

19/11/2023

Ilaria Pedrelli

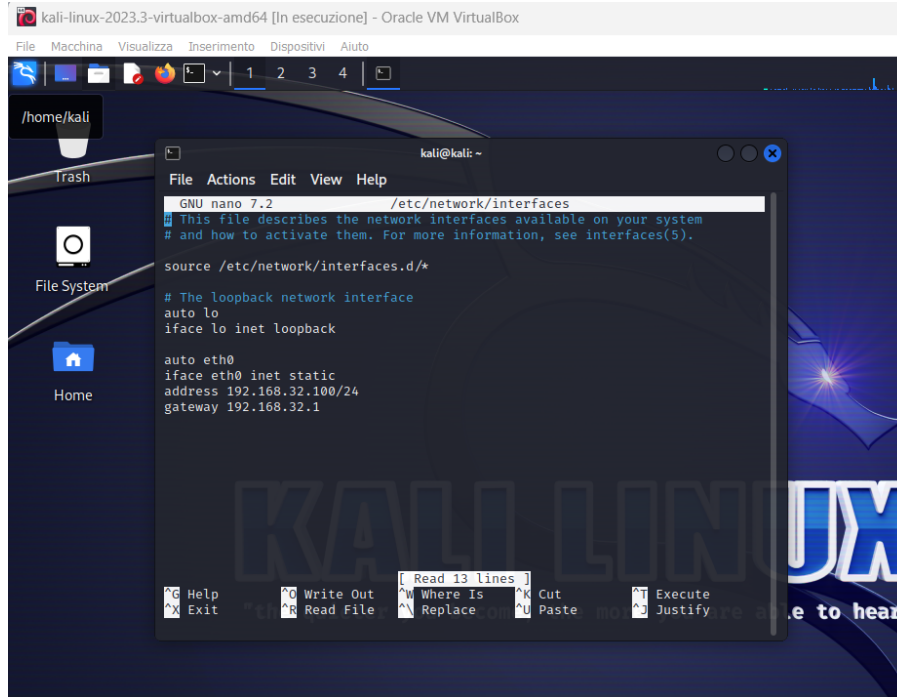
# Esercitazione di fine modulo

## 1. Configurazione indirizzi IP

Inizio con il configurare gli indirizzi IP delle macchine Virtuali Kali Linux e Win7 con gli indirizzi IP assegnati:

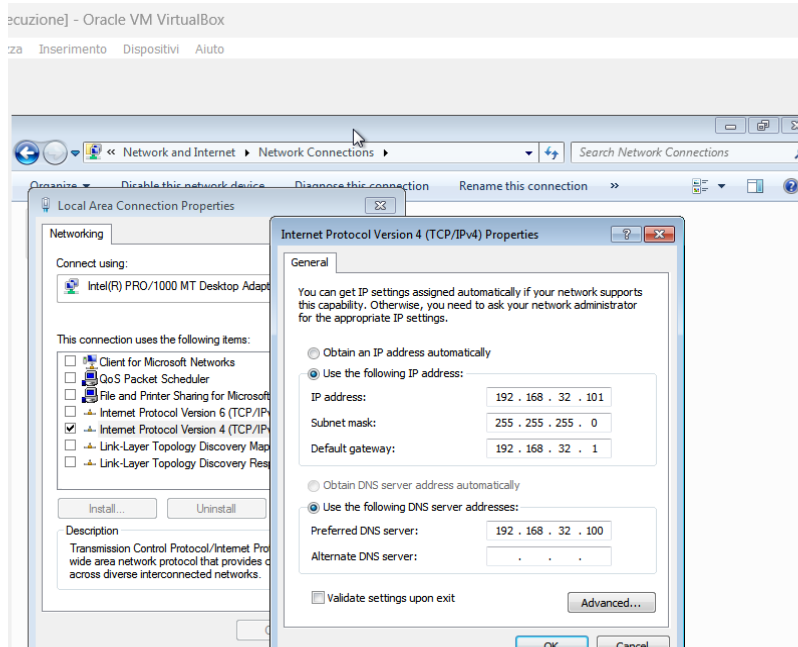
Kali Linux: IP 192.168.32.100

Con il comando **sudo nano /etc/network/interfaces** vado a cambiare l'indirizzo IP di kali



```
kali@kali: ~  
File Actions Edit View Help  
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
```

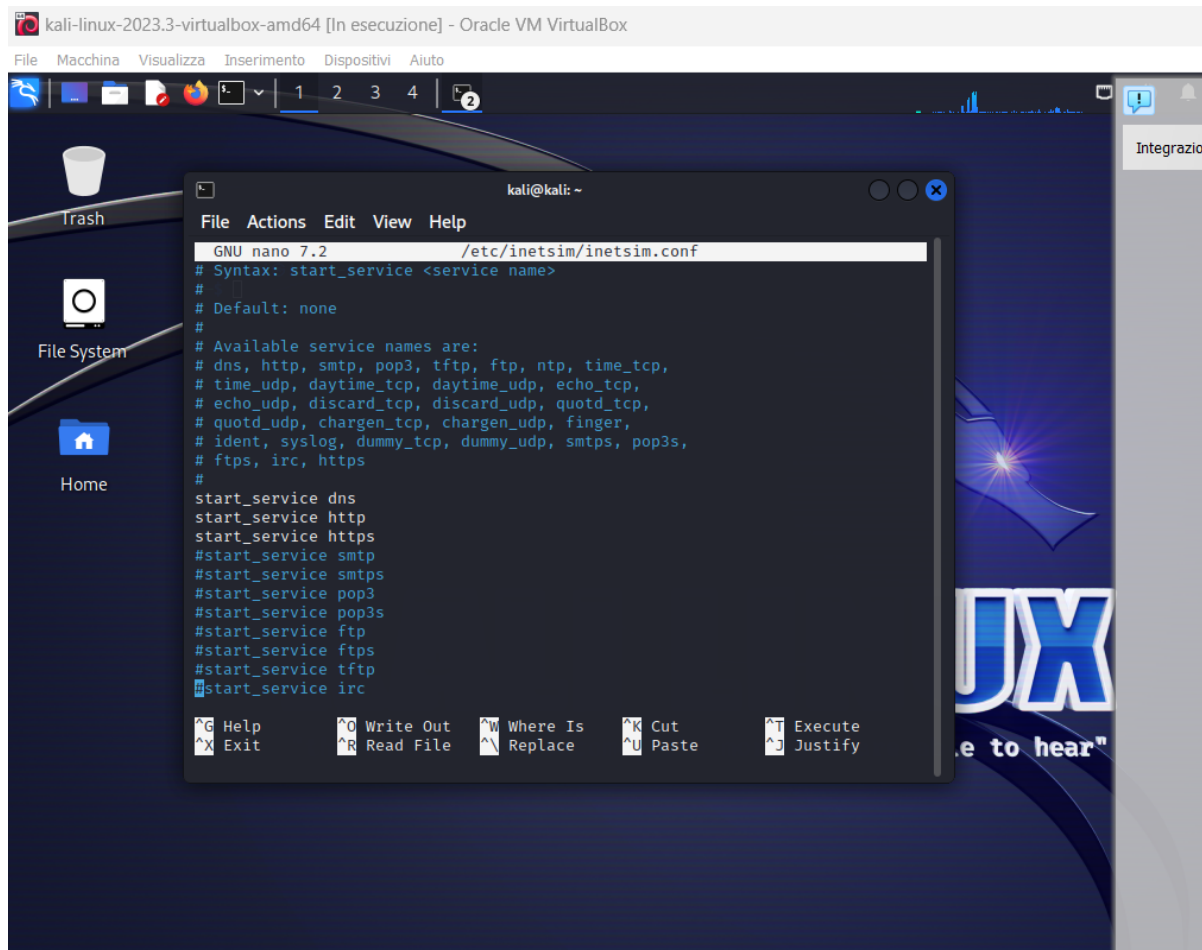
Faccio la stessa cosa con Win 7 assegnando l'indirizzo IP 192.168.32.101 ma con il seguente percorso: **control panel** → **Network and Internet** → **Change adapter settings** → **seleziono scheda di rete e modifico indirizzo IP statico**. Assegno anche come DNS preferito mettendo l'indirizzo IP di Kali.



## 2. Configurazione INetSim

Passo alla macchina Kali e configuro il software INetSim:

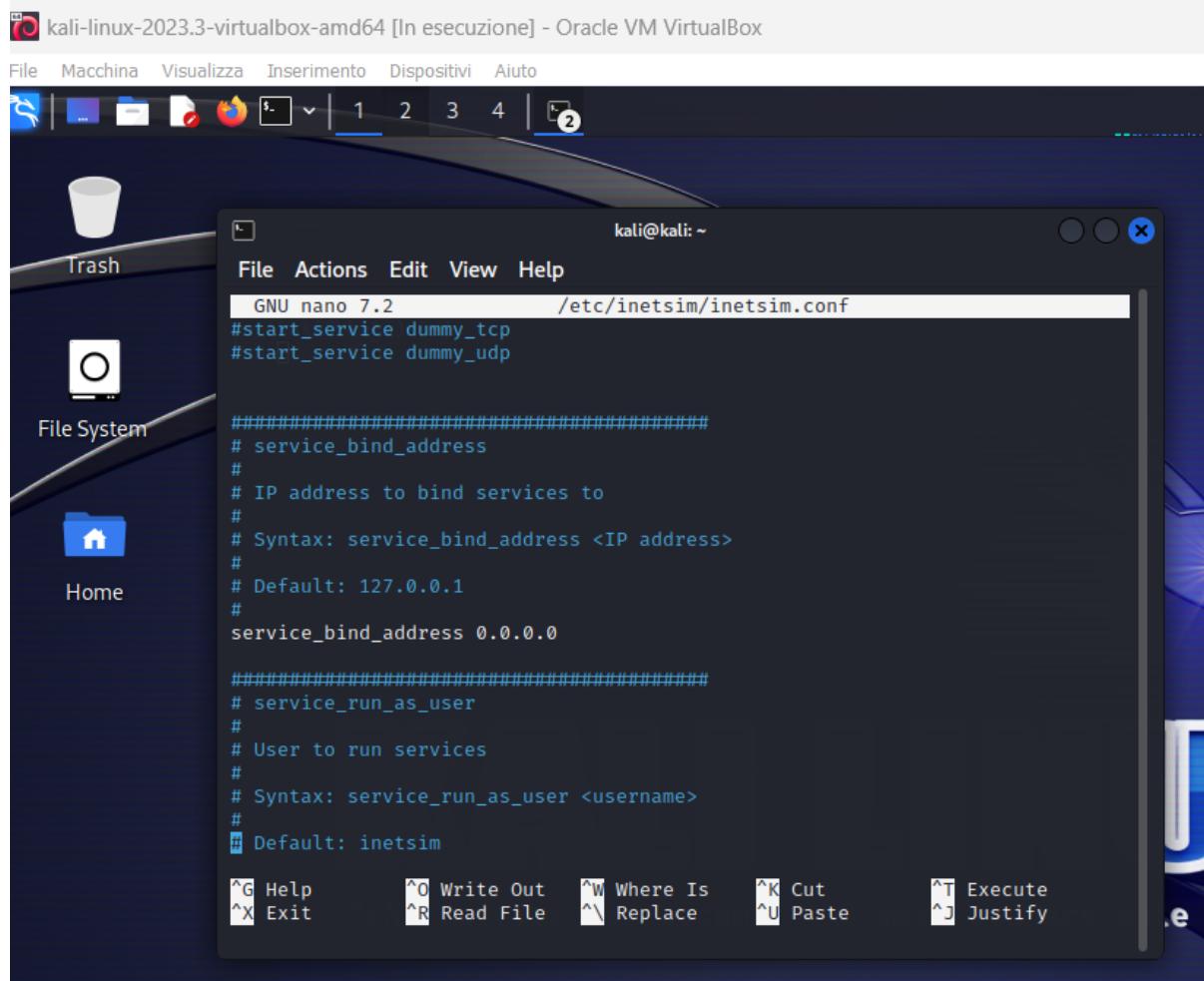
Attivo i servizi che mi servono ai fini dell'esercizio quindi HTTP, HTTPS e il Server DNS.



The screenshot shows a Kali Linux virtual machine running in Oracle VM VirtualBox. The terminal window is open, displaying the configuration of INetSim services. The configuration file is located at `/etc/inetsim/inetsim.conf`. The terminal output shows the following configuration:

```
GNU nano 7.2 /etc/inetsim/inetsim.conf
# 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
```

Successivamente vado ad attivare anche `service_bind_address` e lo imposto a `0.0.0.0` in modo da rendere disponibili tutte le interfacce di rete. Questo rende il servizio accessibile da qualsiasi indirizzo IP.



```
kali-linux-2023.3-virtualbox-amd64 [In esecuzione] - Oracle VM VirtualBox
File  Macchina  Visualizza  Inserimento  Dispositivi  Aiuto
1 2 3 4 5
Trash
File System
Home

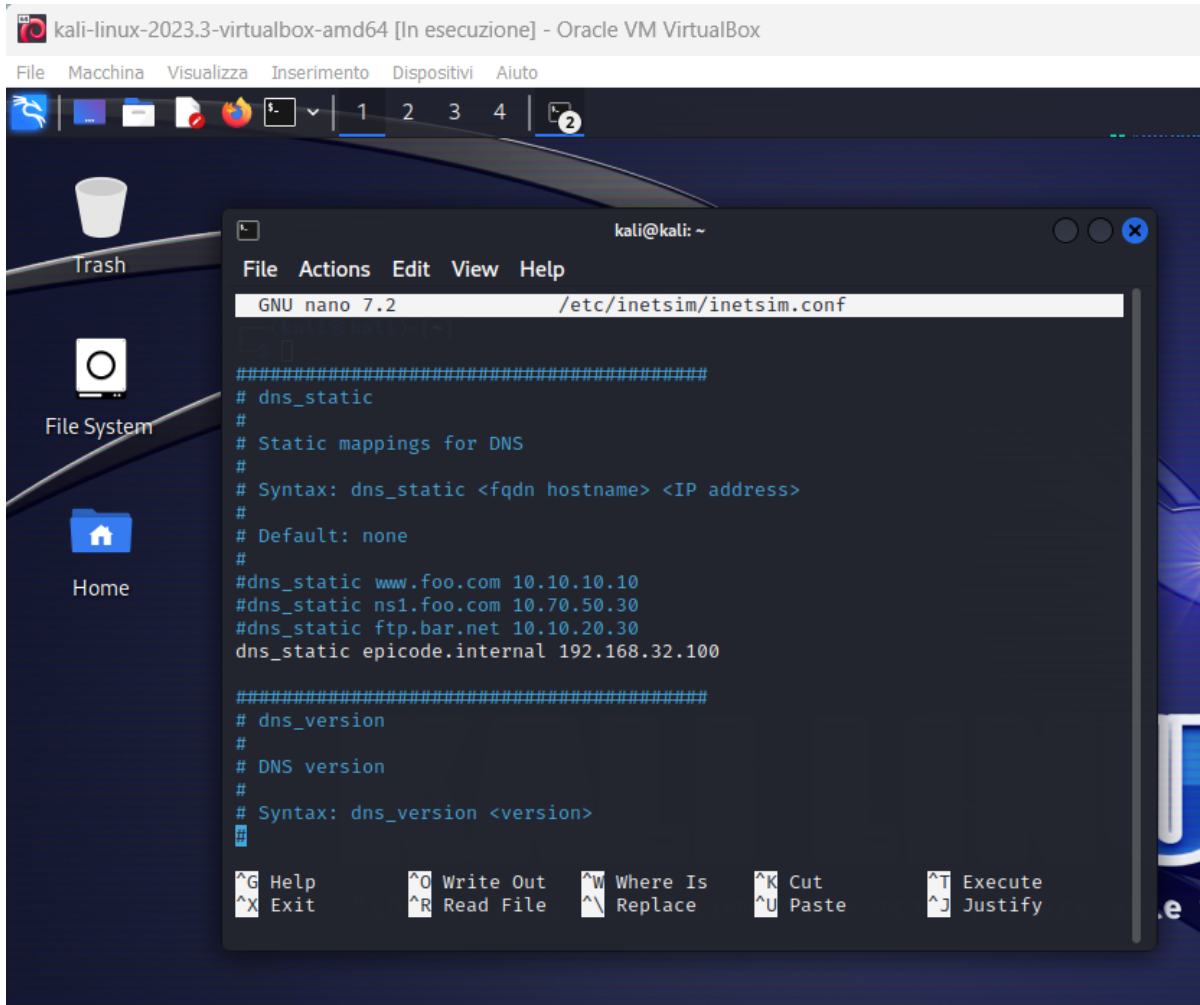
kali@kali: ~
File Actions Edit View Help
GNU nano 7.2 /etc/inetsim/inetsim.conf
#start_service dummy_tcp
#start_service dummy_udp

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

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

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute
^X Exit      ^R Read File  ^N Replace    ^U Paste      ^J Justify
```

Infine, configuro il server DNS e aggiungo un nuovo indirizzo IP statico associato a **epicode.internal**.

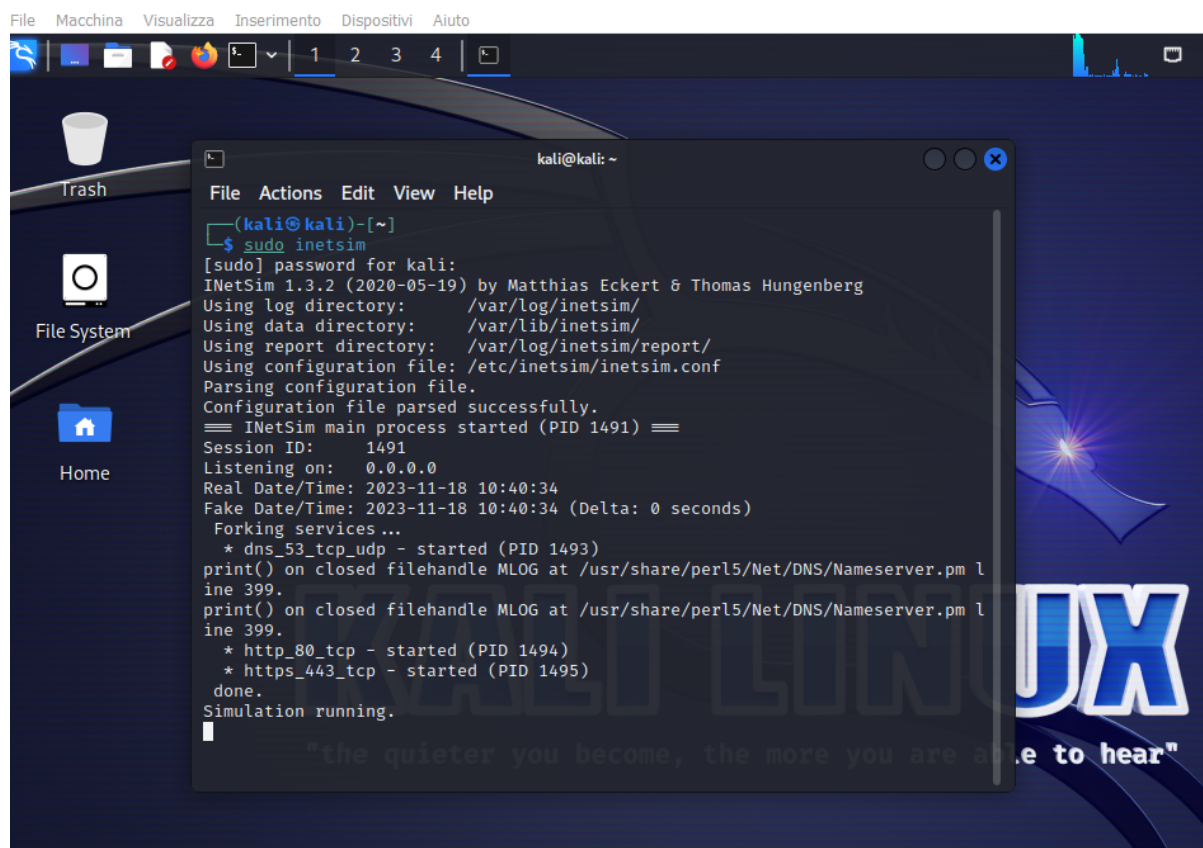


```
kali@kali: ~
File Actions Edit View Help
GNU nano 7.2 /etc/inetsim/inetsim.conf

#####
# 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_version
#
# DNS version
#
# Syntax: dns_version <version>
#
```

In conclusione faccio partire Inetsim con il comando **sudo inetsim**.



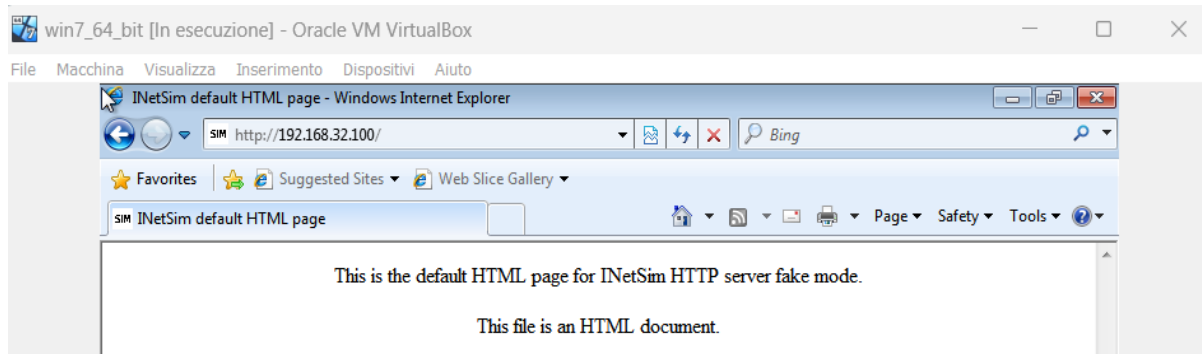
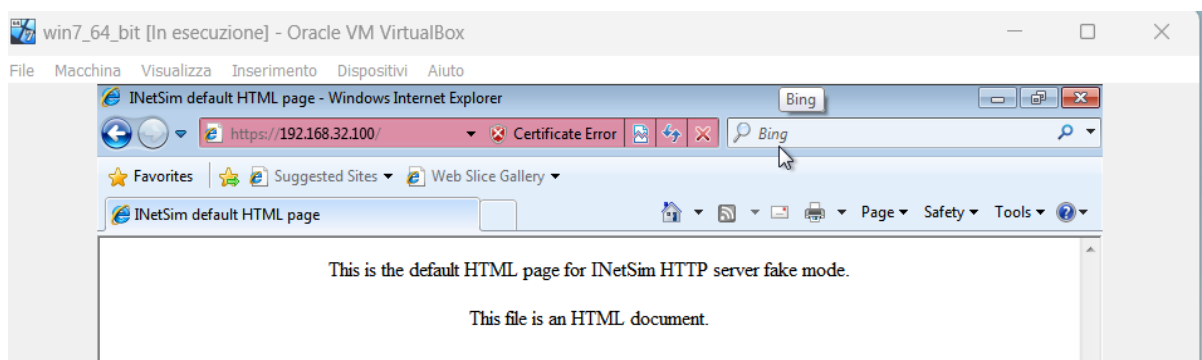
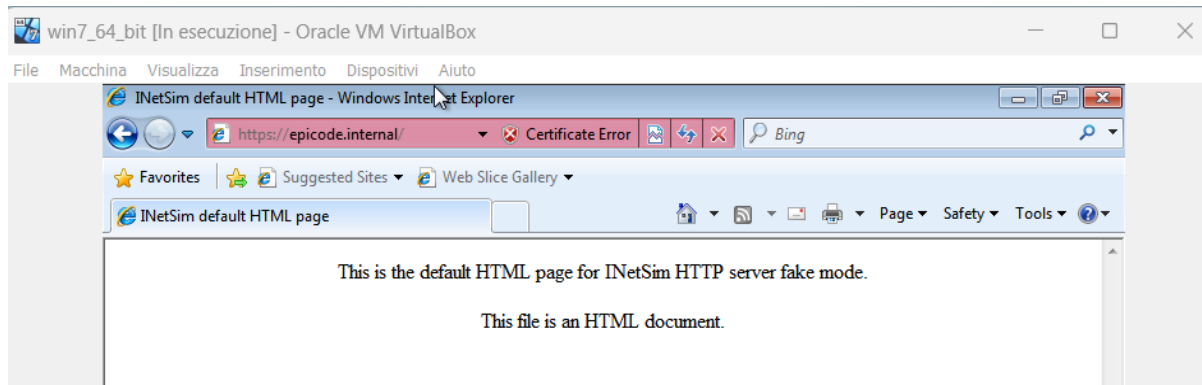
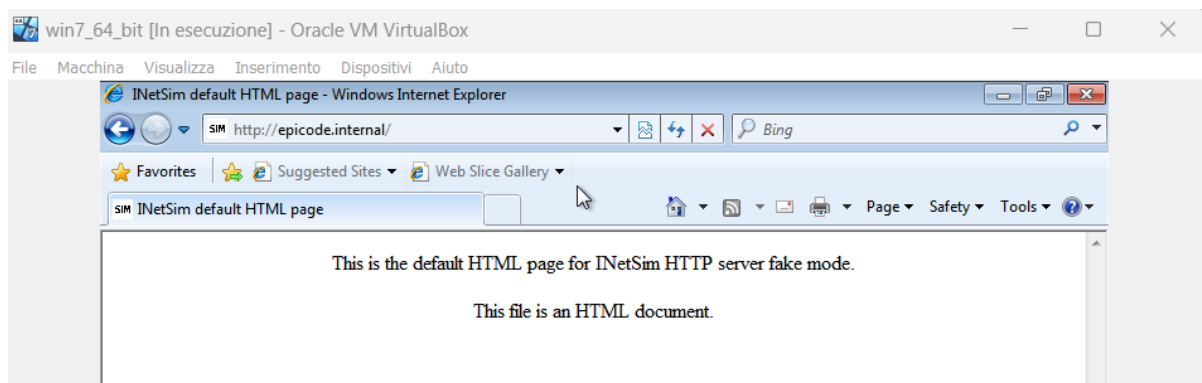
```
kali@kali: ~  
File Actions Edit View Help  
[kali@kali]~  
$ sudo inetsim  
[sudo] password for kali:  
INetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg  
Using log directory: /var/log/inetsim/  
Using data directory: /var/lib/inetsim/  
Using report directory: /var/log/inetsim/report/  
Using configuration file: /etc/inetsim/inetsim.conf  
Parsing configuration file.  
Configuration file parsed successfully.  
== INetSim main process started (PID 1491) ==  
Session ID: 1491  
Listening on: 0.0.0.0  
Real Date/Time: 2023-11-18 10:40:34  
Fake Date/Time: 2023-11-18 10:40:34 (Delta: 0 seconds)  
Forking services ...  
* dns_53_tcp_udp - started (PID 1493)  
print() on closed filehandle MLOG at /usr/share/perl5/Net/DNS/Nameserver.pm l  
ine 399.  
print() on closed filehandle MLOG at /usr/share/perl5/Net/DNS/Nameserver.pm l  
ine 399.  
* http_80_tcp - started (PID 1494)  
* https_443_tcp - started (PID 1495)  
done.  
Simulation running.  
|
```

Dopo una prima prova, dove in win 7 non venivano risolti gli indirizzi **https://epicode.internal** e **https://192.168.32.100**, provo "stoppando" inetsim e facendo un restart con i comandi **Service inetsim stop** e successivamente rilancio Inetsim.

Verifico poi, con il comando Ping e l'indirizzo IP di win7 se comunicano.

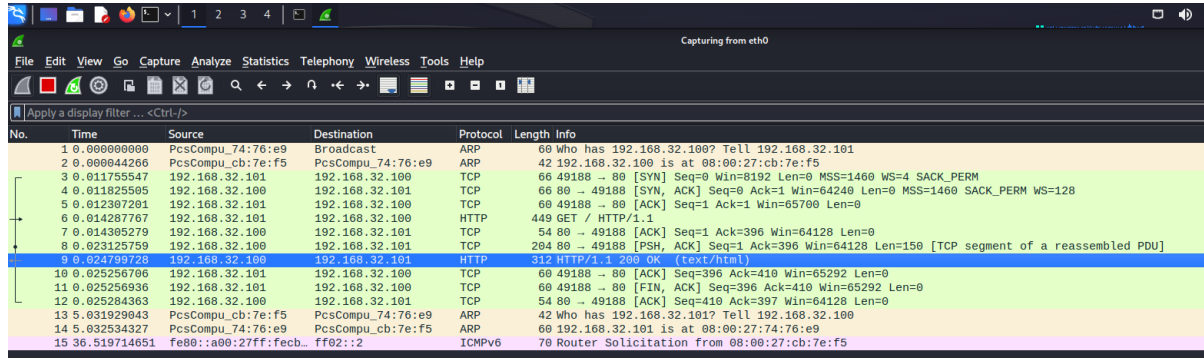
Dal browser di Win7 invio una richiesta all'indirizzo IP **https://192.168.32.100** e viene risolta. Faccio la stessa richiesta con l'indirizzo **https://epicode.ineternal** ma non viene trovato. Provo allora a riavviare il servizio di rete da Inetsim con il comando **sudo systemctl restart networking**.

Riprovo dal browser di Win7 la ricerca http e https con 192.168.32.100 e epicode.internal e vengono risolti



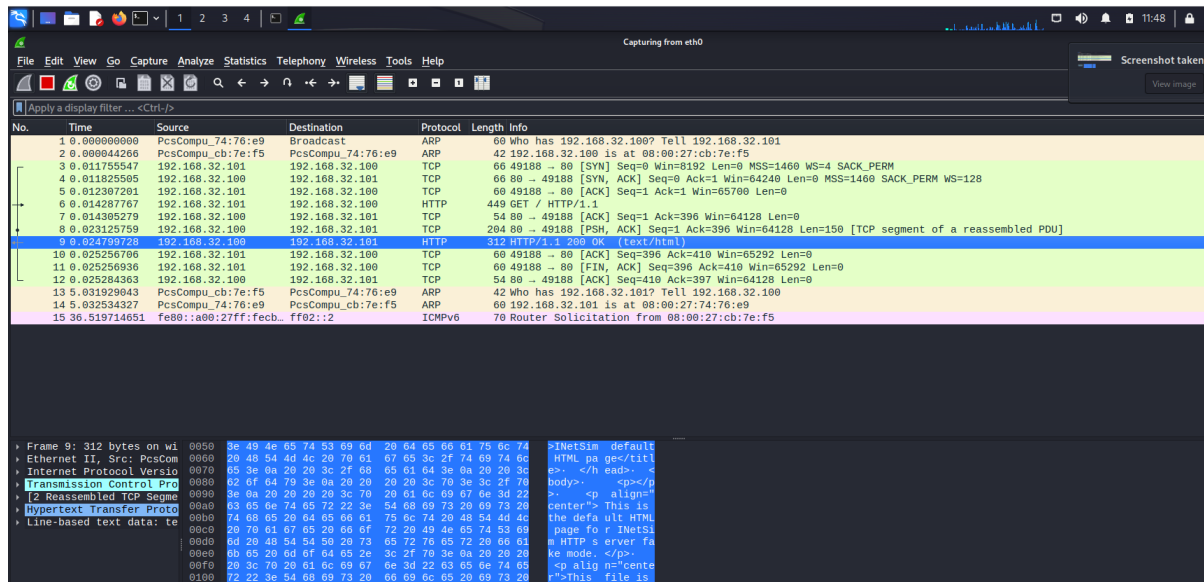
### 3. CATTURA DEI PACCHETTI DA WIRESHARK

Inizio con i pacchetti HTTP in cui posso notare una richiesta GET e posso visualizzare la pagina in HTML nella risposta del server http 200.



The screenshot shows the Wireshark interface with a packet capture from interface eth0. The packet list on the left shows 15 packets. Packet 9 is selected, which is an HTTP GET request. The packet details pane on the right shows the structure of the packet, including Ethernet II, Internet Protocol Version 4, and Hypertext Transfer Protocol. The packet bytes pane at the bottom shows the raw data of the selected packet.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu_74:76:e9	Broadcast	ARP	60	Who has 192.168.32.100? Tell 192.168.32.101
2	0.000044266	PcsCompu_cb:7e:f5	PcsCompu_74:76:e9	ARP	42	192.168.32.100 is at 08:00:27:cb:7e:f5
3	0.011755547	192.168.32.101	192.168.32.100	TCP	60	49188 -> 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
4	0.011825505	192.168.32.100	192.168.32.101	TCP	60	80 -> 49188 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
5	0.012307201	192.168.32.101	192.168.32.100	TCP	60	49188 -> 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
6	0.014287767	192.168.32.101	192.168.32.100	HTTP	449	GET / HTTP/1.1
7	0.014305279	192.168.32.100	192.168.32.101	TCP	54	80 -> 49188 [ACK] Seq=1 Ack=396 Win=64128 Len=0
8	0.023125759	192.168.32.100	192.168.32.101	TCP	204	80 -> 49188 [PSH, ACK] Seq=1 Ack=396 Win=64128 Len=150 [TCP segment of a reassembled PDU]
9	0.024799728	192.168.32.100	192.168.32.101	HTTP	312	HTTP/1.1 200 OK (text/html)
10	0.025256706	192.168.32.101	192.168.32.100	TCP	60	49188 -> 80 [ACK] Seq=396 Ack=410 Win=65292 Len=0
11	0.025256936	192.168.32.101	192.168.32.100	TCP	60	49188 -> 80 [FIN, ACK] Seq=396 Ack=410 Win=65292 Len=0
12	0.025284363	192.168.32.100	192.168.32.101	TCP	54	80 -> 49188 [ACK] Seq=410 Ack=397 Win=64128 Len=0
13	5.031929043	PcsCompu_cb:7e:f5	PcsCompu_74:76:e9	ARP	42	Who has 192.168.32.101? Tell 192.168.32.100
14	5.032534327	PcsCompu_74:76:e9	PcsCompu_cb:7e:f5	ARP	60	192.168.32.101 is at 08:00:27:74:76:e9
15	36.519714651	fe80::a00:27ff:feeb::ff02::2	ff02::2	ICMPv6	70	Router Solicitation from 08:00:27:cb:7e:f5



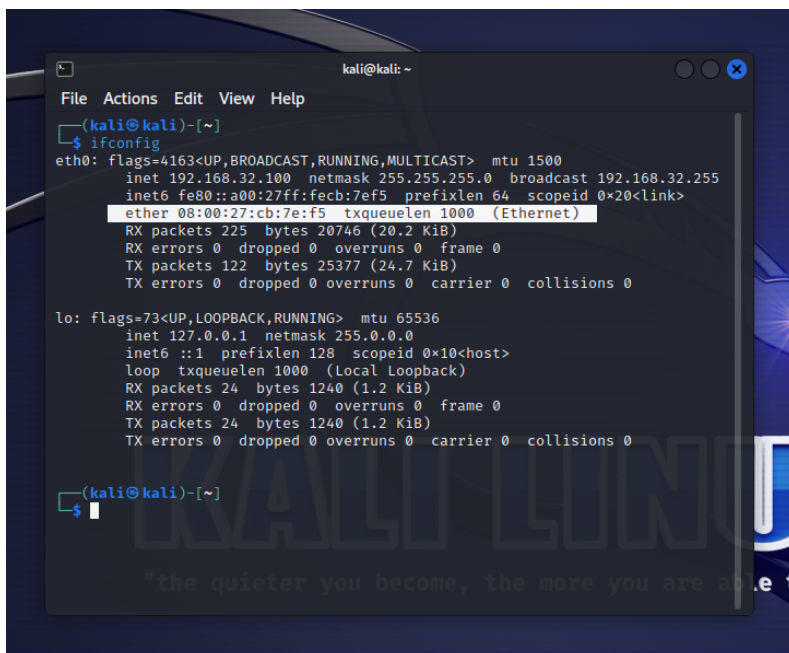
The screenshot shows the Wireshark interface with the same packet capture. The packet list on the left shows 15 packets. Packet 9 is selected, which is an HTTP GET request. The packet details pane on the right shows the structure of the packet, including Ethernet II, Internet Protocol Version 4, and Hypertext Transfer Protocol. The packet bytes pane at the bottom shows the raw data of the selected packet.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu_74:76:e9	Broadcast	ARP	60	Who has 192.168.32.100? Tell 192.168.32.101
2	0.000044266	PcsCompu_cb:7e:f5	PcsCompu_74:76:e9	ARP	42	192.168.32.100 is at 08:00:27:cb:7e:f5
3	0.011755547	192.168.32.101	192.168.32.100	TCP	60	49188 -> 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
4	0.011825505	192.168.32.100	192.168.32.101	TCP	60	80 -> 49188 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
5	0.012307201	192.168.32.101	192.168.32.100	TCP	60	49188 -> 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
6	0.014287767	192.168.32.101	192.168.32.100	HTTP	449	GET / HTTP/1.1
7	0.014305279	192.168.32.100	192.168.32.101	TCP	54	80 -> 49188 [ACK] Seq=1 Ack=396 Win=64128 Len=0
8	0.023125759	192.168.32.100	192.168.32.101	TCP	204	80 -> 49188 [PSH, ACK] Seq=1 Ack=396 Win=64128 Len=150 [TCP segment of a reassembled PDU]
9	0.024799728	192.168.32.100	192.168.32.101	HTTP	312	HTTP/1.1 200 OK (text/html)
10	0.025256706	192.168.32.101	192.168.32.100	TCP	60	49188 -> 80 [ACK] Seq=396 Ack=410 Win=65292 Len=0
11	0.025256936	192.168.32.101	192.168.32.100	TCP	60	49188 -> 80 [FIN, ACK] Seq=396 Ack=410 Win=65292 Len=0
12	0.025284363	192.168.32.100	192.168.32.101	TCP	54	80 -> 49188 [ACK] Seq=410 Ack=397 Win=64128 Len=0
13	5.031929043	PcsCompu_cb:7e:f5	PcsCompu_74:76:e9	ARP	42	Who has 192.168.32.101? Tell 192.168.32.100
14	5.032534327	PcsCompu_74:76:e9	PcsCompu_cb:7e:f5	ARP	60	192.168.32.101 is at 08:00:27:74:76:e9
15	36.519714651	fe80::a00:27ff:feeb::ff02::2	ff02::2	ICMPv6	70	Router Solicitation from 08:00:27:cb:7e:f5



Verifico anche l'indirizzo MAC di sorgente e destinazione (Destination e Source) in Ethernet II e lo confronto con quello di Kali con Ifconfig.

```
▶ Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 b
▼ Ethernet II, Src: PcsCompu_74:76:e9 (08:00:27:74:76:e9), Dst:
  ▶ Destination: PcsCompu_cb:7e:f5 (08:00:27:cb:7e:f5)
  ▶ Source: PcsCompu_74:76:e9 (08:00:27:74:76:e9)
  Type: IPv4 (0x0800)
▶ Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.
▶ Transmission Control Protocol, Src Port: 49159, Dst Port: 80,
```



```
kali@kali: ~
File Actions Edit View Help
(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 0<link>
    ether 08:00:27:cb:7e:f5 txqueuelen 1000 (Ethernet)
    RX packets 225 bytes 20746 (20.2 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 122 bytes 25377 (24.7 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 0<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 24 bytes 1240 (1.2 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 24 bytes 1240 (1.2 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

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

Passo ora al filtraggio dell'https: Posso notare l'invio di pacchetti TLSv1 e ciò indica che client e server si scambiano dati in modo cifrato. Posso vedere richiesta Client e risposta Server: "Client Hello", "Server Hello....server key exchange" Ovviamente non vedo nessuna pagina HTML in chiaro ma solo dati cifrati.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.32.101	192.168.32.100	TCP	66	49187 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
2	0.000061126	192.168.32.100	192.168.32.101	TCP	66	443 → 49187 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
3	0.000060822	192.168.32.101	192.168.32.100	TCP	60	49187 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0
4	0.001194721	192.168.32.101	192.168.32.100	TLSv1	133	Client Hello
5	0.001283974	192.168.32.100	192.168.32.101	TCP	54	443 → 49187 [ACK] Seq=1 Ack=137 Win=64128 Len=0
6	0.026610817	192.168.32.100	192.168.32.101	TLSv1	1373	Server Hello, Certificate, Server Key Exchange, Server Hello Done
7	0.031972672	192.168.32.101	192.168.32.100	TLSv1	188	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
8	0.032718343	192.168.32.100	192.168.32.101	TLSv1	113	Change Cipher Spec, Encrypted Handshake Message
9	0.035159344	192.168.32.101	192.168.32.100	TLSv1	379	Application Data
10	0.043105455	192.168.32.100	192.168.32.101	TLSv1	235	Application Data
11	0.044784356	192.168.32.100	192.168.32.101	TLSv1	384	Application Data, Encrypted Alert
12	0.045288286	192.168.32.101	192.168.32.100	TCP	60	49187 → 443 [ACK] Seq=596 Ack=1891 Win=65700 Len=0
13	0.045288517	192.168.32.101	192.168.32.100	TCP	60	49187 → 443 [FIN, ACK] Seq=596 Ack=1891 Win=65700 Len=0
14	0.045509016	192.168.32.100	192.168.32.101	TCP	54	443 → 49187 [ACK] Seq=1891 Ack=597 Win=64128 Len=0
15	21.789562350	fe80::a00:27ff:feb...	ff02::2	ICMPv6	70	Router Solicitation from 08:00:27:cb:7e:f5

```

Frame 4: 198 bytes on wire (1584 bits) captured (1584 bits) on 0
Ethernet II, Src: PcsCompu_08:00:27:74:76:e9, Dst: PcsCompu_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: 49187, Dst Port: 443, Seq: 1, Win: 65700, Len: 0
Transport Layer Security
0000 08 00 27 cb 7e f5 08 00 27 74 76 e9 08 00 45 00 .....tv...E...
0010 00 b0 01 20 40 00 00 00 37 03 c0 a8 20 65 c0 a8 ...+@...7...e...
0020 20 64 c0 23 01 bb 3b ba af 20 67 fa 33 fc 50 18 ...d#3...g3P...
0030 01 f5 c7 5b 00 00 16 03 01 00 59 02 00 00 55 03 ...[...Y...U...
0040 01 0f 8c 2b 7f b8 37 87 e8 e6 56 1b 44 3c 1b 6a ...+...7...V D<...j
0050 4b 0f e8 75 ff 56 b4 4f cf 44 4f 57 4e 47 52 44 ...K...u.V.O...DOWNGRD
0060 00 20 a8 dc 7c b4 1c 0e d8 11 fb dc 72 78 48 a0 ...|...|...rxH...
0070 93 67 fa 30 7d 7a 81 4d 77 b5 12 bf 5b 21 23 a9 ...g0}z-M w...[!#...
0080 43 29 c0 14 00 00 0d ff 01 00 01 00 00 0b 00 04 ...C).....k...g...d
0090 03 00 01 02 16 03 01 03 0b 0b 00 03 67 00 03 64 ...a0...]0...E...
00a0 00 03 61 30 82 03 5d 30 82 02 45 a0 03 02 01 02

```

```

Frame 10: 1373 bytes on wire (10984 bits) captured (10984 bits) on 0
Ethernet II, Src: PcsCompu_08:00:27:74:76:e9, Dst: PcsCompu_08:00:27:cb:7e:f5
Internet Protocol Version 4, Src: 192.168.32.100, Dst: 192.168.32.101
Transmission Control Protocol, Src Port: 443, Dst Port: 49187, Seq: 596, Win: 65700, Len: 0
Transport Layer Security
0000 08 00 27 74 76 e9 08 00 27 cb 7e f5 08 00 45 00 .....tv...E...
0010 05 4f 7a 17 40 00 40 06 f9 77 c0 a8 20 64 c0 a8 ...Oz@@...w...d...
0020 20 65 01 bb c0 1c d4 81 a0 84 0f ff 6e be 50 18 ...e...n.P...
0030 01 f5 c7 5b 00 00 16 03 01 00 59 02 00 00 55 03 ...[...Y...U...
0040 01 0f 8c 2b 7f b8 37 87 e8 e6 56 1b 44 3c 1b 6a ...+...7...V D<...j
0050 4b 0f e8 75 ff 56 b4 4f cf 44 4f 57 4e 47 52 44 ...K...u.V.O...DOWNGRD
0060 00 20 a8 dc 7c b4 1c 0e d8 11 fb dc 72 78 48 a0 ...|...|...rxH...
0070 93 67 fa 30 7d 7a 81 4d 77 b5 12 bf 5b 21 23 a9 ...g0}z-M w...[!#...
0080 43 29 c0 14 00 00 0d ff 01 00 01 00 00 0b 00 04 ...C).....k...g...d
0090 03 00 01 02 16 03 01 03 0b 0b 00 03 67 00 03 64 ...a0...]0...E...
00a0 00 03 61 30 82 03 5d 30 82 02 45 a0 03 02 01 02

```

Verifico nuovamente gli indirizzi MAC per https:

▶	Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on 0
▶	Ethernet II, Src: PcsCompu_74:76:e9 (08:00:27:74:76:e9), Dst: PcsCompu_cb:7e:f5 (08:00:27:cb:7e:f5)
▶	Destination: PcsCompu_cb:7e:f5 (08:00:27:cb:7e:f5)
▶	Source: PcsCompu_74:76:e9 (08:00:27:74:76:e9)
▶	Type: IPv4 (0x0800)
▶	Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100
▶	Transmission Control Protocol, Src Port: 49167, Dst Port: 443, Seq: 1, Win: 65700, Len: 0

## In Conclusione:

La differenza sostanziale tra HTTPS e HTTP è la cifratura dei contenuti infatti, con http posso vedere il contenuto html della pagina mentre con https non posso farlo. Posso inoltre riconoscere http dalle richieste inviate come GET e https invece, dallo scambio TLS.