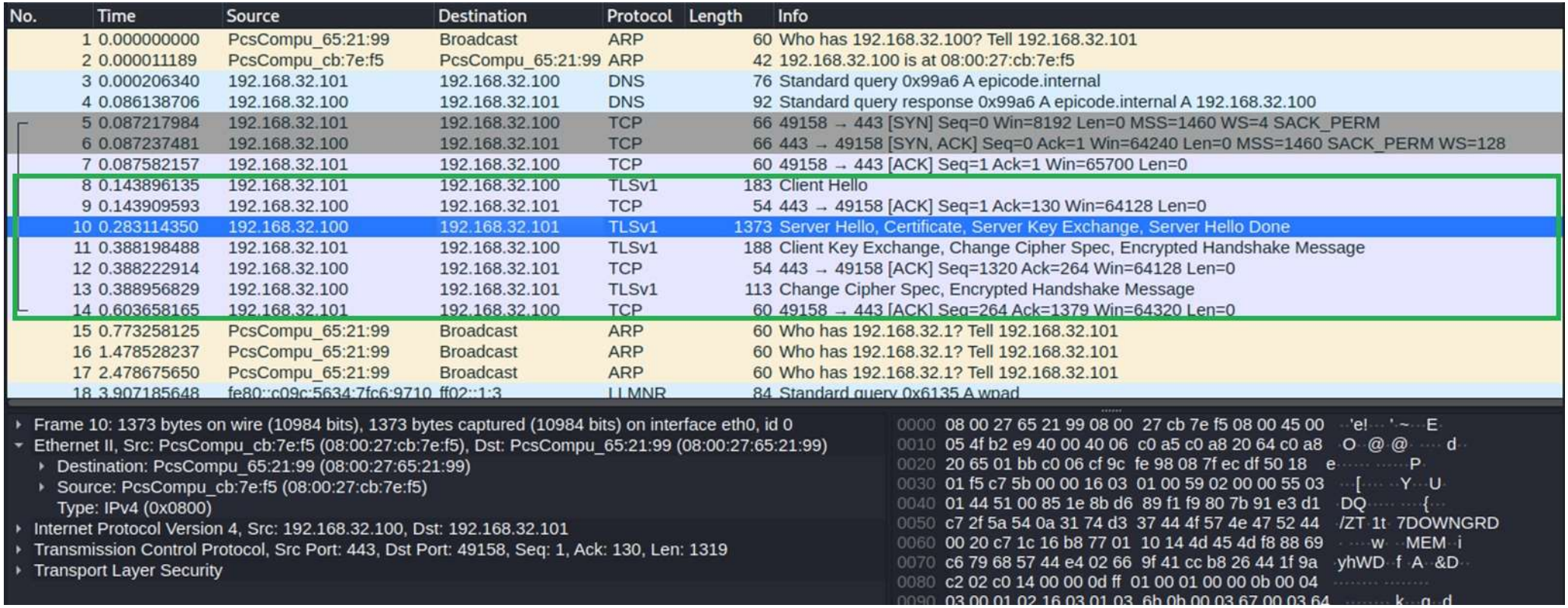


Le principali differenze che ho percepito sui due protocolli, ovvero **HTTP** (HyperText Transport Protocol) e **HTTPS**(http protetto da certificati SSL/TLS), sono le seguenti:

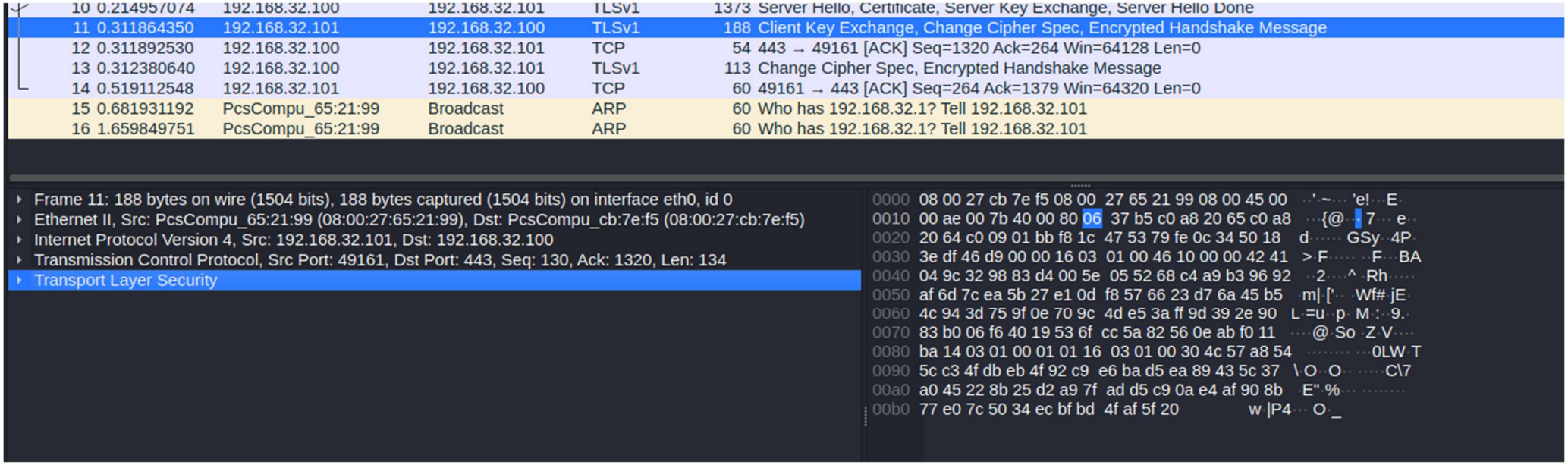
- Nel **protocollo HTTP** il flusso di pacchetti che avviene fra Client e Server è “in chiaro”, ovvero oltre alla three-way-handshake che avviene a livello Trasporto del modello ISO/OSI con protocollo TCP, posso vedere la richiesta GET effettuata dal Client Windows 7 e le risposte del Server Kali-linux in merito. Questo fa capire che i pacchetti intercettati possono essere letti completamente fino all’ultimo livello della pila ISO/OSI (layer Applicazione) e ciò rende la comunicazione insicura, in quanto un Man-in-the-middle potrebbe sniffare i pacchetti e rubare dati potenzialmente sensibili. In esempio prendo il pacchetto con il 200 OK (page found) e riesco a vedere il contenuto che il server manda al client, ovvero quello che sta visualizzando l’utente finale.



- **Nel protocollo HTTPS**, al contrario di quello precedente, i pacchetti non viaggiano in chiaro, ma bensì criptati (in questo caso con la certificazione TLS). Di fatto, a differenza del precedente, dopo la three-way-handshake, tutti i pacchetti sono cifrati e non mi è possibile vedere il tipo di richieste che il client chiede al server se non quello che segue.



Se io provo ad aprire il pacchetto col protocollo TLS, quello che vedrò all'interno è tutto messaggio cifrato. Non posso identificarne il contenuto e non posso vedere nemmeno quello che il client chiede(request) e che il server risponde(response).
Come nell'immagine di seguito.



Come visibile in basso a destra, prendendo una ipotetica richiesta che sta facendo il client al server, il contenuto è cifrato. Tutto questo permette una connessione sicura e un potenziale Man-in-the-middle non può decifrarne il contenuto e capire se vi sono dati sensibili in transito.

In breve le differenze presenti sono:

- In **HTTP** i messaggi sono in chiaro e quindi non sicuri, Client e Server non concordano cifrature e tutte le operazioni eseguite da entrambe le macchine sono visibile ad un malintenzionato.
- In **HTTPS** i messaggi sono criptati tramite protocolli come SSL o TLS e quindi più sicuri, prima di visualizzare concordano il metodo di criptazione dei messaggi, non sono visibili le operazioni fra le due macchine coinvolte e quindi un malintenzionato non sa cosa sta succedendo.