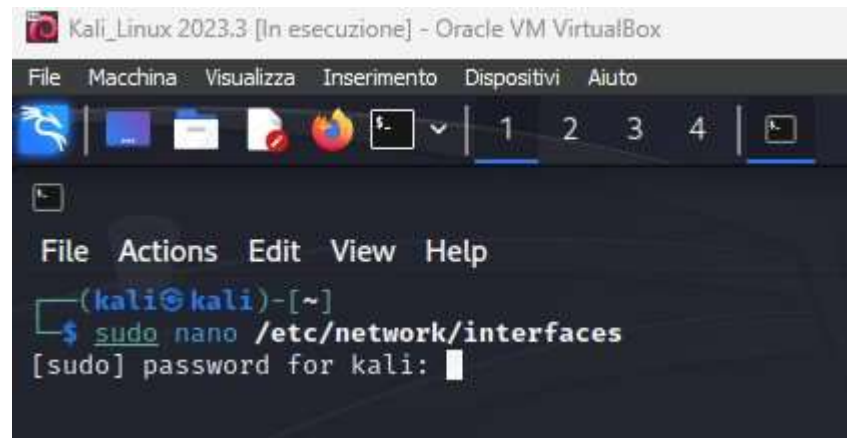


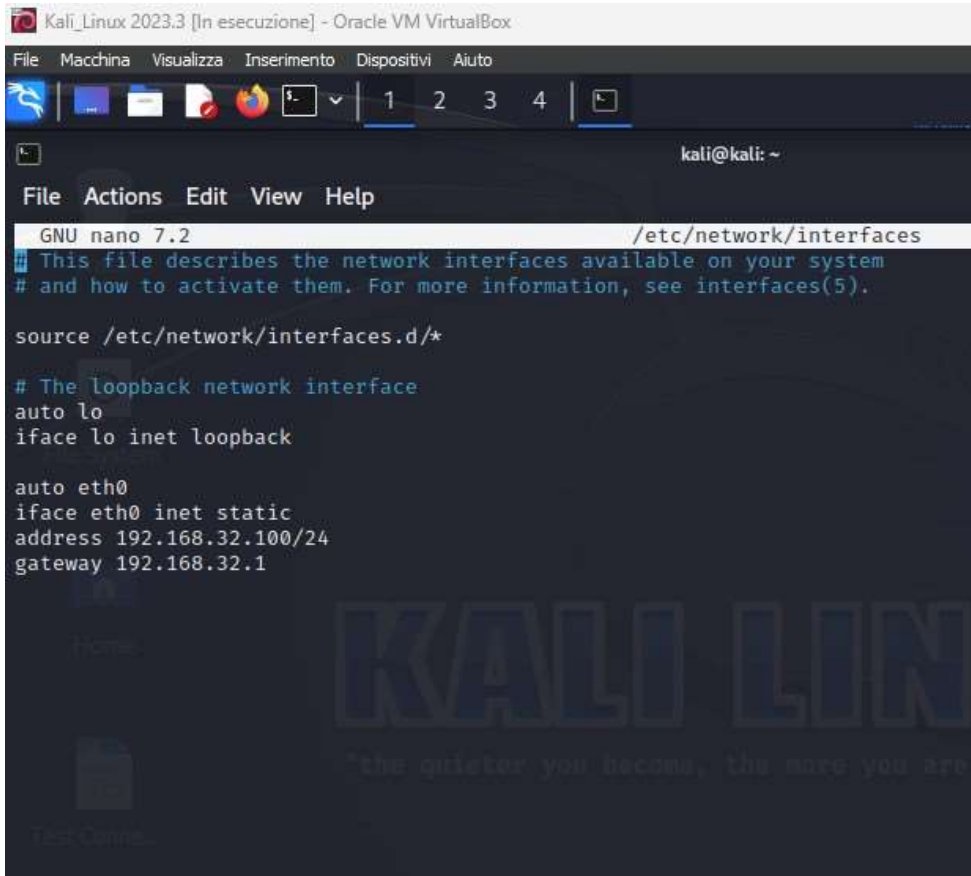
## ESERCITAZIONE WEEK 4 DAY 5

## [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.



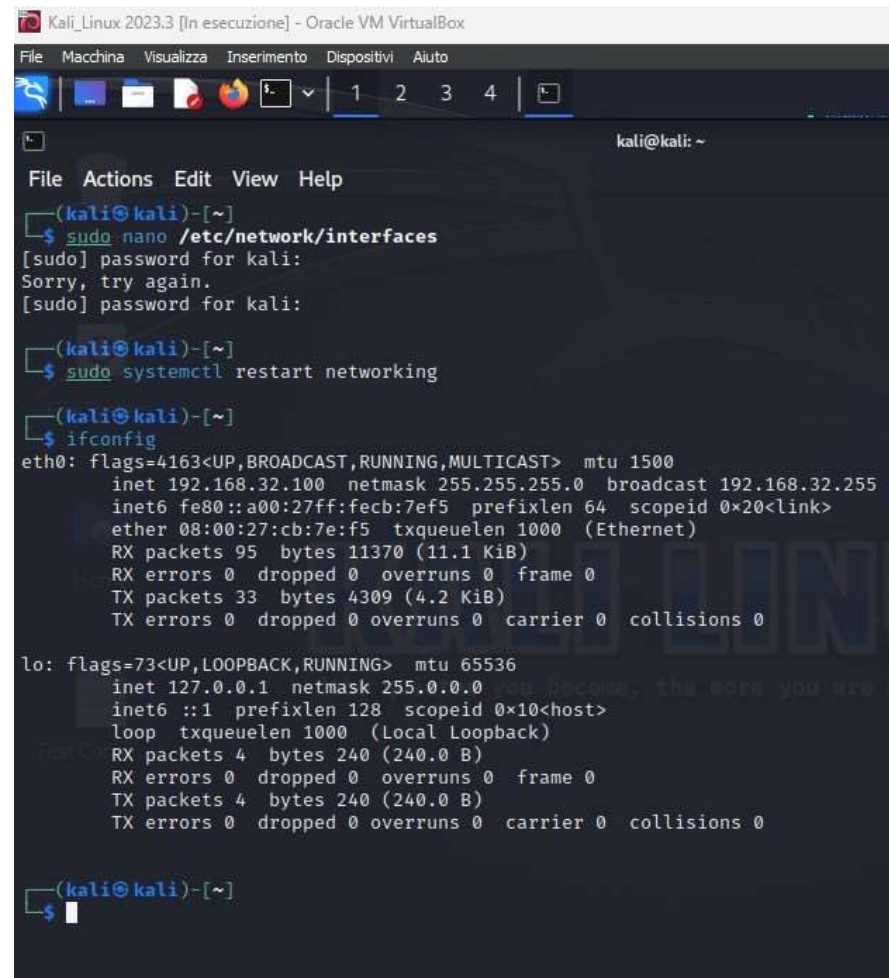
```
Kali_Linux 2023.3 [In esecuzione] - Oracle VM VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
GNU nano 7.2 /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.32.100/24
gateway 192.168.32.1
```

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.



```
Kali_Linux 2023.3 [In esecuzione] - Oracle VM VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto

kali@kali: ~
File Actions Edit View Help
(kali@kali)-[~]
$ sudo nano /etc/network/interfaces
[sudo] password for kali:
Sorry, try again.
[sudo] password for kali:

(kali@kali)-[~]
$ sudo systemctl restart networking

(kali@kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.32.100 netmask 255.255.255.0 broadcast 192.168.32.255
    inet6 fe80::a00:27ff:feeb:7ef5 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:cb:7e:f5 txqueuelen 1000 (Ethernet)
    RX packets 95 bytes 11370 (11.1 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 33 bytes 4309 (4.2 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 4 bytes 240 (240.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 4 bytes 240 (240.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

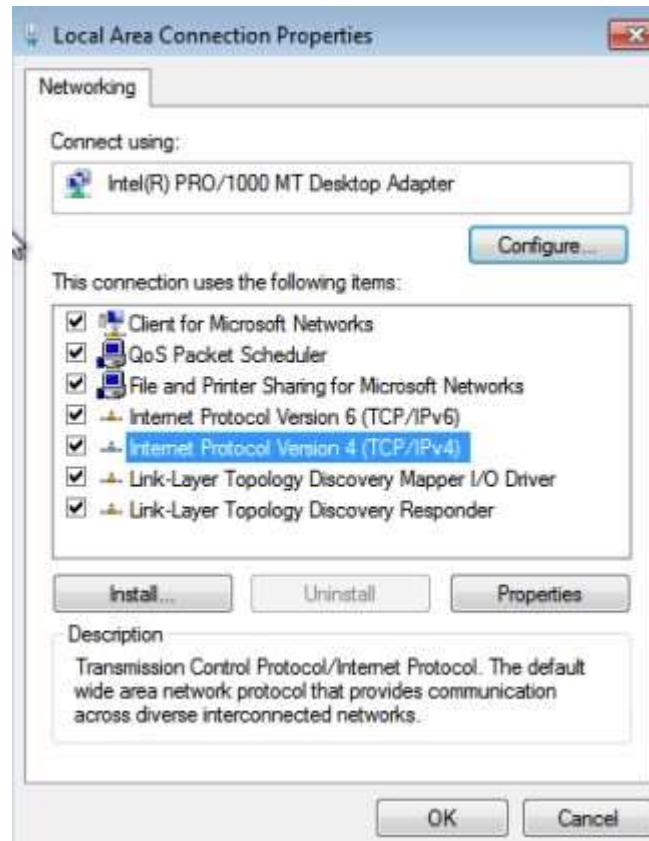
(kali@kali)-[~]
$
```

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.

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\WinEpicode>ipconfig /all

Windows IP Configuration

Host Name . . . . . : Windows7
Primary Dns Suffix . . . . . :
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No

Ethernet adapter Local Area Connection:

Connection-specific DNS Suffix . :
Description . . . . . : Intel(R) PRO/1000 MT Desktop Adapter
Physical Address. . . . . : 08-00-27-65-21-99
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::c09c:5634:7fc6:9710%11(Preferred)
IPv4 Address. . . . . : 192.168.32.101(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.32.1
DHCPv6 IAID . . . . . : 235405351
DHCPv6 Client DUID. . . . . : 00-01-00-01-2C-E1-B0-E0-08-00-27-65-21-99

DNS Servers . . . . . : 192.168.32.100
NetBIOS over Tcpip. . . . . : Enabled

Tunnel adapter isatap.{A0DB5A70-3FF7-4B2C-B006-8AA97AE793F3}:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :
Description . . . . . : Microsoft ISATAP Adapter
Physical Address. . . . . : 00-00-00-00-00-00-00-E0
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . : Yes

C:\Users\WinEpicode>
```



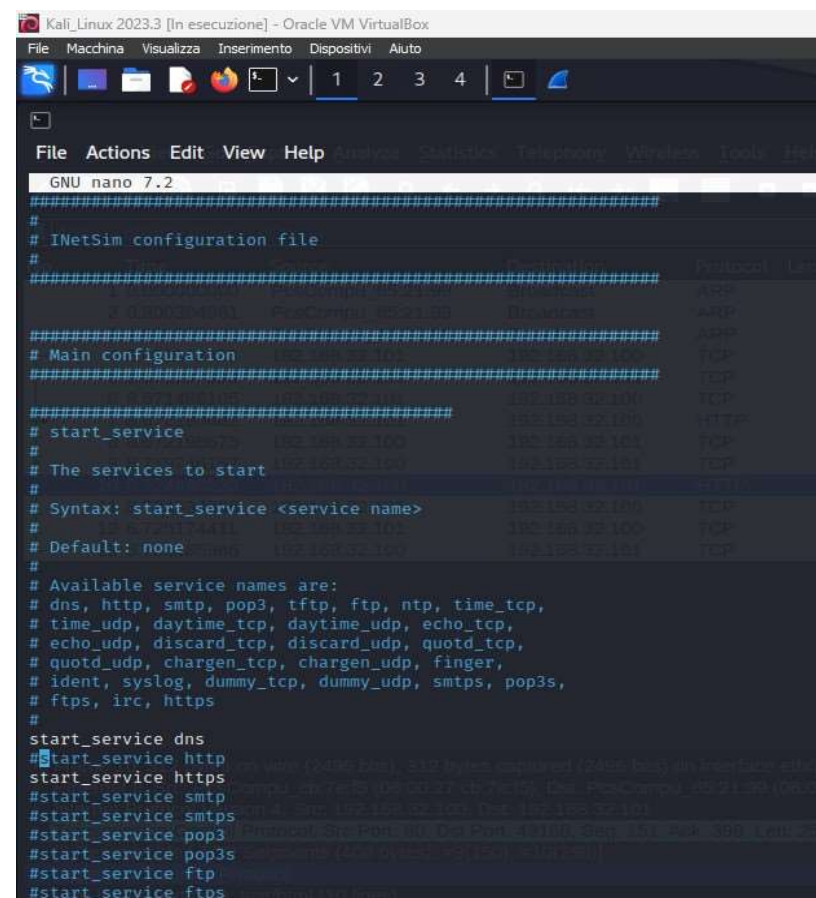
## [2] SIMULAZIONE DI LABORATORIO VIRTUALE DI UN'ARCHITETTURA CLIENT-SERVER

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.



```
Kali_Linux 2023.3 [In esecuzione] - Oracle VM VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
File Actions Edit View Help Analizza Statistica Telefono Wireless Tools Help
GNU nano 7.2
#####
#
# 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
```

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.

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

```
#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

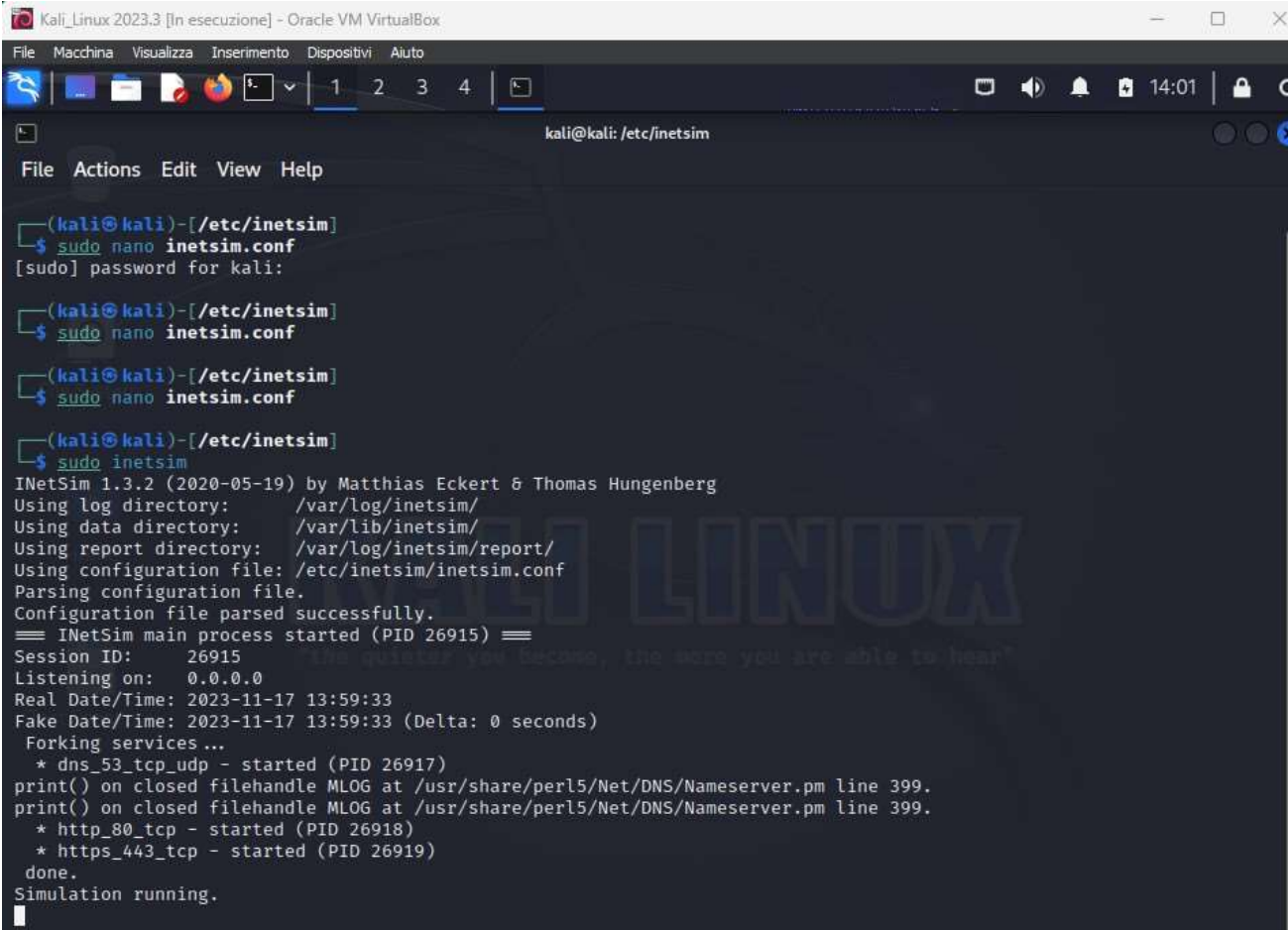
#####
# service_run_as_user
#
# User to run services
#
# Syntax: service_run_as_user <username>

#####
# 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

#####
# dns_service
```

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.

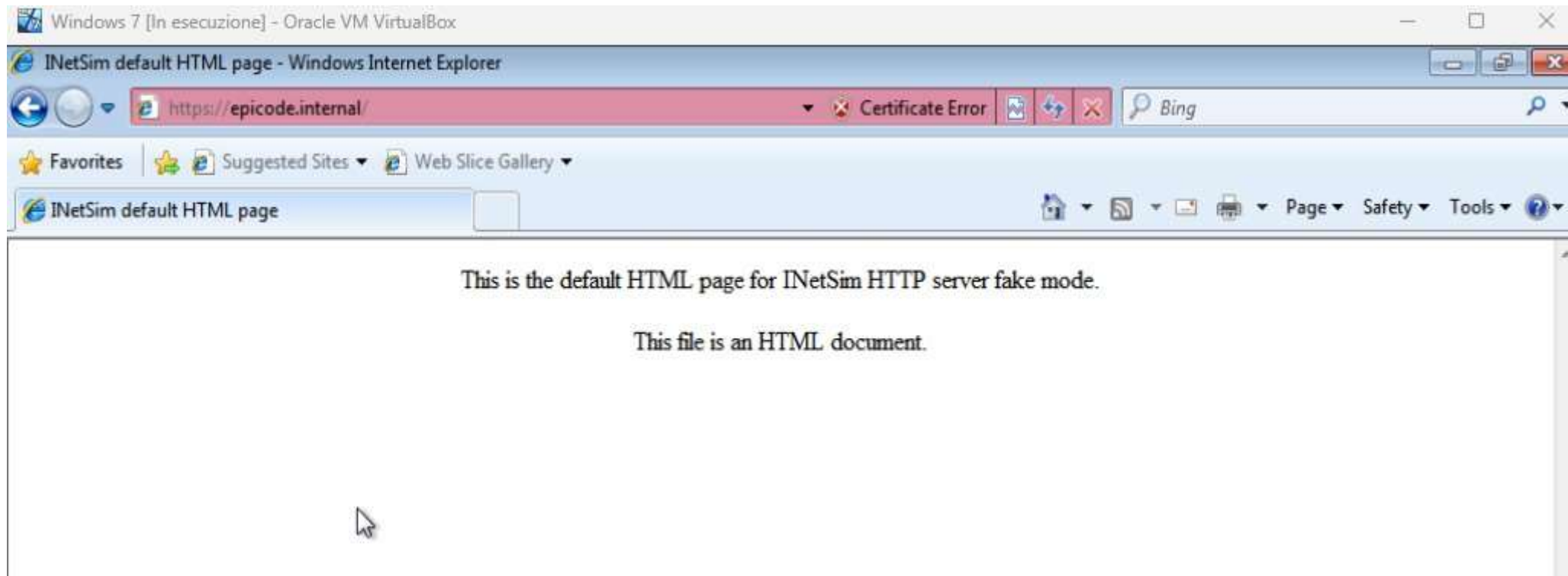
Una volta configurato il file **inetsim.conf** correttamente, lo salvo, chiudo e riavvio i servizi di rete. A processo ultimato, uso il comando **sudo inetsim** per avviare il servizi configurati in precedenza. Con l'ultima voce "Simulation running" e senza altre segnalazioni di anomalie, il simulatore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



```
Kali Linux 2023.3 [In esecuzione] - Oracle VM VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
kali@kali: /etc/inetsim
File Actions Edit View Help
(kali@kali)-[/etc/inetsim]
$ sudo nano inetsim.conf
[sudo] password for kali:
(kali@kali)-[/etc/inetsim]
$ sudo nano inetsim.conf
(kali@kali)-[/etc/inetsim]
$ sudo nano inetsim.conf
(kali@kali)-[/etc/inetsim]
$ 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 26915) ==
Session ID: 26915
Listening on: 0.0.0.0
Real Date/Time: 2023-11-17 13:59:33
Fake Date/Time: 2023-11-17 13:59:33 (Delta: 0 seconds)
Forking services ...
* dns_53_tcp_udp - started (PID 26917)
print() on closed filehandle MLOG at /usr/share/perl5/Net/DNS/Nameserver.pm line 399.
print() on closed filehandle MLOG at /usr/share/perl5/Net/DNS/Nameserver.pm line 399.
* http_80_tcp - started (PID 26918)
* https_443_tcp - started (PID 26919)
done.
Simulation running.
```



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

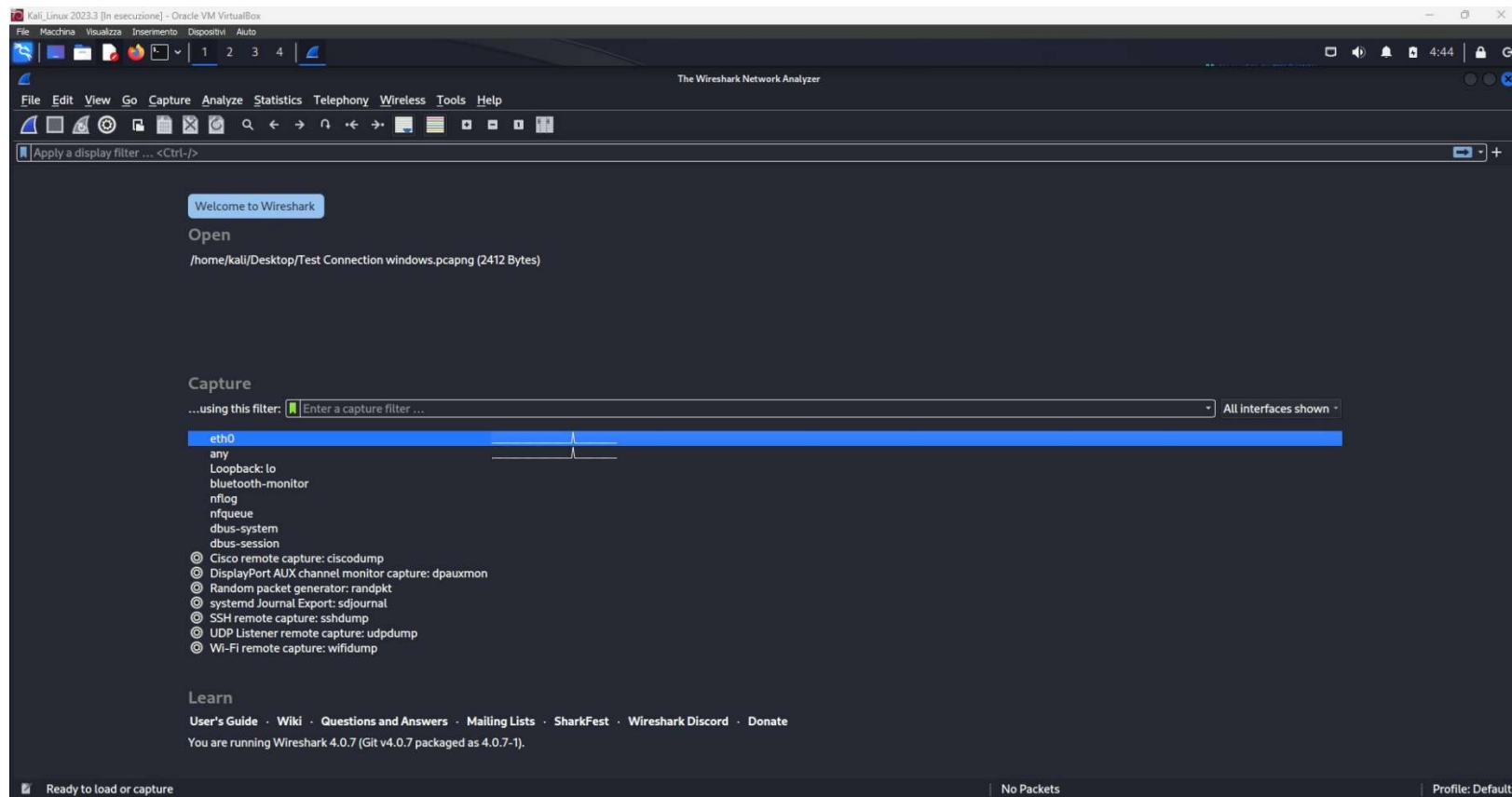


Riporto inoltre la richiesta che ha effettuato Windows 7 al server DNS (impostato su Kali-linux come da configurazione precedente) che chiede a chi corrisponde epicode.internal e il servizio risponde con l'associazione. Pacchetto intercettato con Wireshark.

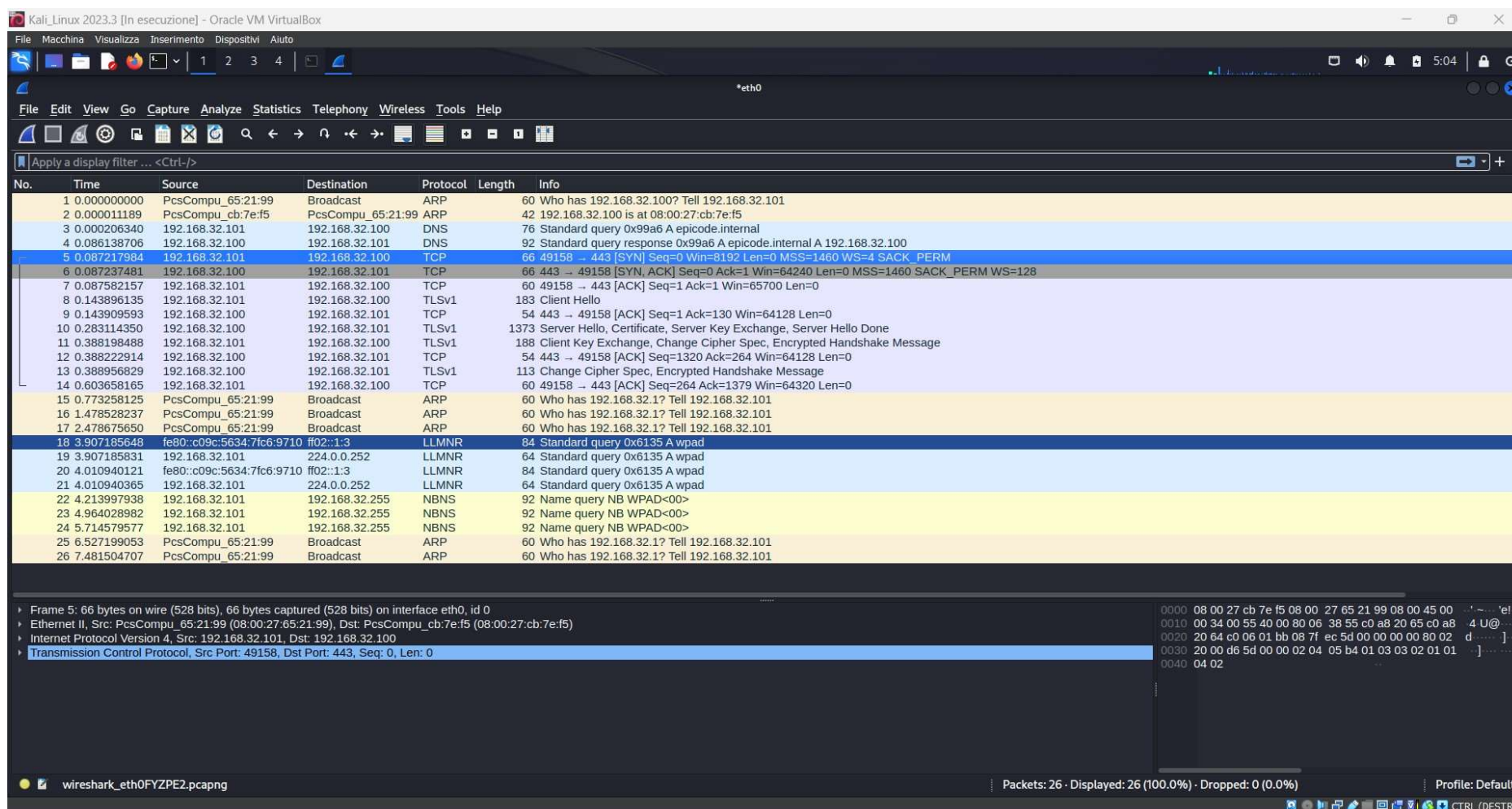
3	0.000206340	192.168.32.101	192.168.32.100	DNS	76	Standard query 0x99a6 A epicode.internal
4	0.086138706	192.168.32.100	192.168.32.101	DNS	92	Standard query response 0x99a6 A epicode.internal A 192.168.32.100

### [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.



The screenshot displays the Wireshark network protocol analyzer interface. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. The packet list pane shows a series of captured packets, with packet 5 selected. The packet details pane shows the selected packet's structure, including Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol. The packet bytes pane shows the raw data of the selected packet.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu_65:21:99	Broadcast	ARP	60	Who has 192.168.32.100? Tell 192.168.32.101
2	0.000011189	PcsCompu_cb:7e:f5	PcsCompu_65:21:99	ARP	42	192.168.32.100 is at 08:00:27:cb:7e:f5
3	0.000206340	192.168.32.101	192.168.32.100	DNS	76	Standard query 0x99a6 A epicode.internal
4	0.086138706	192.168.32.101	192.168.32.100	DNS	92	Standard query response 0x99a6 A epicode.internal A 192.168.32.100
5	0.087217984	192.168.32.101	192.168.32.100	TCP	66	49158 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
6	0.087237481	192.168.32.100	192.168.32.101	TCP	66	443 → 49158 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
7	0.087582157	192.168.32.101	192.168.32.100	TCP	60	49158 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0
8	0.143896135	192.168.32.101	192.168.32.100	TLSv1	183	Client Hello
9	0.143909593	192.168.32.100	192.168.32.101	TCP	54	443 → 49158 [ACK] Seq=1 Ack=130 Win=64128 Len=0
10	0.283114350	192.168.32.100	192.168.32.101	TLSv1	1373	Server Hello, Certificate, Server Key Exchange, Server Hello Done
11	0.388198488	192.168.32.101	192.168.32.100	TLSv1	188	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
12	0.388222914	192.168.32.100	192.168.32.101	TCP	54	443 → 49158 [ACK] Seq=1320 Ack=264 Win=64128 Len=0
13	0.388956829	192.168.32.100	192.168.32.101	TLSv1	113	Change Cipher Spec, Encrypted Handshake Message
14	0.603658165	192.168.32.101	192.168.32.100	TCP	60	49158 → 443 [ACK] Seq=264 Ack=1379 Win=64320 Len=0
15	0.773258125	PcsCompu_65:21:99	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101
16	1.478528237	PcsCompu_65:21:99	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101
17	2.478675650	PcsCompu_65:21:99	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101
18	3.907185648	fe80::c09c:5634:7fc6:9710	#02::1:3	LLMNR	84	Standard query 0x6135 A wpad
19	3.907185831	192.168.32.101	224.0.0.252	LLMNR	64	Standard query 0x6135 A wpad
20	4.010940121	fe80::c09c:5634:7fc6:9710	#02::1:3	LLMNR	84	Standard query 0x6135 A wpad
21	4.010940365	192.168.32.101	224.0.0.252	LLMNR	64	Standard query 0x6135 A wpad
22	4.213997938	192.168.32.101	192.168.32.255	NBNS	92	Name query NB WPAD<00>
23	4.964028982	192.168.32.101	192.168.32.255	NBNS	92	Name query NB WPAD<00>
24	5.714579577	192.168.32.101	192.168.32.255	NBNS	92	Name query NB WPAD<00>
25	6.527199053	PcsCompu_65:21:99	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101
26	7.481504707	PcsCompu_65:21:99	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101

Frame 5: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface eth0, id 0  
 Ethernet II, Src: PcsCompu\_65:21:99 (08:00:27:65:21:99), Dst: PcsCompu\_cb:7e:f5 (08:00:27:cb:7e:f5)  
 Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100  
 Transmission Control Protocol, Src Port: 49158, Dst Port: 443, Seq: 0, Len: 0

0000 08 00 27 cb 7e f5 08 00 27 65 21 99 08 00 45 00 ... 'el...  
 0010 00 34 00 55 40 00 80 06 38 55 c0 a8 20 65 c0 a8 ... 4 U@...  
 0020 20 64 c0 06 01 bb 08 7f ec 5d 00 00 00 80 02 ... d...  
 0030 20 00 d6 5d 00 00 02 04 05 b4 01 03 03 02 01 01 ... ]...  
 0040 04 02

Packets: 26 · Displayed: 26 (100.0%) · Dropped: 0 (0.0%) Profile: Default



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

Kali\_Linux 2023.3 [In esecuzione] - Oracle VM VirtualBox

File Macchina Visualizza Inserimento Dispositivi Aiuto

\*eth0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-I>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu_65:21:99	Broadcast	ARP	60	Who has 192.168.32.100? Tell 192.168.32.101
2	0.000011189	PcsCompu_cb:7e:f5	PcsCompu_65:21:99	ARP	42	192.168.32.100 is at 08:00:27:cb:7e:f5
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
5	0.087217984	192.168.32.101	192.168.32.100	TCP	66	49158 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
6	0.087237481	192.168.32.100	192.168.32.101	TCP	66	443 → 49158 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
7	0.087582157	192.168.32.101	192.168.32.100	TCP	60	49158 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0
8	0.143896135	192.168.32.101	192.168.32.100	TLSv1	183	Client Hello
9	0.143909593	192.168.32.100	192.168.32.101	TCP	54	443 → 49158 [ACK] Seq=1 Ack=130 Win=64128 Len=0
10	0.283114350	192.168.32.100	192.168.32.101	TLSv1	1373	Server Hello, Certificate, Server Key Exchange, Server Hello Done
11	0.388198488	192.168.32.101	192.168.32.100	TLSv1	188	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
12	0.388222914	192.168.32.100	192.168.32.101	TCP	54	443 → 49158 [ACK] Seq=1320 Ack=264 Win=64128 Len=0
13	0.388956829	192.168.32.100	192.168.32.101	TLSv1	113	Change Cipher Spec, Encrypted Handshake Message
14	0.603658165	192.168.32.101	192.168.32.100	TCP	60	49158 → 443 [ACK] Seq=264 Ack=1379 Win=64320 Len=0
15	0.773258125	PcsCompu_65:21:99	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101
16	1.478528237	PcsCompu_65:21:99	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101
17	2.478675650	PcsCompu_65:21:99	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101
18	3.907185648	fe80::c09c:5634:7fc6:9710	ff02::1:3	LLMNR	84	Standard query 0x6135 A wpaad

Frame 8: 183 bytes on wire (1464 bits), 183 bytes captured (1464 bits) on interface eth0, id 0

Ethernet II, Src: PcsCompu\_65:21:99 (08:00:27:65:21:99), Dst: PcsCompu\_cb:7e:f5 (08:00:27:cb:7e:f5)

- Destination: PcsCompu\_cb:7e:f5 (08:00:27:cb:7e:f5)
- Source: PcsCompu\_65:21:99 (08:00:27:65:21:99)
- Type: IPv4 (0x0800)
- Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100
- Transmission Control Protocol, Src Port: 49158, Dst Port: 443, Seq: 1, Ack: 1, Len: 129
- Transport Layer Security

```

0000 08 00 27 cb 7e f5 08 00 27 65 21 99 08 00 45 00  ...~...!e!...E...
0010 00 a9 00 57 40 00 80 06 37 de c0 a8 20 65 c0 a8  ...W@... 7... e...
0020 20 64 c0 06 01 bb 08 7f ec 5e cf 9c fe 98 50 18  d...^...P...
0030 40 29 dd 4a 00 00 16 03 01 00 7c 01 00 00 78 03  @) J...|...x...
0040 01 65 58 8c 24 3c a9 c9 67 8d b1 34 58 93 3d 6b  .eX $<...g-4X=k...
0050 24 54 71 3e e5 d2 0d e5 c4 7a 99 7f bb 21 cd 0d  $Tq>...-z...!...
0060 2a 00 00 18 00 2f 00 35 00 05 00 0a c0 13 c0 14  *.../ 5...
0070 c0 09 c0 0a 00 32 00 38 00 13 00 04 01 00 00 37  ....2 8...7...
0080 ff 01 00 01 00 00 00 00 15 00 13 00 00 10 65 70  .......ep...
0090 69 63 6f 64 65 2e 69 6e 74 65 72 6e 61 6c 00 05  icode.in ternal...
00a0 00 05 01 00 00 00 00 00 0a 00 06 00 04 00 17 00  ....
00b0 18 00 0b 00 02 01 00  ....
  
```

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

```
▶ Frame 8: 183 bytes on wire (1464 bits), 183 bytes captured (1464 bits) on interface eth0, id 0
▶ Ethernet II, Src: PcsCompu_65:21:99 (08:00:27:65:21:99), Dst: PcsCompu_cb:7e:f5 (08:00:27:cb:7e:f5)
  ▶ Destination: PcsCompu_cb:7e:f5 (08:00:27:cb:7e:f5)
  ▶ Source: PcsCompu_65:21:99 (08:00:27:65:21:99)
    Type: IPV4 (0x0800)
  ▶ Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100
  ▶ Transmission Control Protocol, Src Port: 49158, Dst Port: 443, Seq: 1, Ack: 1, Len: 129
  ▶ Transport Layer Security
```

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**.

```
(kali@kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.32.100 netmask 255.255.255.0 broadcast 192.168.32.255
    inet6 fe80::a00:27ff:feeb:7ef5 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:cb:7e:f5 txqueuelen 1000 (Ethernet)
    RX packets 95 bytes 11370 (11.1 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 33 bytes 4309 (4.2 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

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**.

```
C:\Users\WinEpicode>ipconfig /all

Windows IP configuration

Host Name . . . . . : Windows7
Primary Dns Suffix . . . . . :
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No

Ethernet adapter Local Area Connection:

Connection-specific DNS Suffix . :
Description . . . . . : Intel(R) PRO/1000 MT Desktop Adapter
Physical Address. . . . . : 08-00-27-65-21-99
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::c09c:5634:7fc6:9710%11<Preferred>
IPv4 Address. . . . . : 192.168.32.101<Preferred>
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.32.1
DHCPv6 IAID . . . . . : 235405351
DHCPv6 Client DUID. . . . . : 00-01-00-01-2C-E1-B0-E0-08-00-27-65-21-99

DNS Servers . . . . . : 192.168.32.100
NetBIOS over Tcpip. . . . . : Enabled
```

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.

Kali\_Linux 2023.3 [In esecuzione] - Oracle VM VirtualBox

File Macchina Visualizza Inserimento Dispositivi Aiuto

\*eth0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu_65:21:99	Broadcast	ARP	60	Who has 192.168.32.100? Tell 192.168.32.101
2	0.000011189	PcsCompu_cb:7e:f5	PcsCompu_65:21:99	ARP	42	192.168.32.100 is at 08:00:27:cb:7e:f5
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
5	0.087217984	192.168.32.101	192.168.32.100	TCP	66	49158 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
6	0.087237481	192.168.32.100	192.168.32.101	TCP	66	443 → 49158 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
7	0.087582157	192.168.32.101	192.168.32.100	TCP	60	49158 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0
8	0.143896135	192.168.32.101	192.168.32.100	TLSv1	183	Client Hello
9	0.143909593	192.168.32.100	192.168.32.101	TCP	54	443 → 49158 [ACK] Seq=1 Ack=130 Win=64128 Len=0
10	0.283114350	192.168.32.100	192.168.32.101	TLSv1	1373	Server Hello, Certificate, Server Key Exchange, Server Hello Done
11	0.388198488	192.168.32.101	192.168.32.100	TLSv1	188	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
12	0.388222914	192.168.32.100	192.168.32.101	TCP	54	443 → 49158 [ACK] Seq=1320 Ack=264 Win=64128 Len=0
13	0.388956829	192.168.32.100	192.168.32.101	TLSv1	113	Change Cipher Spec, Encrypted Handshake Message
14	0.603658165	192.168.32.101	192.168.32.100	TCP	60	49158 → 443 [ACK] Seq=264 Ack=1379 Win=64320 Len=0
15	0.773258125	PcsCompu_65:21:99	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101
16	1.478528237	PcsCompu_65:21:99	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101
17	2.478675650	PcsCompu_65:21:99	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101
18	3.907185648	fe80::c09c:5634:7fc6:9710	ff02::1:3	LLMNR	84	Standard query 0x6135 A wpaad

Frame 10: 1373 bytes on wire (10984 bits), 1373 bytes captured (10984 bits) on interface eth0, id 0

Ethernet II, Src: PcsCompu\_cb:7e:f5 (08:00:27:cb:7e:f5), Dst: PcsCompu\_65:21:99 (08:00:27:65:21:99)

- Destination: PcsCompu\_65:21:99 (08:00:27:65:21:99)
- Source: PcsCompu\_cb:7e:f5 (08:00:27:cb:7e:f5)

Type: IPv4 (0x0800)

Internet Protocol Version 4, Src: 192.168.32.100, Dst: 192.168.32.101

Transmission Control Protocol, Src Port: 443, Dst Port: 49158, Seq: 1, Ack: 130, Len: 1319

Transport Layer Security

```

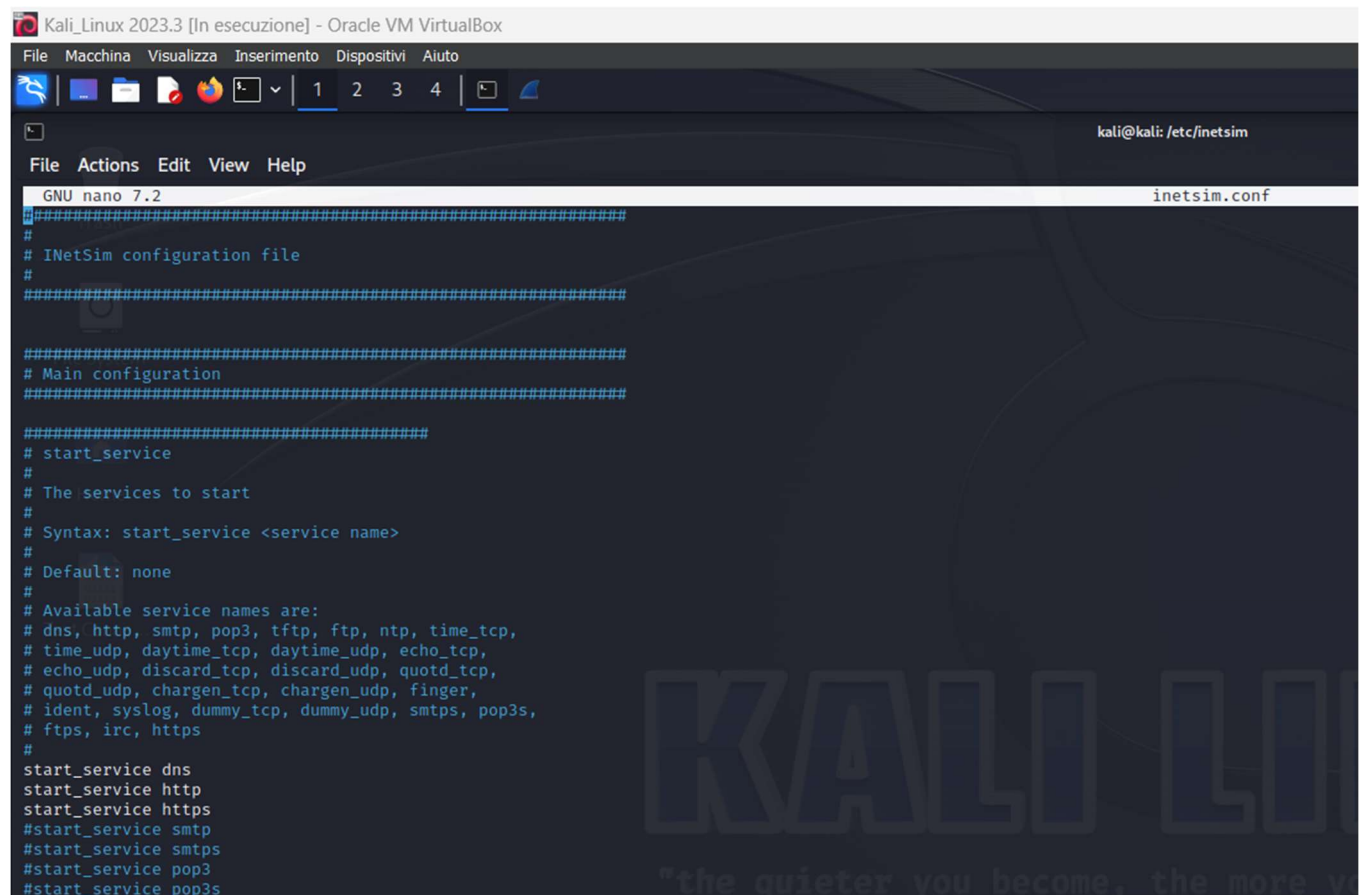
0000 08 00 27 65 21 99 08 00 27 cb 7e f5 08 00 45 00  ..e!...~...E
0010 05 4f b2 e9 40 00 40 06 c0 a5 c0 a8 20 64 c0 a8  ..O..@@...d..
0020 20 65 01 bb c0 06 cf 9c fe 98 08 7f ec df 50 18  e.....P.....
0030 01 f5 c7 5b 00 00 16 03 01 00 59 02 00 00 55 03  ...[.....Y...U
0040 01 44 51 00 85 1e 8b d6 89 f1 f9 80 7b 91 e3 d1  DQ.....{...
0050 c7 2f 5a 54 0a 31 74 d3 37 44 4f 57 4e 47 52 44  /ZT 1t 7DOWNGRD
0060 00 20 c7 1c 16 b8 77 01 10 14 4d 45 4d f8 88 69  ....w...MEM-i
0070 c6 79 68 57 44 e4 02 66 9f 41 cc b8 26 44 1f 9a  yhWD f A &D..
0080 c2 02 c0 14 00 00 0d ff 01 00 01 00 00 0b 00 04  ....k...q..d
0090 03 00 01 02 16 03 01 03 6b 0b 00 03 67 00 03 64

```

## 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 **#** sui servizi **start\_service https** come da schemata. Salvo la configurazione precedente e rieseguo il comando **sudo inetsim** per riattivare il simulatore.



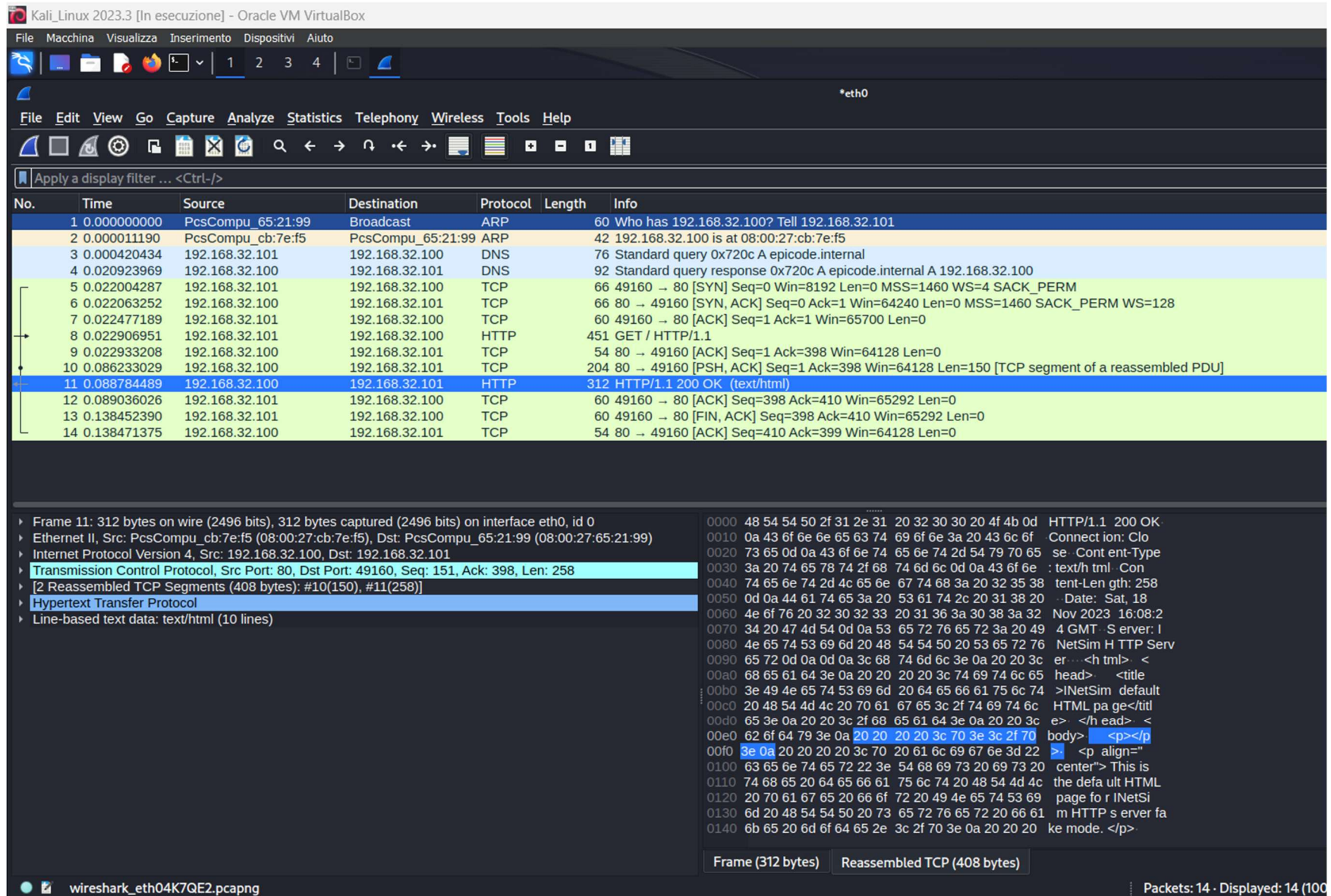
```
Kali_Linux 2023.3 [In esecuzione] - Oracle VM VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
GNU nano 7.2 inetsim.conf
#####
#
# InetSim configuration file
#
#####

#####
# 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
```



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.



Kali\_Linux 2023.3 [In esecuzione] - Oracle VM VirtualBox

File Macchina Visualizza Inserimento Dispositivi Aiuto

\*eth0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu_65:21:99	Broadcast	ARP	60	Who has 192.168.32.100? Tell 192.168.32.101
2	0.000011190	PcsCompu_cb:7e:f5	PcsCompu_65:21:99	ARP	42	192.168.32.100 is at 08:00:27:cb:7e:f5
3	0.000420434	192.168.32.101	192.168.32.100	DNS	76	Standard query 0x720c A epicode.internal
4	0.020923969	192.168.32.100	192.168.32.101	DNS	92	Standard query response 0x720c A epicode.internal A 192.168.32.100
5	0.022004287	192.168.32.101	192.168.32.100	TCP	66	49160 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
6	0.022063252	192.168.32.100	192.168.32.101	TCP	66	80 → 49160 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
7	0.022477189	192.168.32.101	192.168.32.100	TCP	60	49160 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
8	0.022906951	192.168.32.101	192.168.32.100	HTTP	451	GET / HTTP/1.1
9	0.022933208	192.168.32.100	192.168.32.101	TCP	54	80 → 49160 [ACK] Seq=1 Ack=398 Win=64128 Len=0
10	0.086233029	192.168.32.100	192.168.32.101	TCP	204	80 → 49160 [PSH, ACK] Seq=1 Ack=398 Win=64128 Len=150 [TCP segment of a reassembled PDU]
11	0.088784489	192.168.32.100	192.168.32.101	HTTP	312	HTTP/1.1 200 OK (text/html)
12	0.089036026	192.168.32.101	192.168.32.100	TCP	60	49160 → 80 [ACK] Seq=398 Ack=410 Win=65292 Len=0
13	0.138452390	192.168.32.101	192.168.32.100	TCP	60	49160 → 80 [FIN, ACK] Seq=398 Ack=410 Win=65292 Len=0
14	0.138471375	192.168.32.100	192.168.32.101	TCP	54	80 → 49160 [ACK] Seq=410 Ack=399 Win=64128 Len=0

Frame 11: 312 bytes on wire (2496 bits), 312 bytes captured (2496 bits) on interface eth0, id 0

Ethernet II, Src: PcsCompu\_cb:7e:f5 (08:00:27:cb:7e:f5), Dst: PcsCompu\_65:21:99 (08:00:27:65:21:99)

Internet Protocol Version 4, Src: 192.168.32.100, Dst: 192.168.32.101

Transmission Control Protocol, Src Port: 80, Dst Port: 49160, Seq: 151, Ack: 398, Len: 258

[2 Reassembled TCP Segments (408 bytes): #10(150), #11(258)]

Hypertext Transfer Protocol

Line-based text data: text/html (10 lines)

```

0000 48 54 54 50 2f 31 2e 31 20 32 30 30 20 4f 4b 0d HTTP/1.1 200 OK
0010 0a 43 6f 6e 6e 65 63 74 69 6f 6e 3a 20 43 6c 6f Connection: Clo
0020 73 65 0d 0a 43 6f 6e 74 65 6e 74 2d 54 79 70 65 se: Content-Type
0030 3a 20 74 65 78 74 2f 68 74 6d 6c 0d 0a 43 6f 6e : text/html; Con
0040 74 65 6e 74 2d 4c 65 6e 67 74 68 3a 20 32 35 38 tent-Length: 258
0050 0d 0a 44 61 74 65 3a 20 53 61 74 2c 20 31 38 20 -Date: Sat, 18
0060 4e 6f 76 20 32 30 32 33 20 31 36 3a 30 38 3a 32 Nov 2023 16:08:2
0070 34 20 47 4d 54 0d 0a 53 65 72 76 65 72 3a 20 49 4 GMT -Server: I
0080 4e 65 74 53 69 6d 20 48 54 54 50 20 53 65 72 76 NetSim HTTP Serv
0090 65 72 0d 0a 0d 0a 3e 68 74 6d 6c 3e 0a 20 20 3c er...<html> <
00a0 68 65 61 64 3e 0a 20 20 20 3c 74 69 74 6c 65 head> <title
00b0 3e 49 4e 65 74 53 69 6d 20 64 65 66 61 75 6c 74 >INetSim default
00c0 20 48 54 4d 4c 20 70 61 67 65 3c 2f 74 69 74 6c HTML page</titl
00d0 65 3e 0a 20 20 3c 2f 68 65 61 64 3e 0a 20 20 3c e> </head> <
00e0 62 6f 64 79 3e 0a 20 20 20 3c 70 3e 3c 2f 70 body> <p></p>
00f0 3e 0a 20 20 20 20 3c 70 20 61 6c 69 67 6e 3d 22 > <p align="
0100 63 65 6e 74 65 72 22 3e 54 68 69 73 20 69 73 20 center"> This is
0110 74 68 65 20 64 65 66 61 75 6c 74 20 48 54 4d 4c the default HTML
0120 20 70 61 67 65 20 66 6f 72 20 49 4e 65 74 53 69 page for INetSi
0130 6d 20 48 54 54 50 20 73 65 72 76 65 72 20 66 61 m HTTP server fa
0140 6b 65 20 6d 6f 64 65 2e 3c 2f 70 3e 0a 20 20 20 ke mode. </p>
  
```

Frame (312 bytes) Reassembled TCP (408 bytes)

wireshark\_eth04K7QE2.pcapng

Packets: 14 · Displayed: 14 (100%)

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.

11	0.088784489	192.168.32.100	192.168.32.101	HTTP	312 HTTP/1.1 200 OK (text/html)
12	0.089036026	192.168.32.101	192.168.32.100	TCP	60 49160 → 80 [ACK] Seq=398 Ack=410 Win=65292 Len=0
13	0.138452390	192.168.32.101	192.168.32.100	TCP	60 49160 → 80 [FIN, ACK] Seq=398 Ack=410 Win=65292 Len=0
14	0.138471375	192.168.32.100	192.168.32.101	TCP	54 80 → 49160 [ACK] Seq=410 Ack=399 Win=64128 Len=0

▶ Frame 11: 312 bytes on wire (2496 bits), 312 bytes captured (2496 bits) on interface eth0, id 0 ▶ Ethernet II, Src: PcsCompu_cb:7e:f5 (08:00:27:cb:7e:f5), Dst: PcsCompu_65:21:99 (08:00:27:65:21:99) ▶ Internet Protocol Version 4, Src: 192.168.32.100, Dst: 192.168.32.101 ▶ Transmission Control Protocol, Src Port: 80, Dst Port: 49160, Seq: 151, Ack: 398, Len: 258 ▶ [2 Reassembled TCP Segments (408 bytes): #10(150), #11(258)] ▶ Hypertext Transfer Protocol ▶ Line-based text data: text/html (10 lines)	<pre> 0000 48 54 54 50 2f 31 2e 31 20 32 30 30 20 4f 4b 0d HTTP/1.1 200 OK 0010 0a 43 6f 6e 6e 65 63 74 69 6f 6e 3a 20 43 6c 6f Connect ion: Clo 0020 73 65 0d 0a 43 6f 6e 74 65 6e 74 2d 54 79 70 65 se Cont ent-Type 0030 3a 20 74 65 78 74 2f 68 74 6d 6c 0d 0a 43 6f 6e : text/h tml Con 0040 74 65 6e 74 2d 4c 65 6e 67 74 68 3a 20 32 35 38 tent-Len gth: 258 0050 0d 0a 44 61 74 65 3a 20 53 61 74 2c 20 31 38 20 . Date: Sat, 18 0060 4e 6f 76 20 32 30 32 33 20 31 36 3a 30 38 3a 32 Nov 2023 16:08:2 0070 34 20 47 4d 54 0d 0a 53 65 72 76 65 72 3a 20 49 4 GMT- S erver: I 0080 4e 65 74 53 69 6d 20 48 54 54 50 20 53 65 72 76 NetSim H TTP Serv 0090 65 72 0d 0a 0d 0a 3c 68 74 6d 6c 3e 0a 20 20 3c er &lt;h tml&gt; &lt; 00a0 68 65 61 64 3e 0a 20 20 20 20 3c 74 69 74 6c 65 head&gt; &lt;title 00b0 3e 49 4e 65 74 53 69 6d 20 64 65 66 61 75 6c 74 &gt;INetSim default 00c0 20 48 54 4d 4c 20 70 61 67 65 3c 2f 74 69 74 6c HTML pa ge&lt;titl 00d0 65 3e 0a 20 20 3c 2f 68 65 61 64 3e 0a 20 20 3c e&gt; &lt;/h ead&gt; &lt; 00e0 62 6f 64 79 3e 0a 20 20 20 20 3c 70 3e 3c 2f 70 body&gt; &lt;p&gt;&lt;p 00f0 3e 0a 20 20 20 20 3c 70 20 61 6c 69 67 6e 3d 22 &gt; &lt;p align=" 0100 63 65 6e 74 65 72 22 3e 54 68 69 73 20 69 73 20 center"&gt; This is 0110 74 68 65 20 64 65 66 61 75 6c 74 20 48 54 4d 4c the defa ult HTML 0120 20 70 61 67 65 20 66 6f 72 20 49 4e 65 74 53 69 page fo r INetSi 0130 6d 20 48 54 54 50 20 73 65 72 76 65 72 20 66 61 m HTTP s erver fa 0140 6b 65 20 6d 6f 64 65 2e 3c 2f 70 3e 0a 20 20 20 ke mode. &lt;/p&gt; </pre>
--	---



- **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.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu_65:21:99	Broadcast	ARP	60	Who has 192.168.32.100? Tell 192.168.32.101
2	0.000011189	PcsCompu_cb:7e:f5	PcsCompu_65:21:99	ARP	42	192.168.32.100 is at 08:00:27:cb:7e:f5
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
5	0.087217984	192.168.32.101	192.168.32.100	TCP	66	49158 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
6	0.087237481	192.168.32.100	192.168.32.101	TCP	66	443 → 49158 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
7	0.087582157	192.168.32.101	192.168.32.100	TCP	60	49158 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0
8	0.143896135	192.168.32.101	192.168.32.100	TLSv1	183	Client Hello
9	0.143909593	192.168.32.100	192.168.32.101	TCP	54	443 → 49158 [ACK] Seq=1 Ack=130 Win=64128 Len=0
10	0.283114350	192.168.32.100	192.168.32.101	TLSv1	1373	Server Hello, Certificate, Server Key Exchange, Server Hello Done
11	0.388198488	192.168.32.101	192.168.32.100	TLSv1	188	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
12	0.388222914	192.168.32.100	192.168.32.101	TCP	54	443 → 49158 [ACK] Seq=1320 Ack=264 Win=64128 Len=0
13	0.388956829	192.168.32.100	192.168.32.101	TLSv1	113	Change Cipher Spec, Encrypted Handshake Message
14	0.603658165	192.168.32.101	192.168.32.100	TCP	60	49158 → 443 [ACK] Seq=264 Ack=1379 Win=64320 Len=0
15	0.773258125	PcsCompu_65:21:99	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101
16	1.478528237	PcsCompu_65:21:99	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101
17	2.478675650	PcsCompu_65:21:99	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101
18	3.907185648	fe80::c09c:5634:7fc6:9710	ff02::1:3	LLMNR	84	Standard query 0x6135 A wpad

<ul style="list-style-type: none"> <li>▶ Frame 10: 1373 bytes on wire (10984 bits), 1373 bytes captured (10984 bits) on interface eth0, id 0</li> <li>▶ Ethernet II, Src: PcsCompu_cb:7e:f5 (08:00:27:cb:7e:f5), Dst: PcsCompu_65:21:99 (08:00:27:65:21:99) <ul style="list-style-type: none"> <li>▶ Destination: PcsCompu_65:21:99 (08:00:27:65:21:99)</li> <li>▶ Source: PcsCompu_cb:7e:f5 (08:00:27:cb:7e:f5)</li> <li>Type: IPv4 (0x0800)</li> </ul> </li> <li>▶ Internet Protocol Version 4, Src: 192.168.32.100, Dst: 192.168.32.101</li> <li>▶ Transmission Control Protocol, Src Port: 443, Dst Port: 49158, Seq: 1, Ack: 130, Len: 1319</li> <li>▶ Transport Layer Security</li> </ul>	<pre> 0000 08 00 27 65 21 99 08 00 27 cb 7e f5 08 00 45 00  ..e!...~...E. 0010 05 4f b2 e9 40 00 40 06 c0 a5 c0 a8 20 64 c0 a8  ..O @ @ ...d . 0020 20 65 01 bb c0 06 cf 9c fe 98 08 7f ec df 50 18  e.....P 0030 01 f5 c7 5b 00 00 16 03 01 00 59 02 00 00 55 03  ...[...Y...U 0040 01 44 51 00 85 1e 8b d6 89 f1 f9 80 7b 91 e3 d1  .DQ.....{.. 0050 c7 2f 5a 54 0a 31 74 d3 37 44 4f 57 4e 47 52 44  ./ZT 1t 7DOWNGRD 0060 00 20 c7 1c 16 b8 77 01 10 14 4d 45 4d f8 88 69  ....w .MEM .i 0070 c6 79 68 57 44 e4 02 66 9f 41 cc b8 26 44 1f 9a  .yhWD .f .A .&amp;D.. 0080 c2 02 c0 14 00 00 0d ff 01 00 01 00 00 0b 00 04  ....k...a...d 0090 03 00 01 02 16 03 01 03 6b 0b 00 03 67 00 03 64  .... </pre>
---	--

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.

10	0.214957074	192.168.32.100	192.168.32.101	TLSv1	13/3 Server Hello, Certificate, Server Key Exchange, Server Hello Done
11	0.311864350	192.168.32.101	192.168.32.100	TLSv1	188 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
12	0.311892530	192.168.32.100	192.168.32.101	TCP	54 443 → 49161 [ACK] Seq=1320 Ack=264 Win=64128 Len=0
13	0.312380640	192.168.32.100	192.168.32.101	TLSv1	113 Change Cipher Spec, Encrypted Handshake Message
14	0.519112548	192.168.32.101	192.168.32.100	TCP	60 49161 → 443 [ACK] Seq=264 Ack=1379 Win=64320 Len=0
15	0.681931192	PcsCompu_65:21:99	Broadcast	ARP	60 Who has 192.168.32.1? Tell 192.168.32.101
16	1.659849751	PcsCompu_65:21:99	Broadcast	ARP	60 Who has 192.168.32.1? Tell 192.168.32.101

<ul style="list-style-type: none"> <li>▶ Frame 11: 188 bytes on wire (1504 bits), 188 bytes captured (1504 bits) on interface eth0, id 0</li> <li>▶ Ethernet II, Src: PcsCompu_65:21:99 (08:00:27:65:21:99), Dst: PcsCompu_cb:7e:f5 (08:00:27:cb:7e:f5)</li> <li>▶ Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100</li> <li>▶ Transmission Control Protocol, Src Port: 49161, Dst Port: 443, Seq: 130, Ack: 1320, Len: 134</li> <li>▶ Transport Layer Security</li> </ul>	<pre> 0000 08 00 27 cb 7e f5 08 00 27 65 21 99 08 00 45 00  ...~... 'e!...E. 0010 00 ae 00 7b 40 00 80 06 37 b5 c0 a8 20 65 c0 a8  ...{ @ 7... e. 0020 20 64 c0 09 01 bb f8 1c 47 53 79 fe 0c 34 50 18  d....GSy 4P. 0030 3e df 46 d9 00 00 16 03 01 00 46 10 00 00 42 41  &gt; F....F...BA 0040 04 9c 32 98 83 d4 00 5e 05 52 68 c4 a9 b3 96 92  .2...^ .Rh.... 0050 af 6d 7c ea 5b 27 e1 0d f8 57 66 23 d7 6a 45 b5  m  [.. Wf# jE. 0060 4c 94 3d 75 9f 0e 70 9c 4d e5 3a ff 9d 39 2e 90  L =u .p M.: 9. 0070 83 b0 06 f6 40 19 53 6f cc 5a 82 56 0e ab f0 11  ....@ So Z V... 0080 ba 14 03 01 00 01 01 16 03 01 00 30 4c 57 a8 54  .......OLW T 0090 5c c3 4f db eb 4f 92 c9 e6 ba d5 ea 89 43 5c 37  \ O . O .....C7 00a0 a0 45 22 8b 25 d2 a9 7f ad d5 c9 0a e4 af 90 8b  E"% ..... 00b0 77 e0 7c 50 34 ec bf bd 4f af 5f 20          w  P4 .. O _ </pre>
---	---

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.