



Nell'esercizio di oggi metteremo insieme le competenze acquisite finora. Lo studente verrà valutato sulla base della risoluzione al problema seguente.

Requisiti e servizi:

- Kali Linux ↗ IP 192.168.32.100
- Windows 7 ↗ IP 192.168.32.101
- HTTPS server: attivo
- Servizio DNS per risoluzione nomi di dominio: attivo

Traccia:

Simulare, in ambiente di laboratorio virtuale, un'architettura client server in cui un client con indirizzo 192.168.32.101 (Windows 7) richiede tramite web browser una risorsa all'hostname epicode.internal che risponde all'indirizzo 192.168.32.100 (Kali).

Si intercetti poi la comunicazione con Wireshark, evidenziando i MAC address di sorgente e destinazione ed il contenuto della richiesta HTTPS.

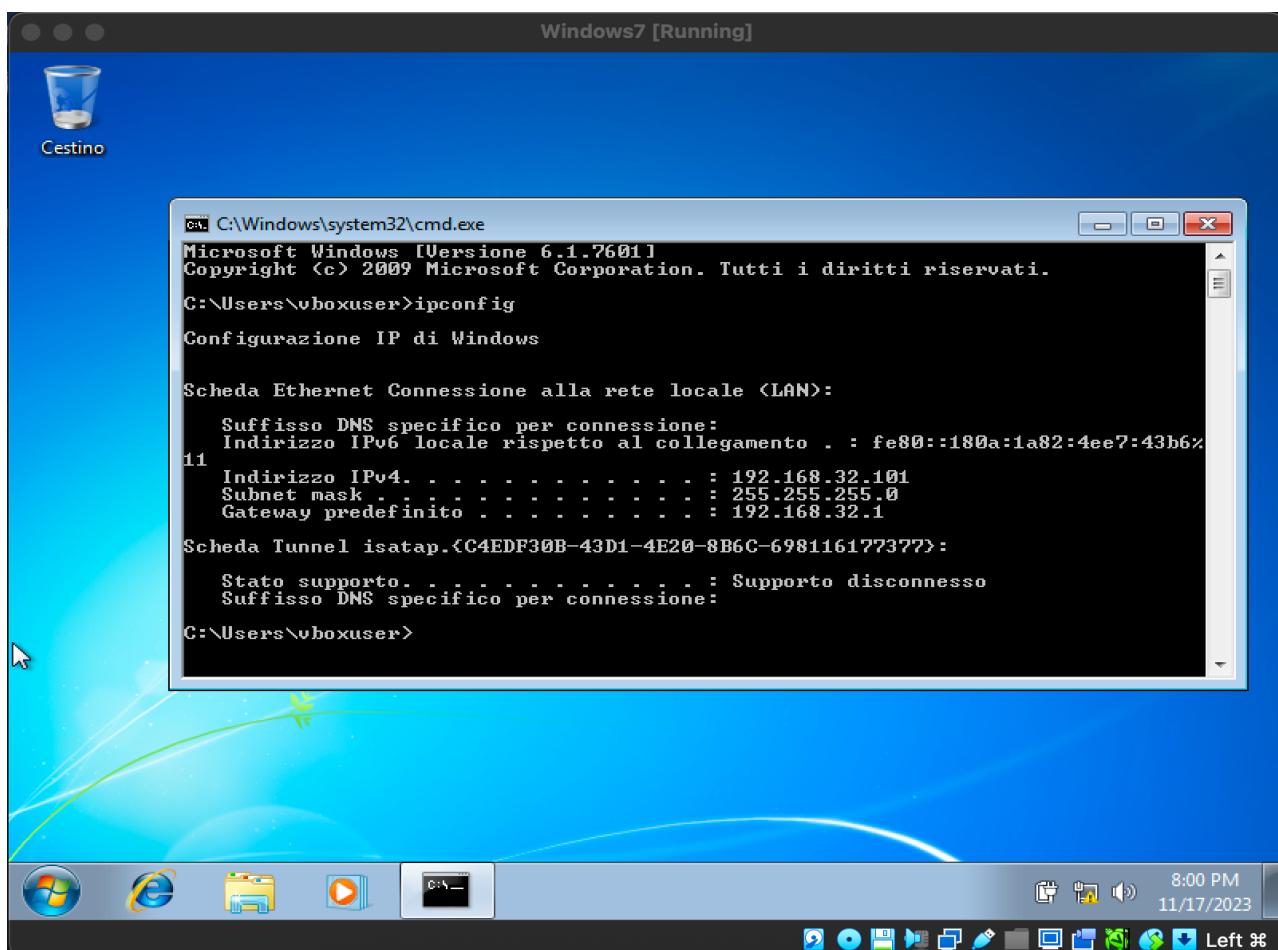
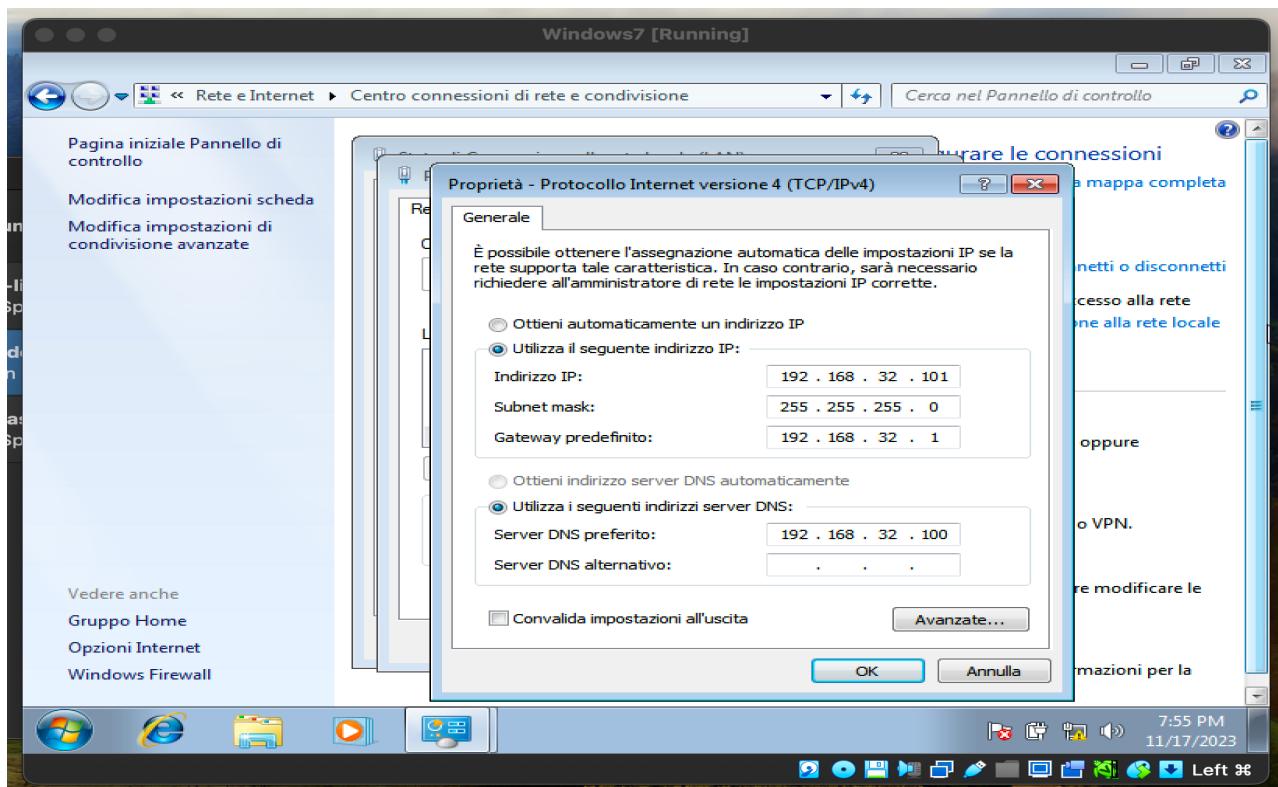
Ripetere l'esercizio, sostituendo il server HTTPS, con un server HTTP. Si intercetti nuovamente il traffico, evidenziando le eventuali differenze tra il traffico appena catturato in HTTP ed il traffico precedente in HTTPS. Spiegare, motivandole, le principali differenze se presenti.

SVOLGIMENTO:

Passo 1 → Impostare sulla macchina Windows7 l'indirizzo IP 192.168.32.101

Per eseguire questo primo passo utilizziamo i tool ad interfaccia grafica messi a disposizione dal sistema operativo Windows. Nelle configurazioni delle reti LAN protocollo IPv4 impostiamo ip, subnet, gateway (anche se non dobbiamo uscire sulla rete esterna) e già che ci siamo anche il puntamento al DNS che sarà attivo sulla macchina Kali che ospiterà i server attraverso il servizio di Inetsim.

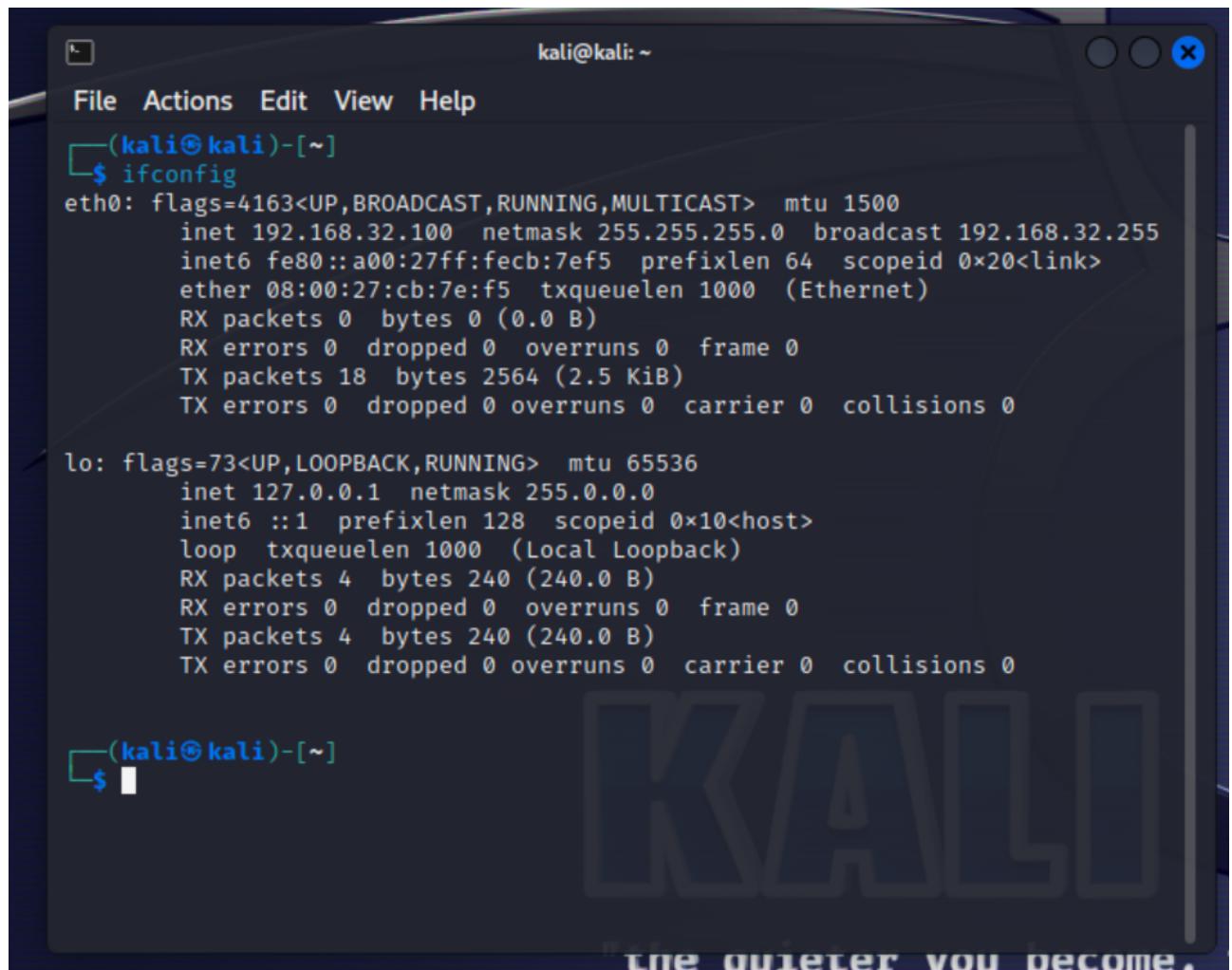
Si allegano screen shot



Passo 2 → Impostare sulla macchina Kali l'indirizzo IP 192.168.32.100

In questa seconda macchina andiamo invece a modificare l'indirizzo IP modificando il file interfaces nella directory network con l'editor di testo che si apre con il comando nano ed i privilegi di super user (sudo nano /etc/network/interfaces)

Si allega screen della modifica IPv4 sulla macchina Kali



```
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:fecc:7ef5 prefixlen 64 scopeid 0x20<link>
            ether 08:00:27:cb:7e:f5 txqueuelen 1000 (Ethernet)
              RX packets 0 bytes 0 (0.0 B)
              RX errors 0 dropped 0 overruns 0 frame 0
              TX packets 18 bytes 2564 (2.5 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

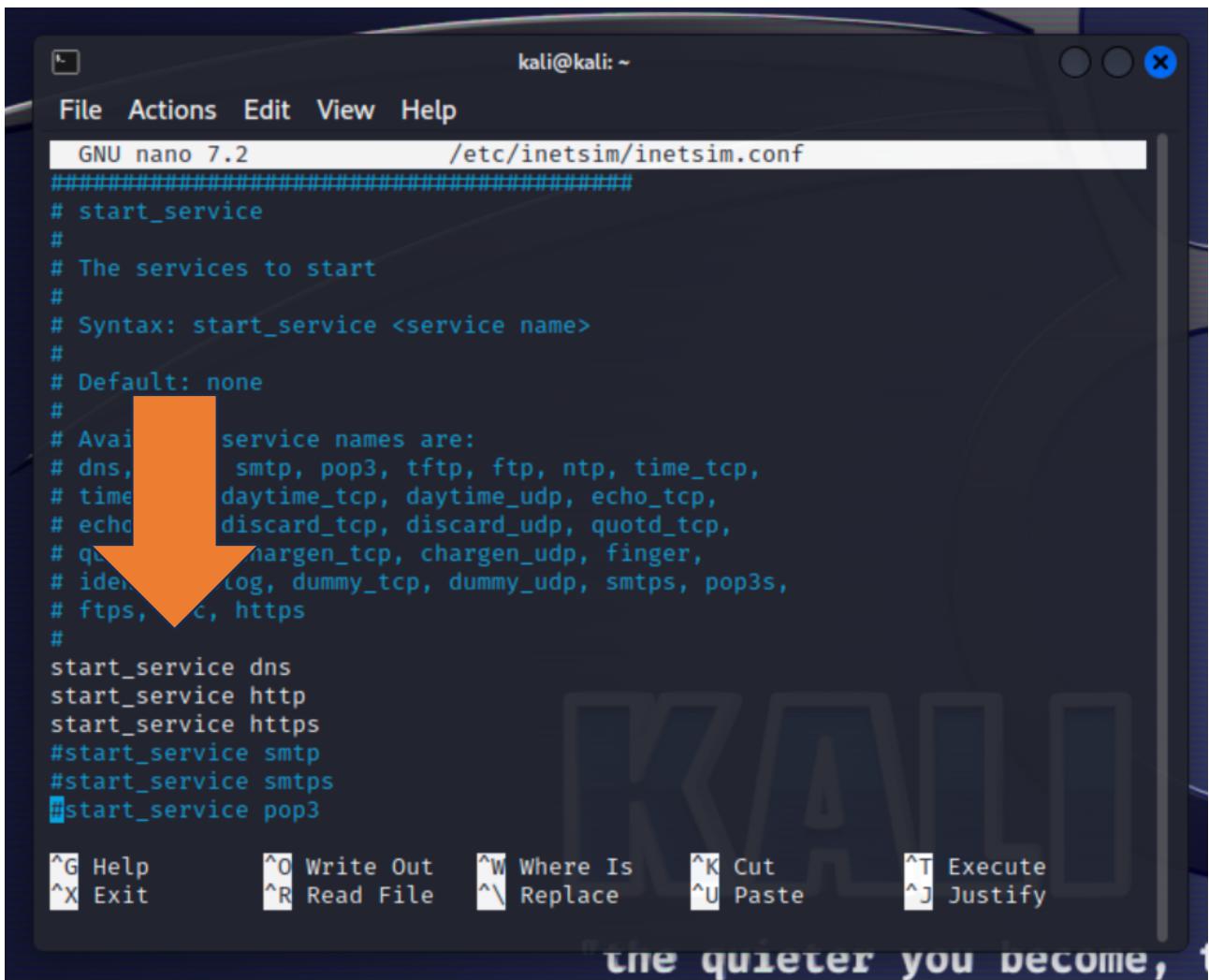
└─(kali㉿kali)-[~]
$
```

Passaggio 3 → Configurazione dei Servizi

In questo passaggio andiamo a configurare i servizi HTTP e HTTPS e DNS sulla macchina che fungerà da fake web server attraverso Inetsim che gira sulla ns macchina kali

Per aprire la configurazione di Inetsim usiamo il comando (sudo nano /etc/inetsim/inetsim.conf)
Per startare il servizio utilizziamo invece sudo inetsim

Di seguito gli screen della configurazione di Inetsim



kali@kali: ~

File Actions Edit View Help

GNU nano 7.2 /etc/inetsim/inetsim.conf

```
#####
# start_service
#
# The services to start
#
# Syntax: start_service <service name>
#
# Default: none
#
# Available service names are:
# dns, smtp, pop3, tftp, ftp, ntp, time_tcp,
# time, daytime_tcp, daytime_udp, echo_tcp,
# echo, discard_tcp, discard_udp, quotd_tcp,
# quota, chargen_tcp, chargen_udp, finger,
# ident, log, dummy_tcp, dummy_udp, smtps, pop3s,
# ftps, https
#
start_service dns
start_service http
start_service https
#start_service smtp
#start_service smtps
#start_service pop3
```

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

The screenshot shows a terminal window titled "GNU nano 7.2" with the file path "/etc/inetsim/inetsim.conf". The file contains configuration for various network services. A yellow box highlights a note about the "service_bind_address" setting:

Il bind_adress non viene settato a 0.0.0.0 come richiesto ma a 192.168.32.100 che permette lo svolgimento dell'esercizio e al tempo stesso rende più sicura la configurazione (anche se tutte le macchine sulla rete possono raggiungerlo posto che abbiamo 192.168.32.100 come DNS) ma li si potrebbe intervenire con delle policy per permettere solo alla macchina Windows7 di contattare la macchina Kali.

La configurazione è stata presa da un esercizio molto simile al ns dell'Università del Maryland.

```
# 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 service
#
# Syntax: service_bind_address
#
# Default: 127.0.0.1
#
service_bind_address 192.168.32.100

#####

# service_run_as_user

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

Punto 4 → Associare il nome di dominio episode.internal all'indirizzo IP della macchina Kali in modo che sia risolvibile con una chiamata get con protocollo http o https direttamente dal web browser della macchina Windows7

Si allegano Screen

kali@kali: ~

File Actions Edit View Help

GNU nano 7.2 /etc/inetsim/inetsim.conf

```
# 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 episode.internal 192.168.32.100

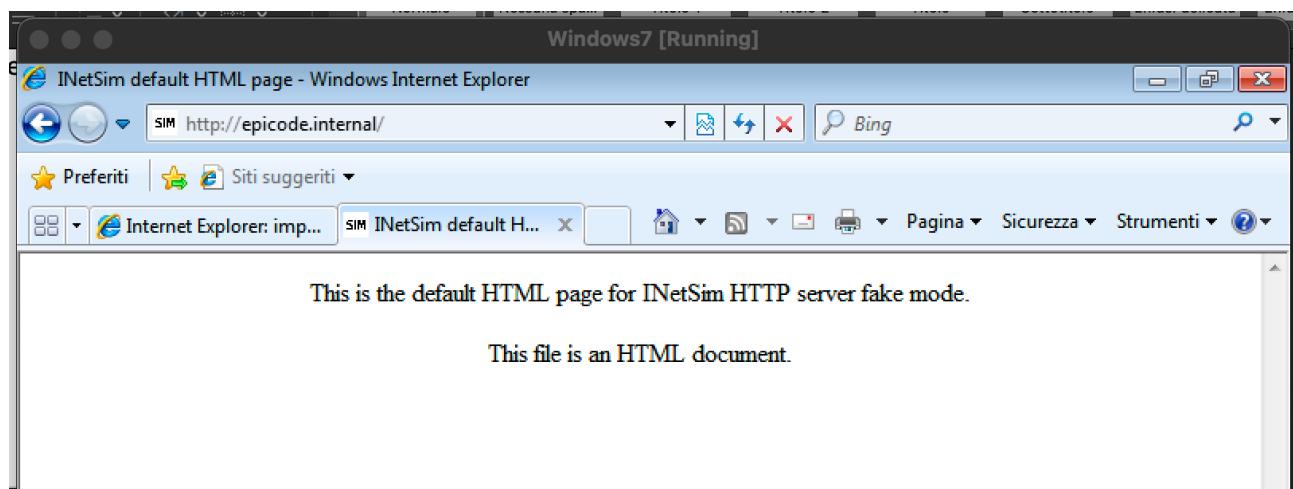
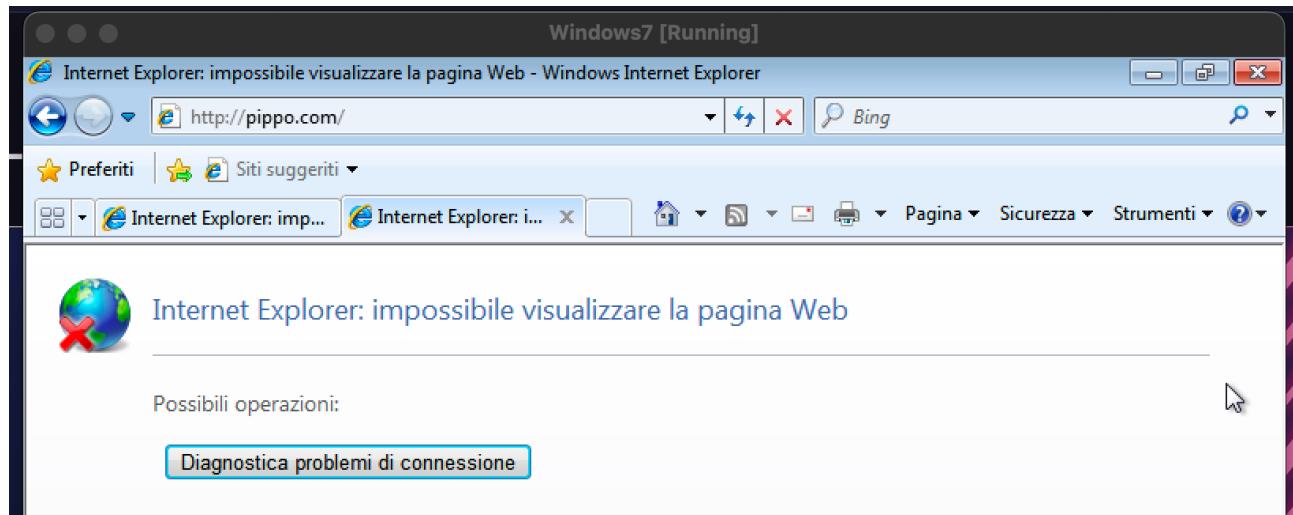
#####
# dns_version
#
# DNS version
```

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

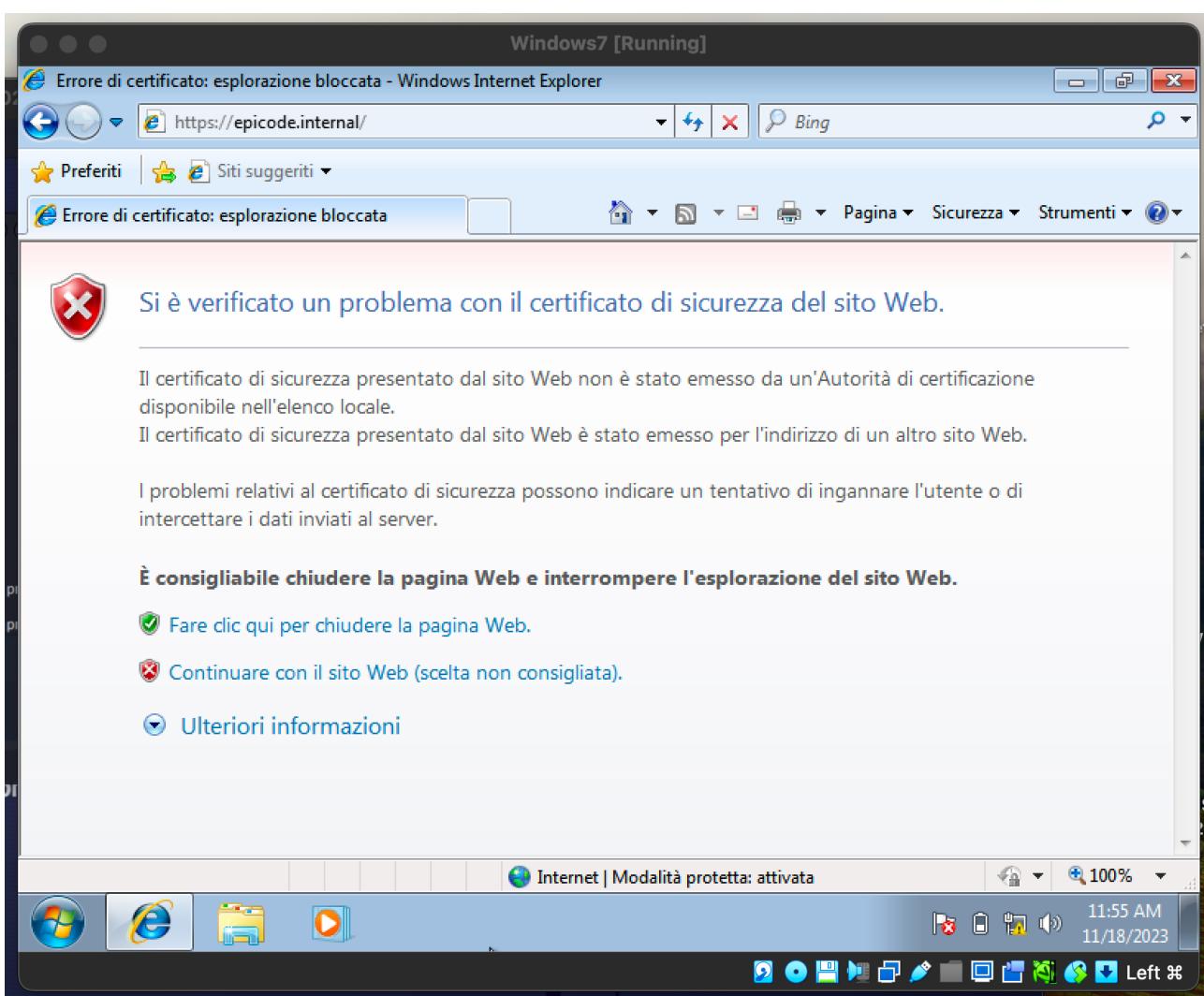
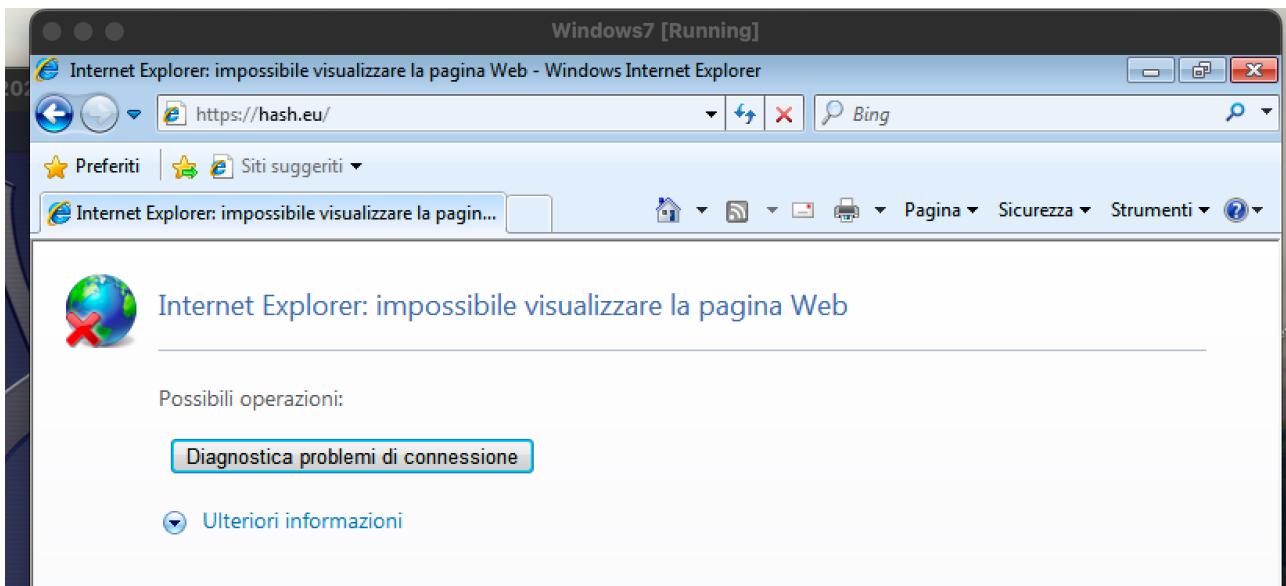
Passo 5 → Ora che tutto è pronto andremo ad operare in questa maniera:

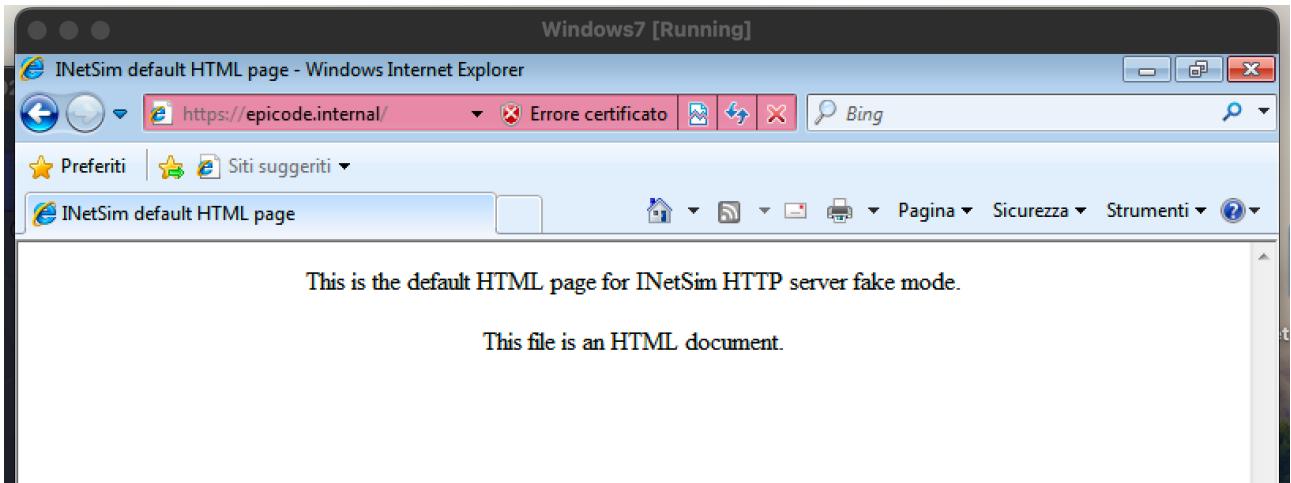
Verifichiamo prima il raggiungimento e la risoluzione del nome `epicode.internal` dal browser della macchina Windows7 sia con protocollo http sia con protocollo https, poi andremo ad intercettare il traffico prima durante l'utilizzo di un protocollo e poi dell'altro.

Dimostrazione grafica della comunicazione della macchina Windows7 che può chiamare <http://epicode.internal> dal browser e raggiunge il fake web server inetsim sulla macchina kali

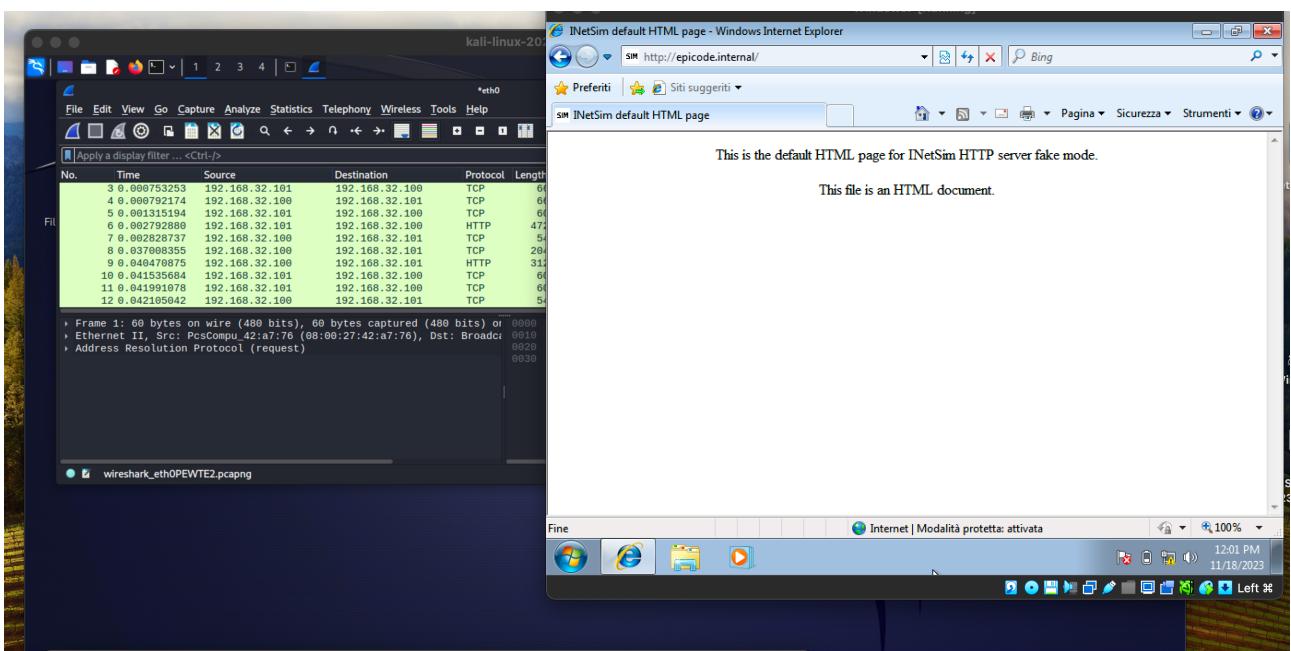


Dimostrazione grafica della comunicazione della macchina Windows7 che può chiamare <https://epicode.internal> dal browser e raggiunge il fake web server inetsim sulla macchina kali





Passo 6 → Avviando Wireshark sulla macchina Kali partiamo con l'intercettazione del traffico http



Handshake

```

3 0.000753253 192.168.32.101      192.168.32.100      TCP      66 49164 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
4 0.000792174 192.168.32.100      192.168.32.101      TCP      66 80 → 49164 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
5 0.001315194 192.168.32.101      192.168.32.100      TCP      60 49164 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
  
```

MAC address destinazione

Protocol	Length	Info
ARP	60	Who has 192.168.32.100? Tell 192.168.32.101
ARP	42	192.168.32.100 is at 08:00:27:cb:7e:f5

Chiamata GET che contiene entrambi i MAC e IP

5	0.001315194	192.168.32.101	192.168.32.100	TCP	60	49164 → 80 [ACK]
6	0.002792880	192.168.32.101	192.168.32.100	HTTP	472	GET / HTTP/1.1
7	0.002828737	192.168.32.100	192.168.32.101	TCP	54	80 → 49164 [ACK]
8	0.037008355	192.168.32.100	192.168.32.101	TCP	204	80 → 49164 [PSH]
9	0.040470875	192.168.32.100	192.168.32.101	HTTP	312	HTTP/1.1 200 OK
10	0.041535684	192.168.32.101	192.168.32.100	TCP	60	49164 → 80 [ACK]
11	0.041991078	192.168.32.101	192.168.32.100	TCP	60	49164 → 80 [FIN]
12	0.042105042	192.168.32.100	192.168.32.101	TCP	54	80 → 49164 [ACK]

Frame 6: 472 bytes on wire (3776 bits), 472 bytes captured (3776 bits) on interface eth0
 Ethernet II, Src: PcsCompu_42:a7:76 (08:00:27:42:a7:76), Dst: PcsCompu_cb:7e:f5 (08:00:27:cb:7e:f5)
 ↳ Destination: PcsCompu_cb:7e:f5 (08:00:27:cb:7e:f5)
 ↳ Source: PcsCompu_42:a7:76 (08:00:27:42:a7:76)
 Type: IPv4 (0x0800)
 Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100
 Transmission Control Protocol, Src Port: 49164, Dst Port: 80, Seq: 1, Ack: 1, Len: 418
 Hypertext Transfer Protocol

Contenuto in chiaro del pacchetto http

kali-linux-2023.3-virtualbox-amd64 [Running]						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	PcsCompu_42:a7:76	Broadcast	ARP	60	Who has 192.168.32.100? Tell 192.168.32.101
2	0.0000302200	PcsCompu_cb:7e:f5	PcsCompu_42:a7:76	ARP	42	192.168.32.100 is at 08:00:27:cb:7e:f5
3	0.0007532533	192.168.32.101	192.168.32.100	TCP	66	49164 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
4	0.0007921744	192.168.32.100	192.168.32.101	TCP	66	80 → 49164 [SYN, ACK] Seq=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
5	0.0013151944	192.168.32.101	192.168.32.100	TCP	66	49164 → 80 [ACK] Seq=1 Win=65700 Len=0
6	0.0027928800	192.168.32.101	192.168.32.100	HTTP	472	GET / HTTP/1.1
7	0.0028287373	192.168.32.100	192.168.32.101	TCP	54	80 → 49164 [ACK] Seq=1 Ack=419 Win=64128 Len=0
8	0.0370083555	192.168.32.100	192.168.32.101	TCP	204	80 → 49164 [PSH, ACK] Seq=1 Ack=419 Win=64128 Len=150 [TCP segment of a reassembled PDU]
9	0.0404708755	192.168.32.100	192.168.32.101	HTTP	312	HTTP/1.1 200 OK (text/html)
10	0.0415356844	192.168.32.101	192.168.32.100	TCP	60	49164 → 80 [ACK] Seq=419 Ack=410 Win=65292 Len=0
11	0.0419910784	192.168.32.101	192.168.32.100	TCP	60	49164 → 80 [FIN, ACK] Seq=419 Ack=410 Win=65292 Len=0
12	0.0421050424	192.168.32.100	192.168.32.101	TCP	54	80 → 49164 [ACK] Seq=410 Ack=420 Win=64128 Len=0

Content-Length: 258\r\nDate: Sat, 18 Nov 2023 11:00:34 GMT\r\nConnection: Close\r\nContent-Type: text/html\r\n\r\n[HTTP response 1/1]\r\n[Time since request: 0.037677995 seconds]\r\n[Request in frame: 0]\r\n[Request URI: http://epicode.internal/]\r\nFile data: 258 bytes\r\nLine-based text data: text/html (10 lines)\r\n<html>\r\n <head>\r\n <title>INetSim default HTML page</title>\r\n </head>\r\n <body>\r\n <p><align="center">This is the default HTML page for INetSim HTTP server fake inc

```
0090 6d 6c 0d 0a 0d 0a 3c 68 74 6d 6c 3e 0a 20 20 3c ml...<h tml>. <
0091 68 65 61 64 3e 0a 20 20 20 3c 74 6d 6c 65 head>. <title
0092 3e 49 4e 65 74 53 69 6d 20 64 65 66 61 75 6c 74 >INetSim default
0093 20 48 54 4d 4c 20 70 61 67 65 3c 2f 74 69 74 6c HTML pa ge</titl
0094 65 3e 0a 20 20 3c 2f 68 65 61 64 3e 0a 20 20 3c e>. </h ead>. <
0095 62 6f 64 79 3e 0a 20 20 20 20 3c 70 3e 3c 2f 70 body>. <p></p>
0096 0f 3e 0a 20 20 20 3c 70 20 61 6c 69 67 6e 3d 22 >.< <p align="center">This
0097 63 65 66 64 65 20 3c 56 68 69 73 29 69 67 6c >is the default HTML
0098 61 6e 65 64 65 66 67 75 6c 74 69 74 6c 65 64 6c page fo r INetSi
0099 20 70 61 67 65 20 66 67 72 69 49 46 65 74 53 69 m HTTP s erver fa
0100 6d 20 48 54 54 50 20 73 65 72 76 65 72 20 66 61 ke mode </p>.
0101 0b 65 20 6d 6f 64 65 26 3c 2f 70 3e 0a 20 20 20 <p align n="Cente
0102 20 3c 70 20 61 6c 69 67 6e 3d 22 63 65 66 74 65 r=">This file is
0103 72 22 3e 54 68 69 73 20 66 69 6c 65 20 69 73 20 an HTML document
0104 61 6e 20 48 54 4d 4c 20 64 67 63 75 6d 65 66 74 .</p>.
0105 20 3c 2f 70 3e 0a 20 20 3c 2f 62 6f 64 79 3e 0a </html>.
```

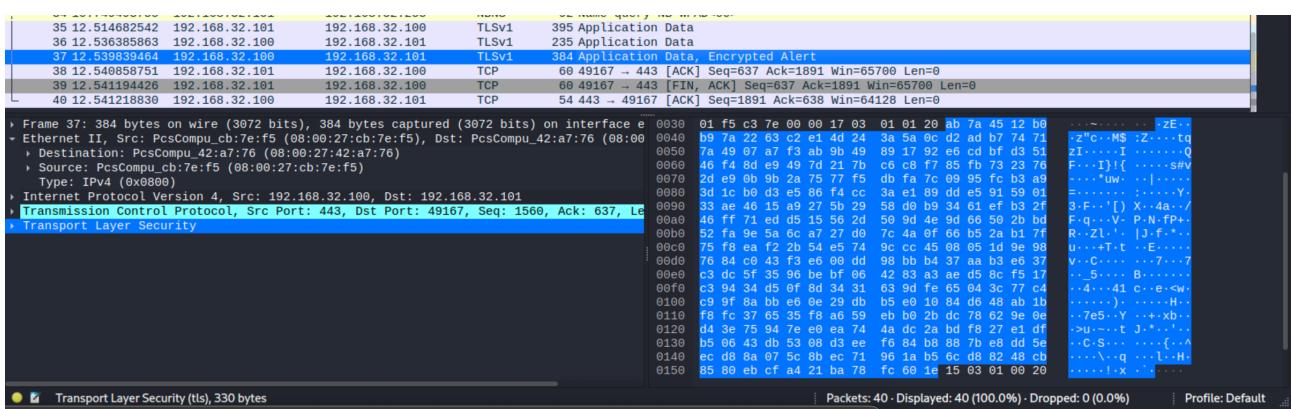
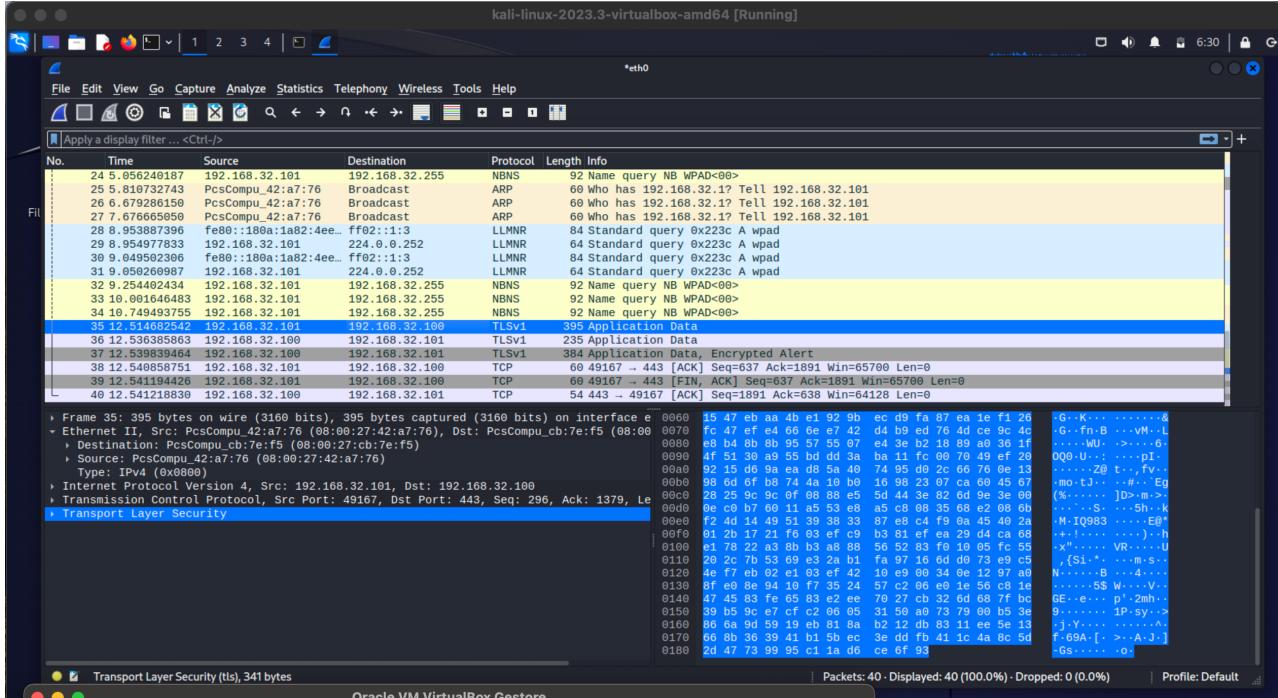
Frame (312 bytes) Reassembled TCP (408 bytes)

Passo 7 → Si ripete il passo 6 con il protocollo https

Handshake

3	0.001147272	192.168.32.101	192.168.32.100	TCP	66	49165 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
4	0.001173475	192.168.32.100	192.168.32.101	TCP	66	443 → 49165 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
5	0.005036530	192.168.32.101	192.168.32.100	TCP	60	49165 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0

Pacchetto HTTPS



DIFFERENZE riscontrate tra le intercettazioni dei pacchetti che viaggiano sui due diversi protocolli

In questa fase conclusiva andiamo a mettere per iscritto le differenze che già a livello di immagini si possono cogliere.

Dalla macchina Windows7 quando chiamiamo `epicode.internal` con il protocollo http ci viene restituita direttamente la pagina richiesta, quando invece facciamo la stessa cosa in https veniamo bloccati dall'assenza del certificato sul server. Solo dopo un'ulteriore conferma riusciamo ad ottenere la pagina, la cui barra degli indirizzi resta evidenziata in rosso.

Una seconda differenza sta proprio nella quantità di pacchetti scambiati che sono molti meno con il protocollo http.

Infine la differenza più importante e fondamentale è che la comunicazione con il protocollo http viaggiava in chiaro (siamo riusciti infatti a leggerne il contenuto) mentre i pacchetti https sono criptati (SSL/TSL) e risultano quindi indecifrabiili senza svolgere operazioni di reverse o senza conoscere la chiave per poterli decifrare. Abbiamo quindi una comunicazione nettamente più sicura. Anche le porte dei protocolli sono differenti: porta 80 per http e porta 443 per https.