Esercizio W4-D4 Garzoni Michele

Traccia esercizio:

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.

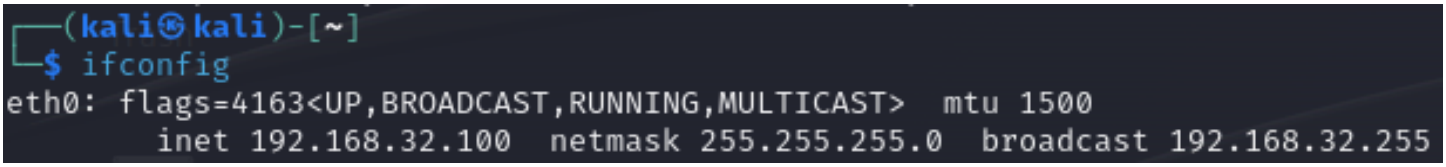
Ripetere l'esercizio, sostituendo il server HTTPS, con un server HTTP. Si intercetti nuovamente il traffico, evidenziando le eventuali differenze tra il traffico appena catturato in HTTP ed il traffico precedente in HTTPS. Spiegare, motivandole, le principali differenze se presenti.

Iniziamo con impostare i corretti settaggi IP nelle due VM come scritto nella traccia, quindi:

Kali

Digito sudo nano /etc/network/interfaces e imposto l'ip 192.168.32.100 alla macchina Kali e 192.168.32.1 come Gateway con subnet 255.255.255.0 (24)

Digito ifconfig sul terminale e controllo i settaggi appena impostati.



Windows

Da Windows entro nelle connessioni di rete, Protocollo internet versione 4 (TCP/IP) e inserisco i valori corretti:

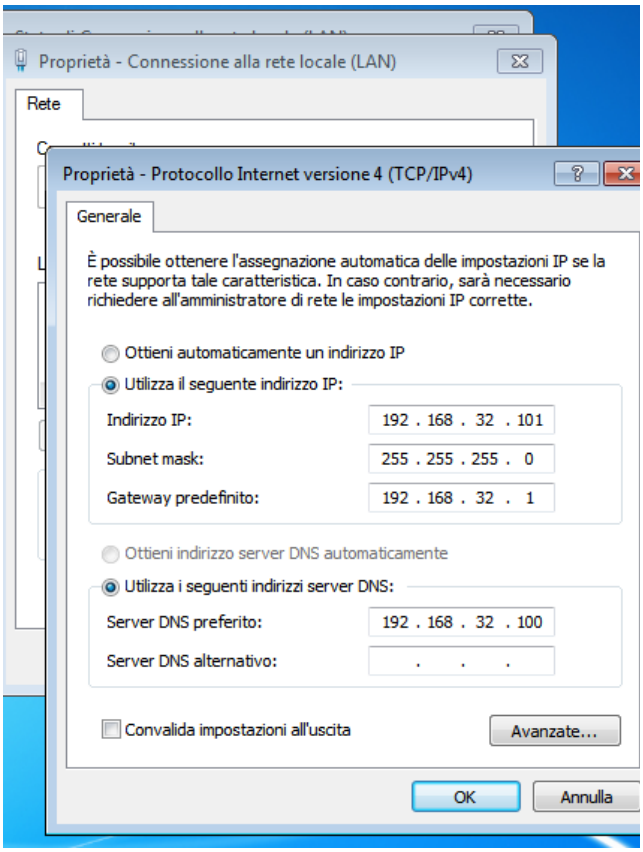
IP: 192.168.32.101

Subnet: 255.255.255.0

Gateway: 192.168.32.1

Impostiamo anche il server DNS sulla macchina Kali per dopo: *192.168.32.100*

Premo Ok per confermare:



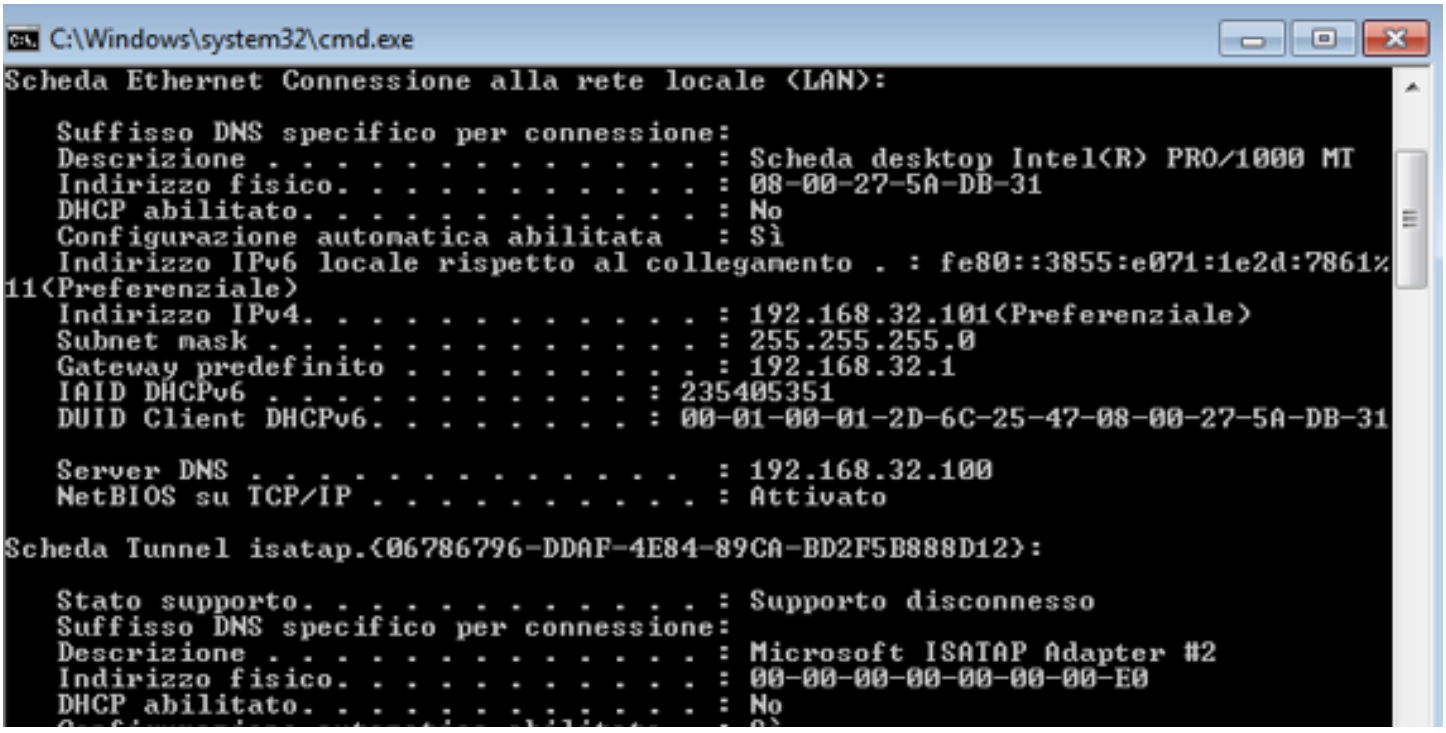
Faccio una prova di Ping, per confermare la corretta comunicazione in rete delle due VM:

```
C:\Users\vboxuser>ping 192.168.32.100

Esecuzione di Ping 192.168.32.100 con 32 byte di dati:
Risposta da 192.168.32.100: byte=32 durata<1ms TTL=64
Risposta da 192.168.32.100: byte=32 durata<1ms TTL=64
Risposta da 192.168.32.100: byte=32 durata<1ms TTL=64
Risposta da 192.168.32.100: byte=32 durata<1ms TTL=64

Statistiche Ping per 192.168.32.100:
    Pacchetti: Trasmessi = 4, Ricevuti = 4,
    Persi = 0 (0% persi),
Tempo approssimativo percorsi andata/ritorno in millisecondi:
    Minimo = 0ms, Massimo =  0ms, Medio =  0ms
```

```
┌──(kali㉿kali)-[~]
└─$ ping 192.168.32.101
PING 192.168.32.101 (192.168.32.101) 56(84) bytes of data.
64 bytes from 192.168.32.101: icmp_seq=1 ttl=128 time=0.154 ms
64 bytes from 192.168.32.101: icmp_seq=2 ttl=128 time=0.153 ms
64 bytes from 192.168.32.101: icmp_seq=3 ttl=128 time=0.166 ms
64 bytes from 192.168.32.101: icmp_seq=4 ttl=128 time=0.148 ms
64 bytes from 192.168.32.101: icmp_seq=5 ttl=128 time=0.155 ms
64 bytes from 192.168.32.101: icmp_seq=6 ttl=128 time=0.173 ms
^C
── 192.168.32.101 ping statistics ──
6 packets transmitted, 6 received, 0% packet loss, time 5113ms
rtt min/avg/max/mdev = 0.148/0.158/0.173/0.008 ms

┌──(kali㉿kali)-[~]
└─$
```

Allego screen anche delle relative config dove si può vedere anche i MAC delle due schede di rete:

Ipconfig /all in Windows (Indirizzo fisico è il Mac Address)



ifconfig di Kali Linux (Ether è il Mac Address)

```
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:fe21:b1d0  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:21:b1:d0  txqueuelen 1000  (Ethernet)
        RX packets 434  bytes 47754 (46.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 237  bytes 121324 (118.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Proseguiamo su Kali aprendo la configurazione di INetSim, che fungerà da simulatore di una rete Internet.

Su Kali Terminale digitiamo “Sudo Nano /etc/inetsim/inetsim.conf con password ”kali”

```
┌──(kali㉿kali)-[~]
└─$ sudo nano /etc/inetsim/inetsim.conf
[sudo] password for kali:
```

Siamo nella configurazione del programma INetSim, dove andremo a cambiare i parametri per riflettere i nostri IP di rete e servizi necessari:

Modifichiamo il “Service bind address” inserendo l’IP della nostra macchina Kali:

```
##########################################################
# service_bind_address
#
# IP address to bind services to
#
# Syntax: service_bind_address <IP address>
#
# Default: 127.0.0.1
#
service_bind_address 192.168.32.100
```

Modifichiamo i settaggi DNS per risolvere correttamente il testo “epicode.internal” se digitato dal web browser Windows

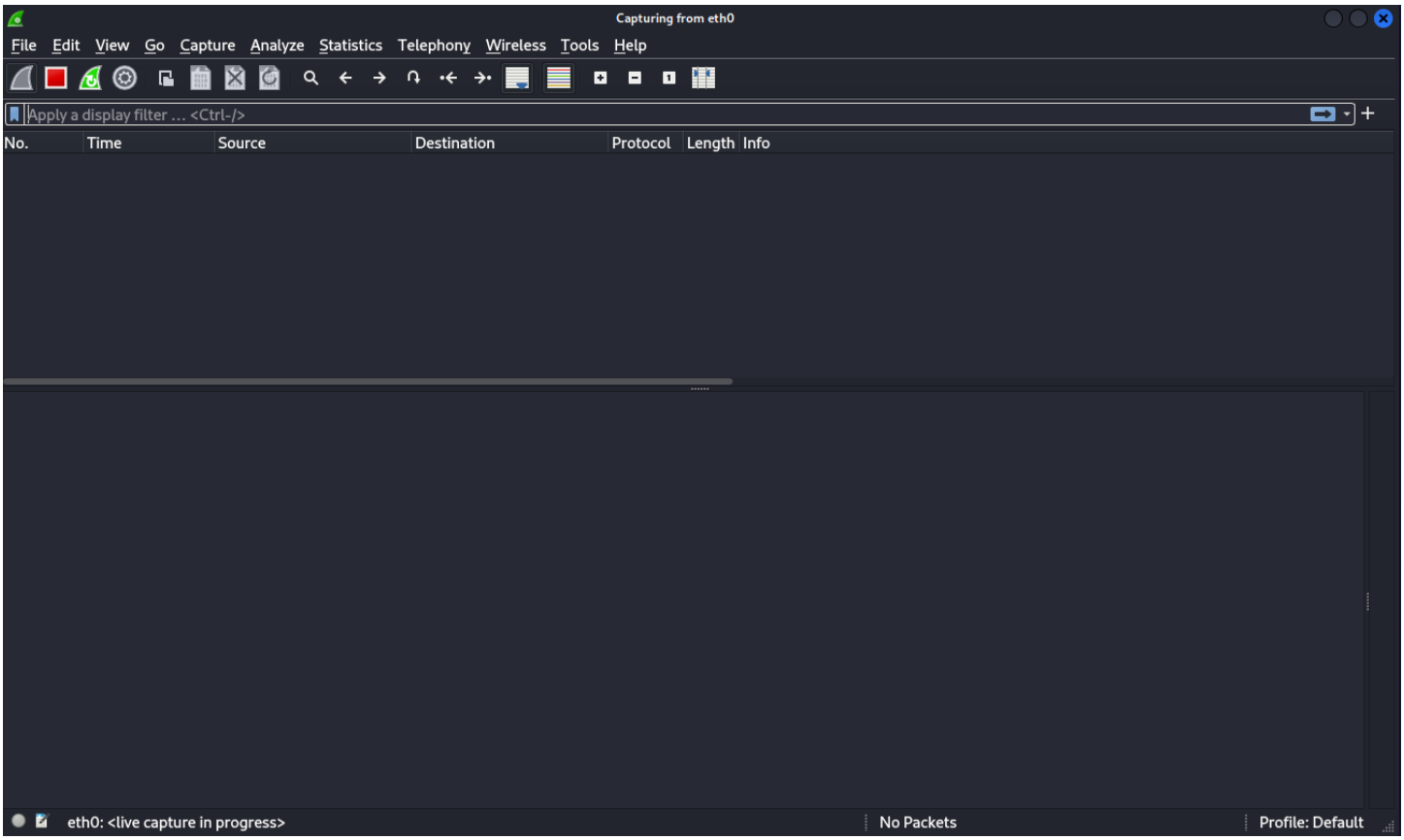
```
##################################################################
# dns_static
#
# Static mappings for DNS
#
# Syntax: dns_static <fqdn hostname> <IP address>
#
# Default: none
#
dns_static epicode.internal 192.168.32.100
#dns_static ns1.foo.com 10.70.50.30
#dns_static ftp.bar.net 10.10.20.30
```

Usciamo dal nano salvando le modifiche con le combinazioni di tasti “Ctrl + O” e “Ctrl + X”

Avvio Inetsim (Sudo INetSim) da terminale Kali

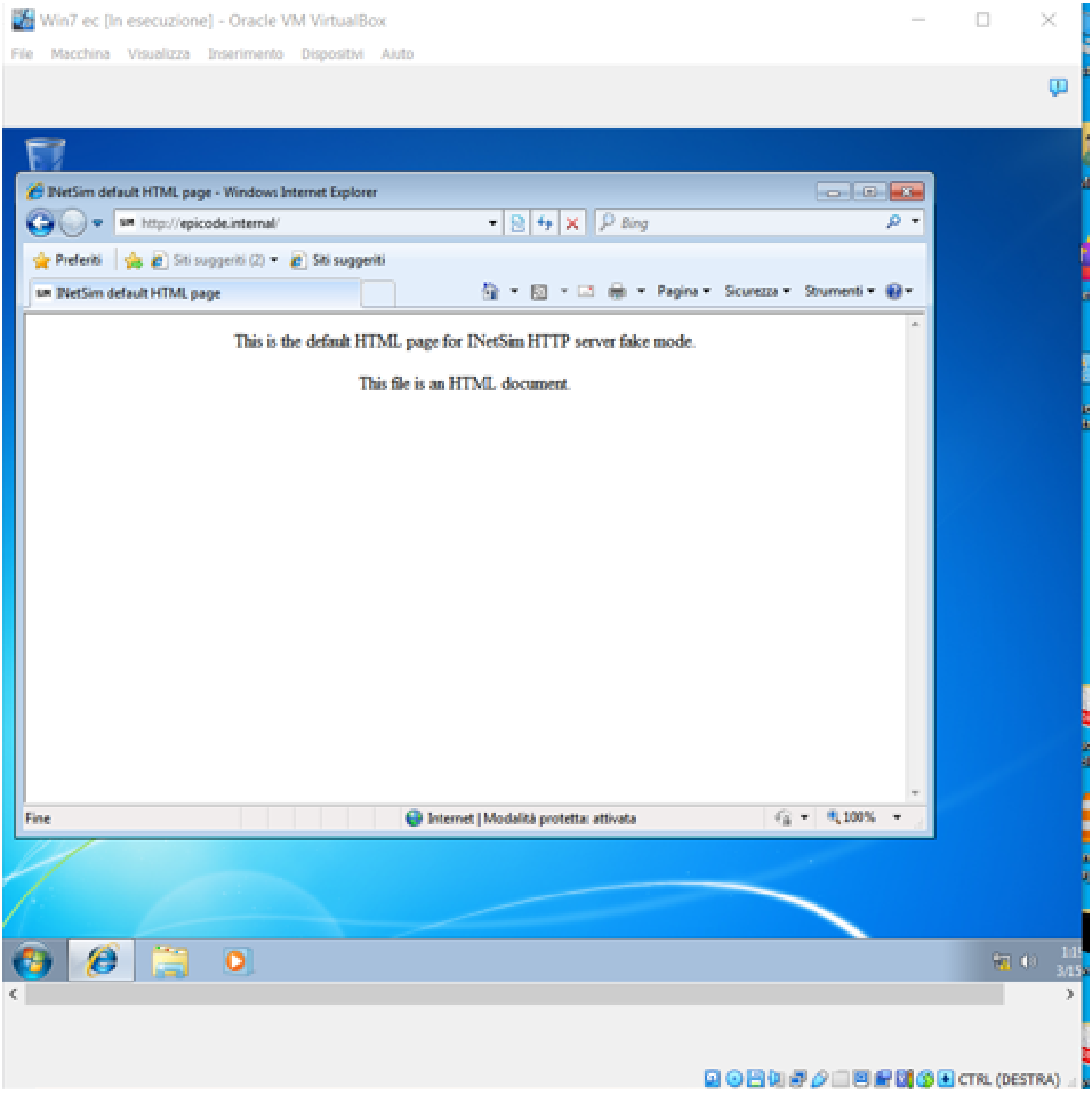
```
└─$ sudo inetsim
INetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
Using log directory:      /var/log/inetsim/
Using data directory:     /var/lib/inetsim/
Using report directory:   /var/log/inetsim/report/
Using configuration file: /etc/inetsim/inetsim.conf
Parsing configuration file.
Configuration file parsed successfully.
=== INetSim main process started (PID 10182) ===
Session ID:     10182
Listening on:   192.168.32.100
Real Date/Time: 2024-03-15 08:32:35
Fake Date/Time: 2024-03-15 08:32:35 (Delta: 0 seconds)
 Forking services...
  * dns_53_tcp_udp - started (PID 10192)
print() on closed filehandle MLOG at /usr/share/perl5/Net/DNS/Nameserver.pm l
print() on closed filehandle MLOG at /usr/share/perl5/Net/DNS/Nameserver.pm l
  * irc_6667_tcp - started (PID 10202)
  * smtps_465_tcp - started (PID 10196)
  * pop3s_995_tcp - started (PID 10198)
  * ntp_123_udp - started (PID 10203)
  * echo_7_udp - started (PID 10212)
  * finger_79_tcp - started (PID 10204)
  * echo_7_tcp - started (PID 10211)
  * ident_113_tcp - started (PID 10205)
  * daytime_13_tcp - started (PID 10209)
  * syslog_514_udp - started (PID 10206)
  * discard_9_udp - started (PID 10216)
  * discard_9_tcp - started (PID 10215)
  * daytime_13_udp - started (PID 10210)
  * time_37_tcp - started (PID 10207)
  * time_37_udp - started (PID 10208)
  * pop3_110_tcp - started (PID 10197)
  * http_80_tcp - started (PID 10193)
  * ftps_990_tcp - started (PID 10200)
  * tftp_69_udp - started (PID 10201)
  * quotd_17_tcp - started (PID 10217)
  * chargen_19_tcp - started (PID 10219)
  * dummy_1_tcp - started (PID 10221)
  * dummy_1_udp - started (PID 10222)
  * chargen_19_udp - started (PID 10220)
  * quotd_17_udp - started (PID 10218)
  * ftp_21_tcp - started (PID 10199)
  * smtp_25_tcp - started (PID 10195)
  * https_443_tcp - started (PID 10194)
 done.
Simulation running.
```

Con la simulazione attiva e funzionante, avvio Wireshark, seleziono “Eth0” e faccio partire lo sniffing dei pacchetti su Eth0.



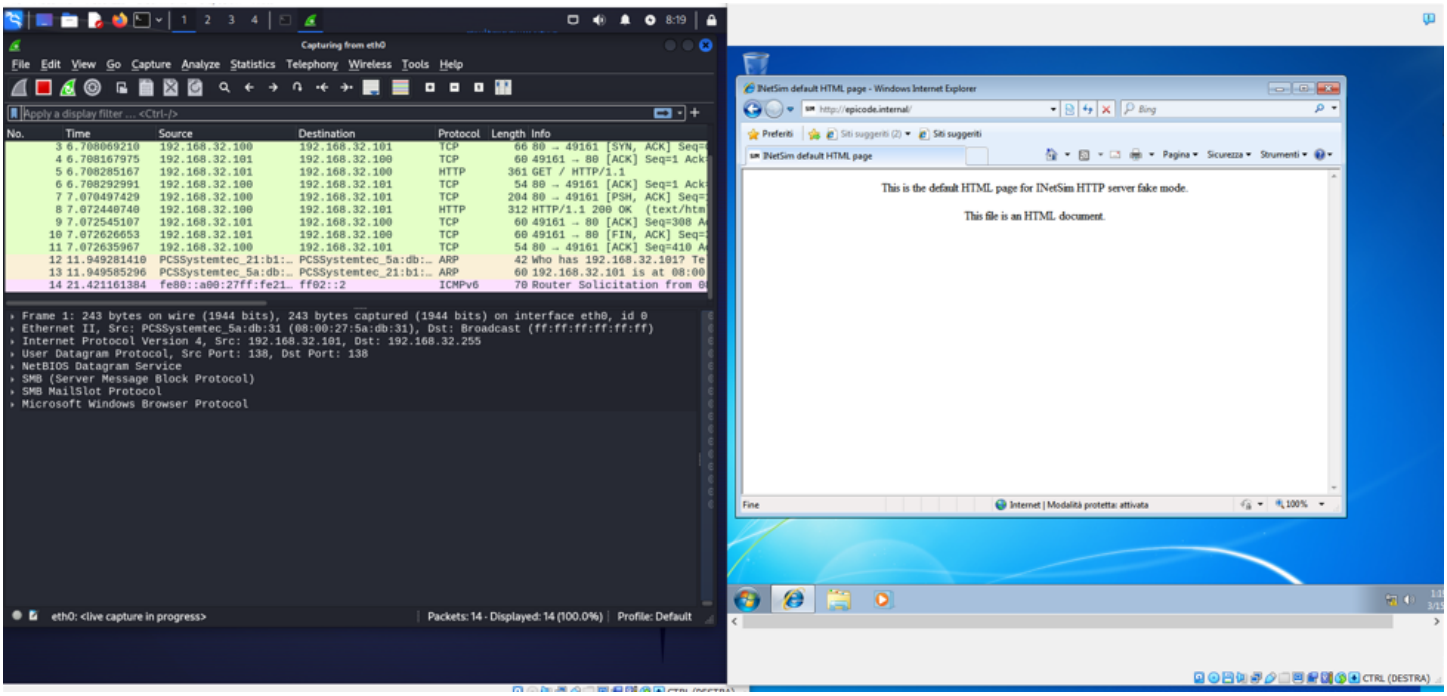
VERSIONE HTTP

Dalla macchina Windows apriamo il browser predefinito (Internet Explorer) e nella barra indirizzo digitiamo il nostro sito web simulato con INetSim

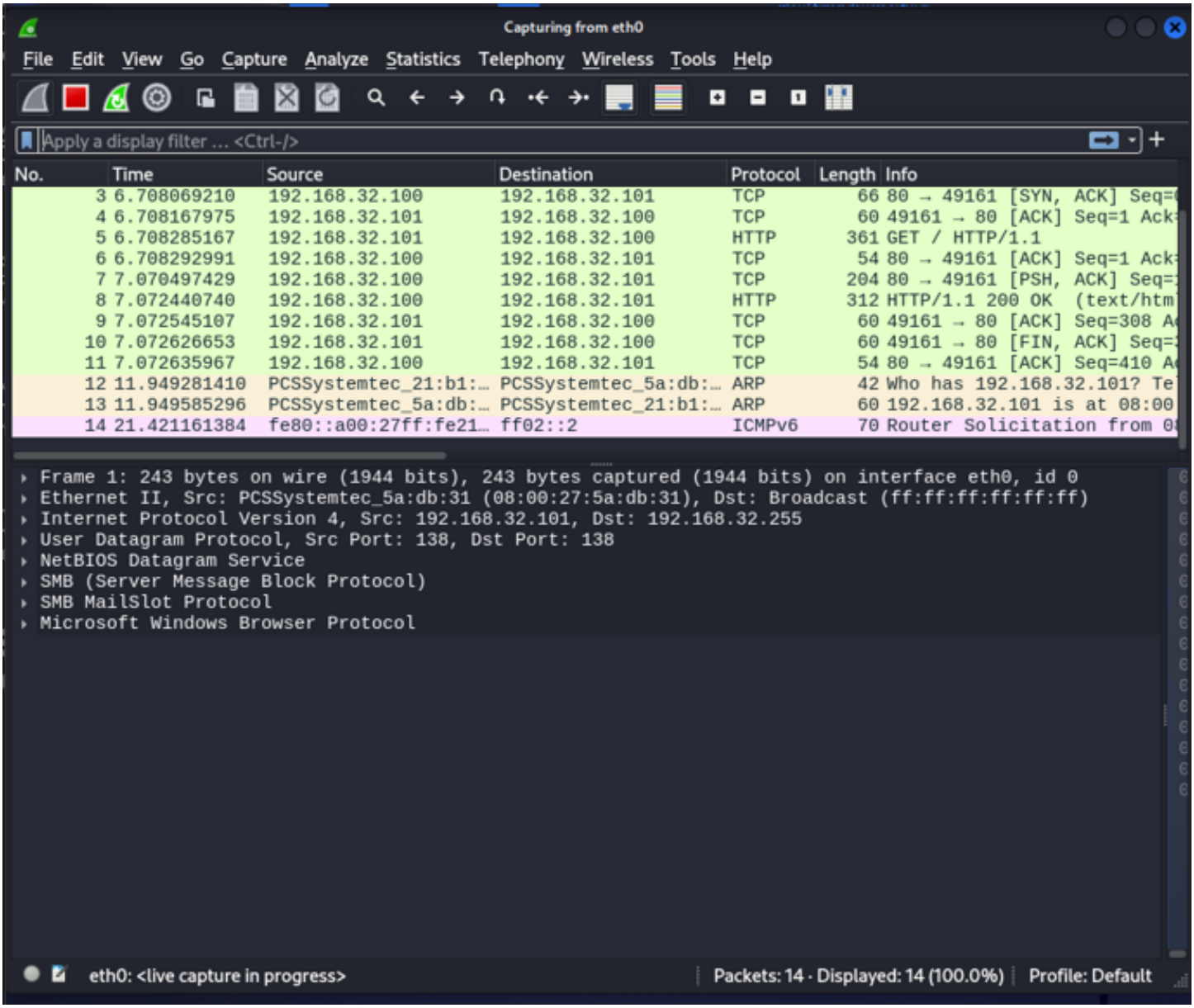


Come da screen, Il DNS risolve correttamente la pagina web convertendo l'indirizzo nell'IP della macchina Kali dove è attivo INetSim.

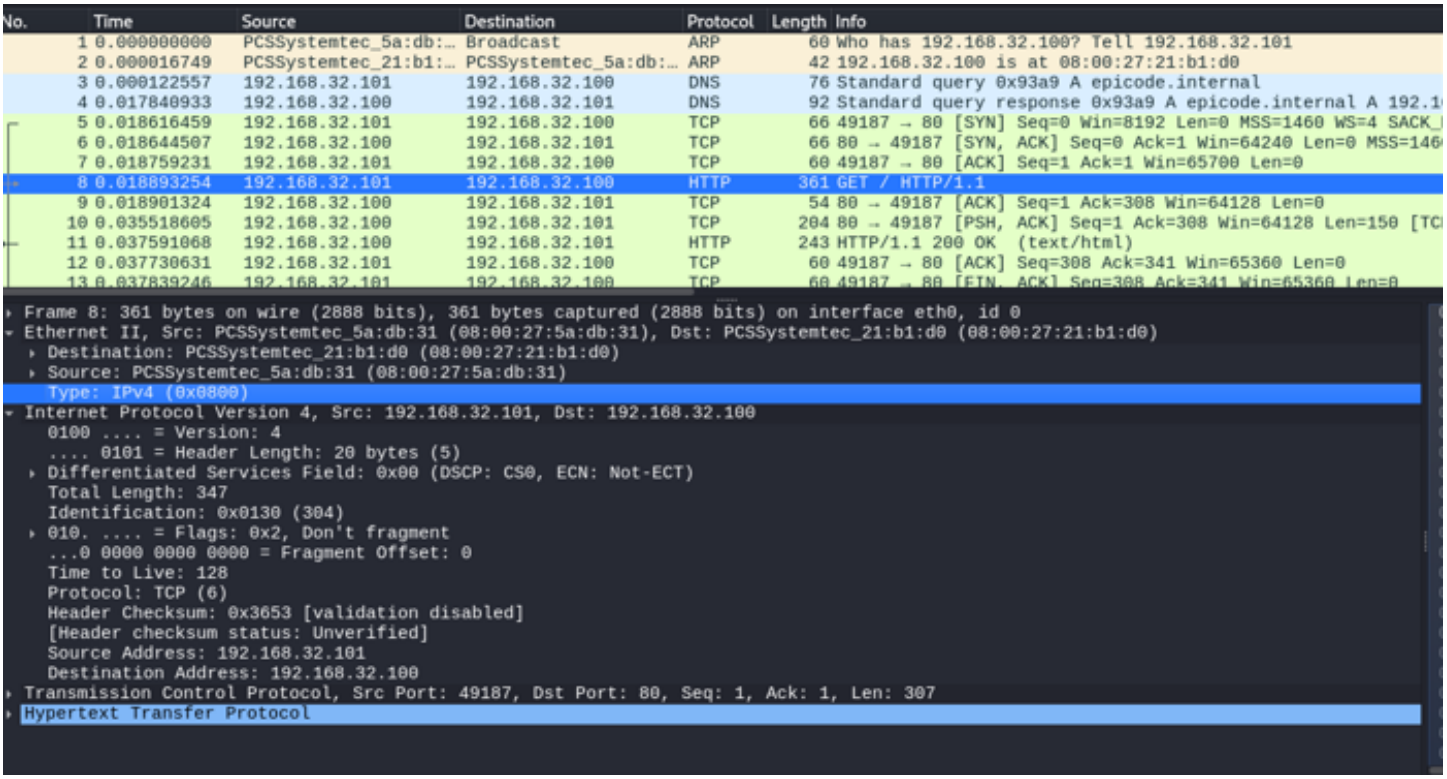
Nello stesso momento sulla Macchina Kali, Wireshark avrà sniffato i pacchetti che ha generato la richiesta fatta dalla macchina Windows:



Analizzando in dettaglio i pacchetti, si vedono i pacchetti (SYN,ACK) che è il processo di handshake a tre vie che serve a stabilire una connessione tra un client e un server. Il client avvia l'handshake a tre vie per stabilire una connessione TCP con il server.



Nel dettaglio, il pacchetto che ci interessa per l’esercizio è il pacchetto con protocollo HTTP:

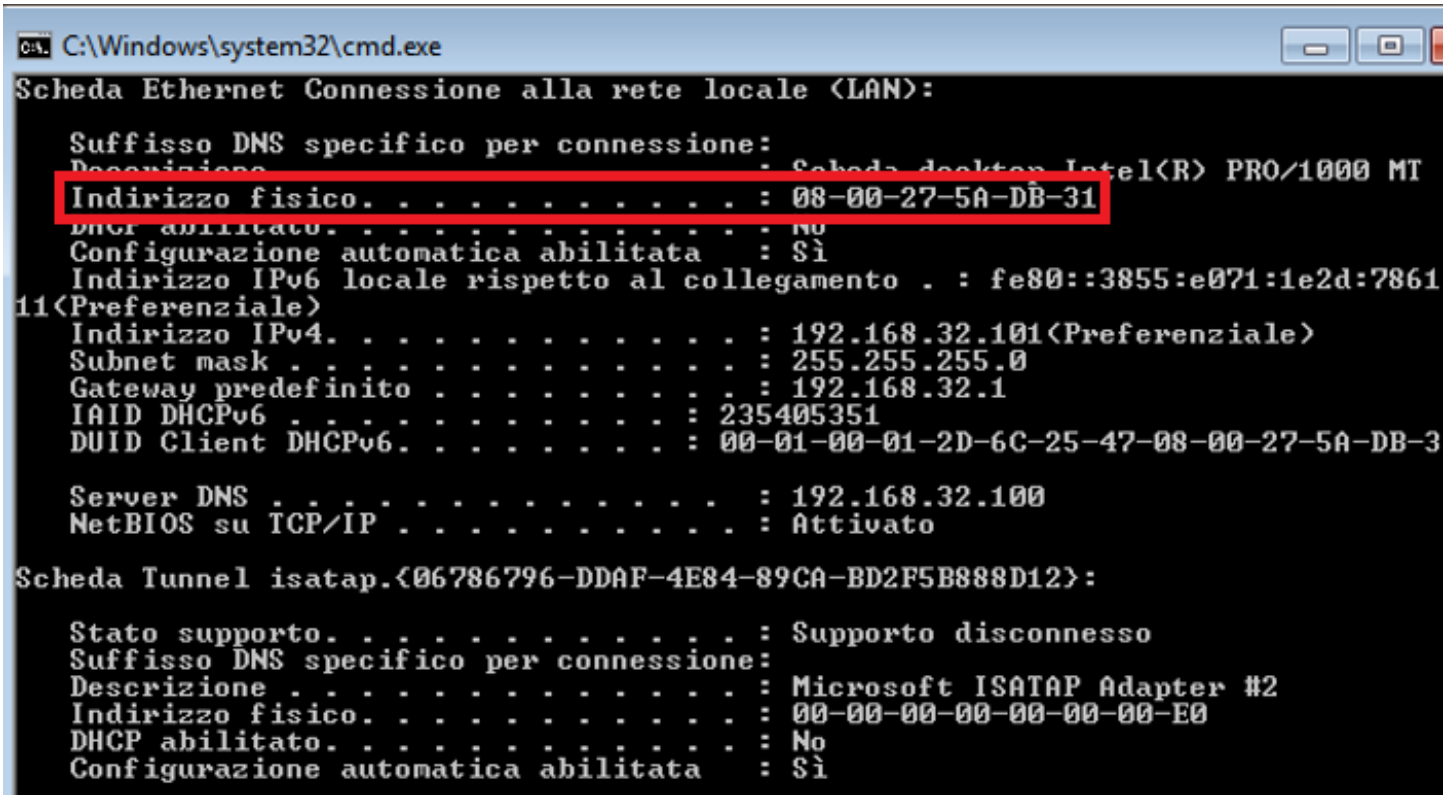


Dove su Wireshark possiamo trovare le seguenti informazioni:

Mac adress delle due macchine virtuali

Source (**08:00:27:5a:db:31**) corrispondente alla VM Windows7

Desination (**08:00:27:21:b1:d0**) corrispondente alla macchina Kali [Ether]



```
└─$ 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:fe21:b1d0  prefixlen 64  scopeid 0×20<link>
        ether 08:00:27:21:b1:d0  txqueuelen 1000  (Ethernet)
        RX packets 434  bytes 47754 (46.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 237  bytes 121324 (118.4 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 0×10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 54  bytes 4336 (4.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 54  bytes 4336 (4.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Catturato sul ETH0 (Kali)

```
Frame 13: 472 bytes on wire (3776 bits), 472 bytes captured (3776 bits)
  Section number: 1
▸ Interface id: 0 (eth0)
  Encapsulation type: Ethernet (1)
```

IP delle macchine:

Si può notare nel pacchetto sniffato l'indirizzo IP delle due macchine virtuali.

```
    [Header checksum status: Unverified]
    Source Address: 192.168.32.101
    Destination Address: 192.168.32.100
```

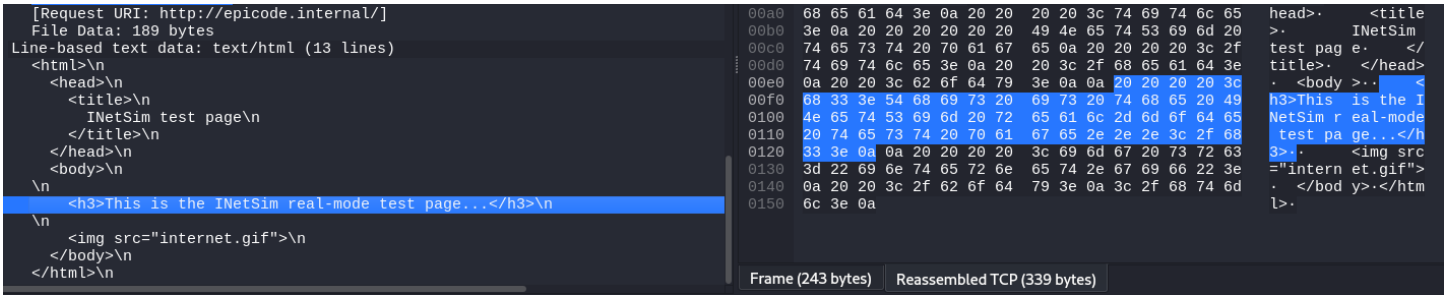
Tipo di protocollo:

SI nota che il Pacchetto è con protocollo HTTP, E che è stata usata la porta 80

```
    [Protocols in frame: eth:ethertype:ip:tcp:http]
    [Coloring Rule Name: HTTP]
    [Coloring Rule String: http || tcp.port == 80 || http2]
```

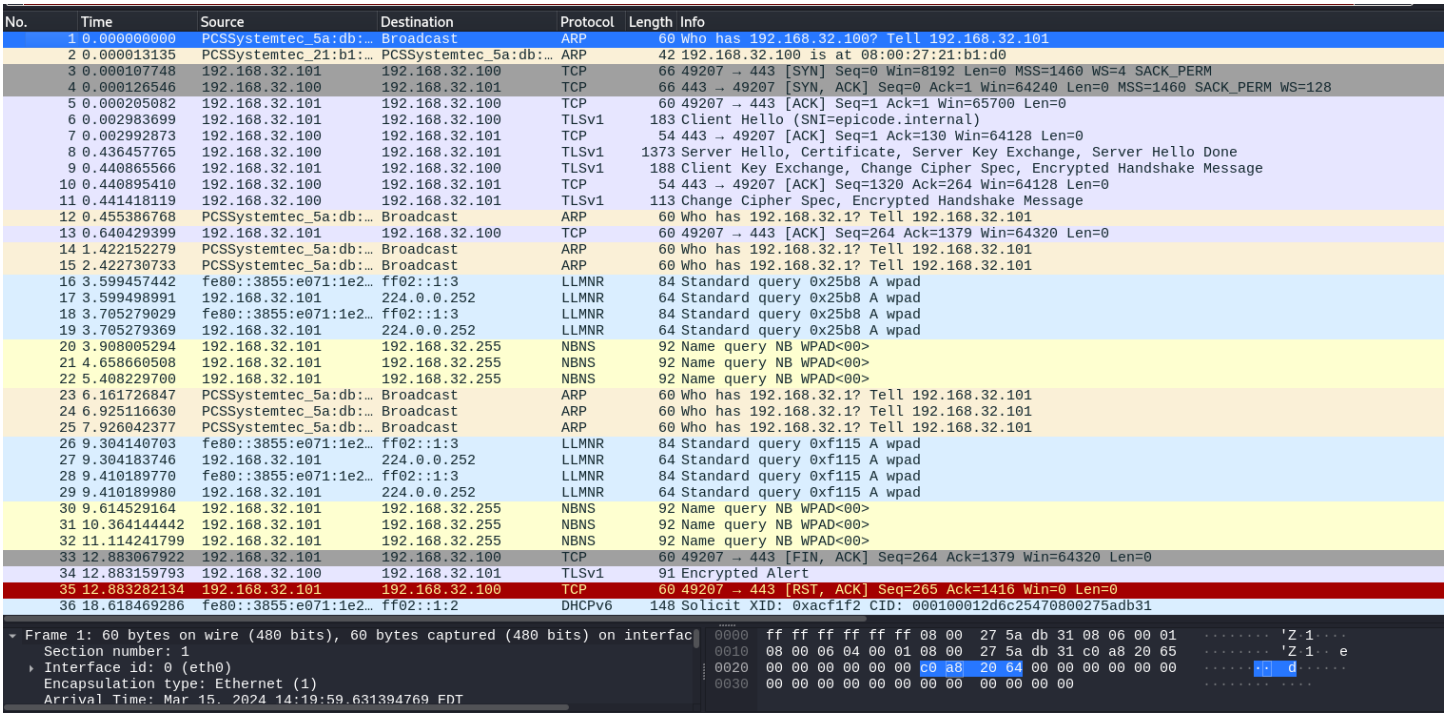
Contenuto del pacchetto:

Si nota che il contenuto sniffato di risposta alla domanda del pacchetto “get” è una stringa testuale in chiaro e un’immagine in .gif. Non è presente incriptazione essendo l’HTTP un protocollo che non la prevede quindi Header e Payload sono totalmente in chiaro.

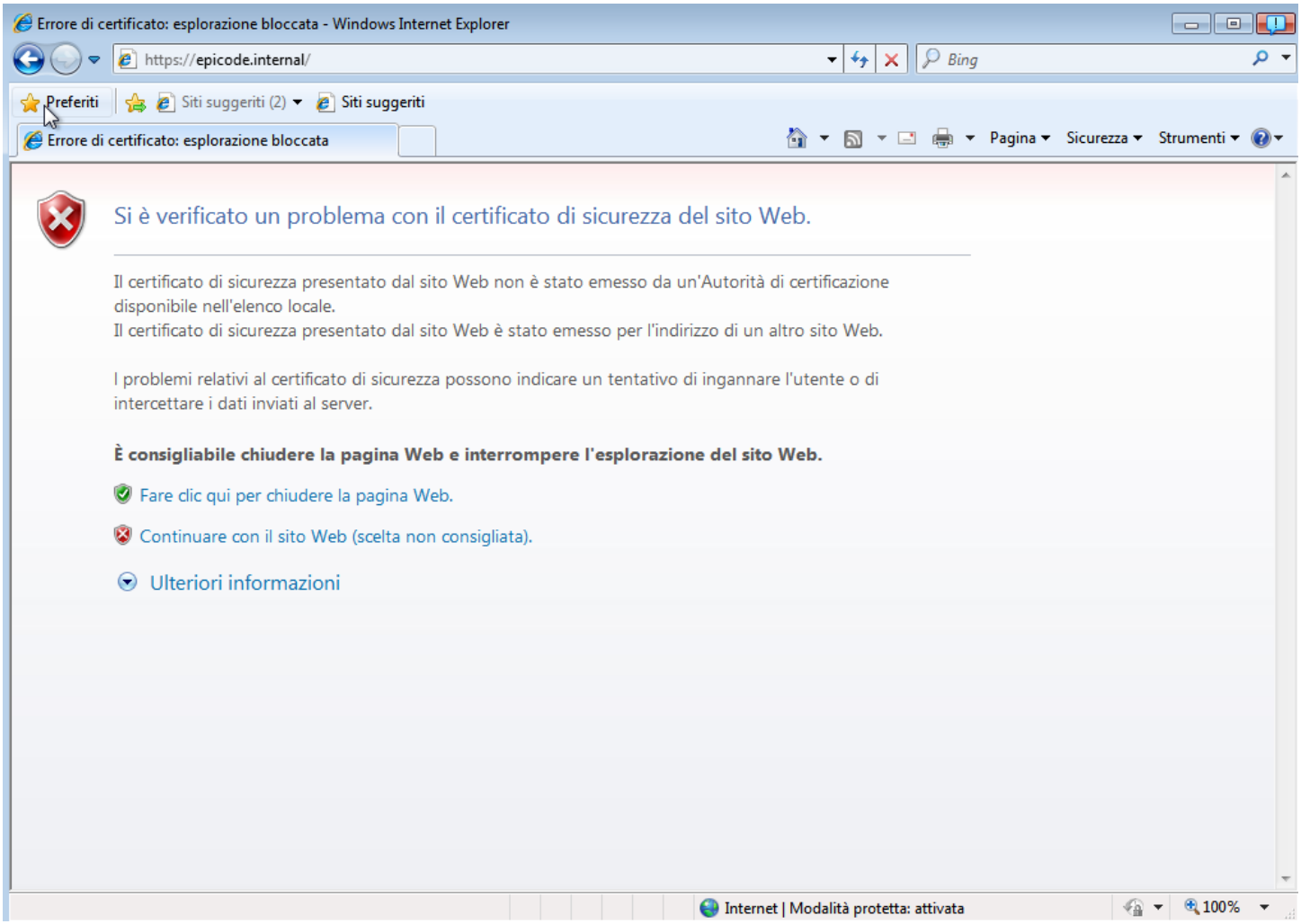


VERSIONE HTTPS

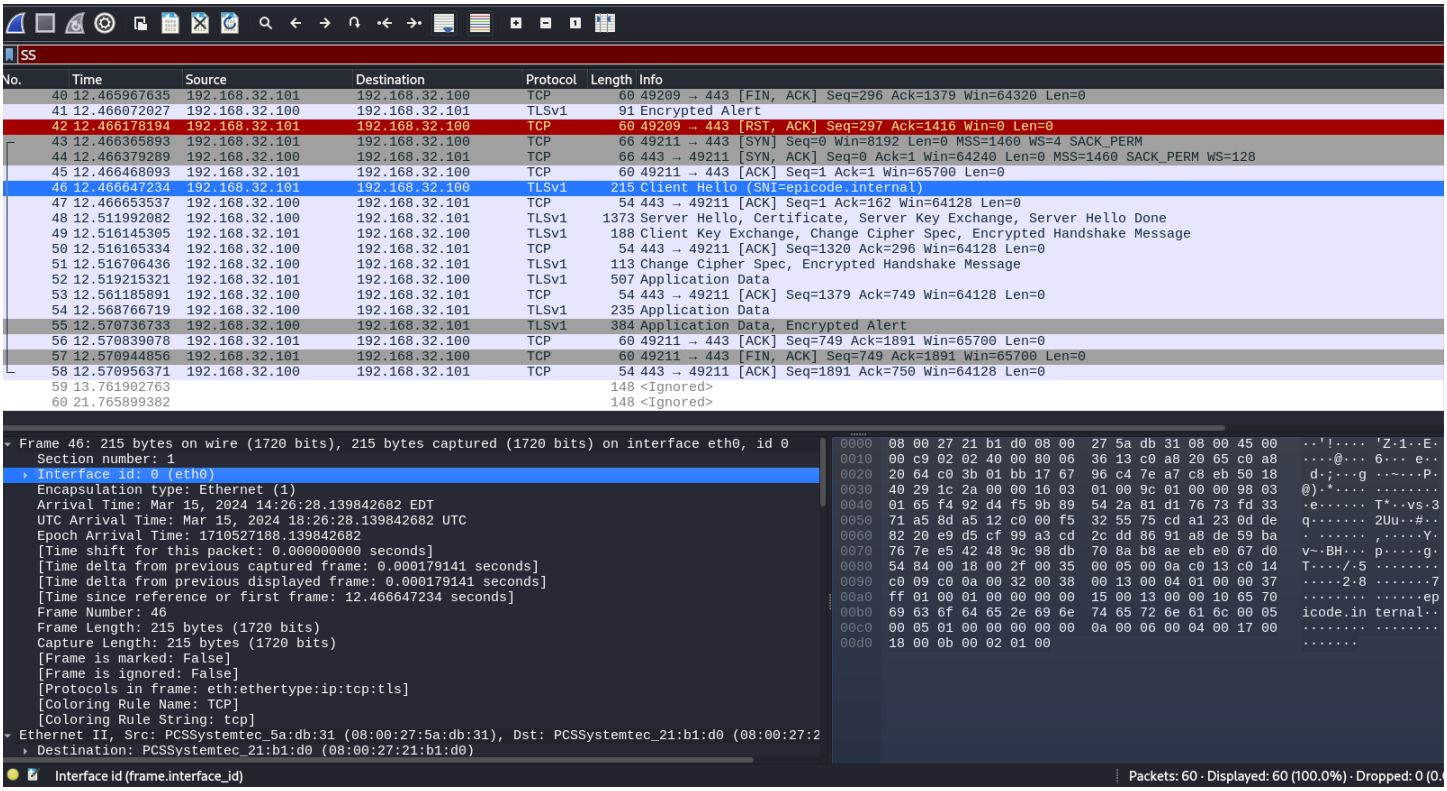
Dalla macchina Windows apriamo il browser predefinito (Internet Explorer) e nella barra indirizzo digitiamo il nostro sito web simulato con INetSim specificando il protocollo HTTPS **HTTPS**://Epicode.internal



Succederà che il browser di Windows7 blocca la renderizzazione della pagina a causa di un certificato non valido. Già questo controllo fa notare delle differenze sostanziali tra le due tipologie di protocolli (HTTP vs HTTPS)



Forziamo comunque l'apertura della nostra fake pagina web premendo su "Continuare con il sito Web" e cambiamo la visuale sulla macchina Kali



Risulteranno dei pacchetti sniffati. Notiamo subito la differenza di porta in 443 (HTTPS) e il protocollo TLSv1 (Trasport Layer Security) l'evoluzione del protocollo SSL, che sono protocolli di incriptazione.

Dettagli sul pacchetto sniffato HTTPS

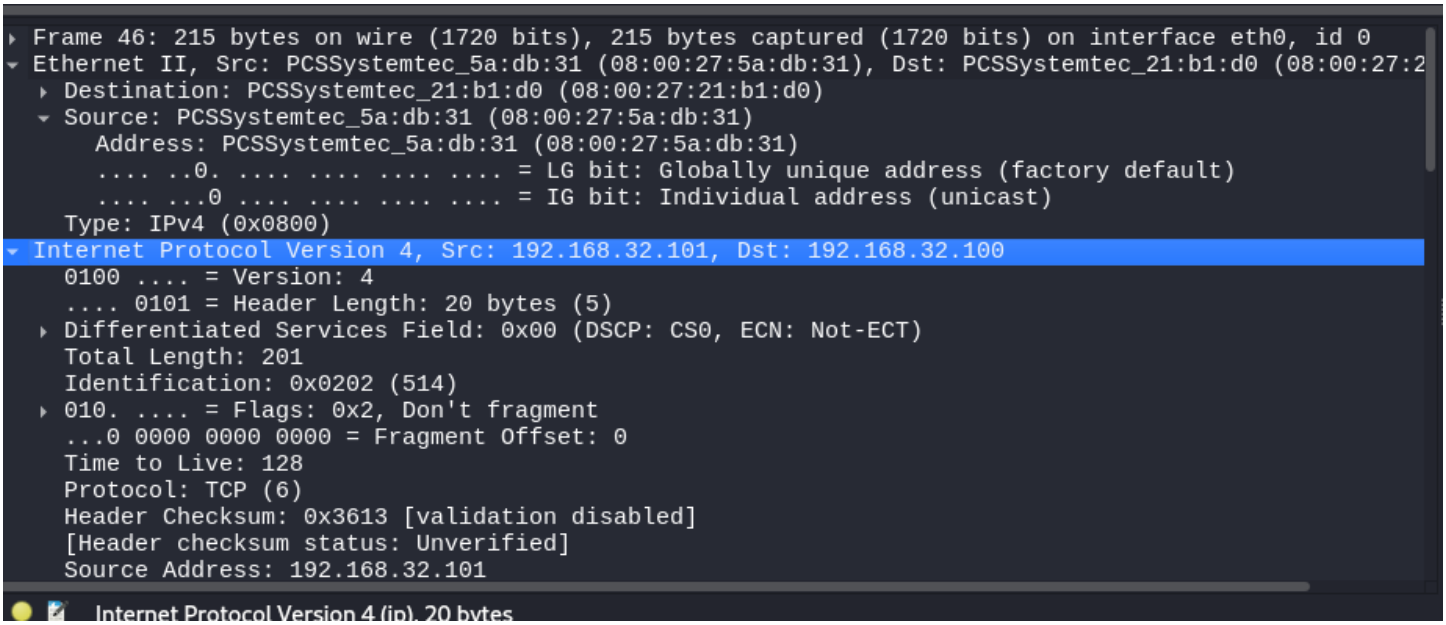
Mac address e IP delle due macchine virtuali

Come in HTTP, troviamo i due Mac Address:

Source (**08:00:27:5a:db:31**) corrispondente alla VM Windows7

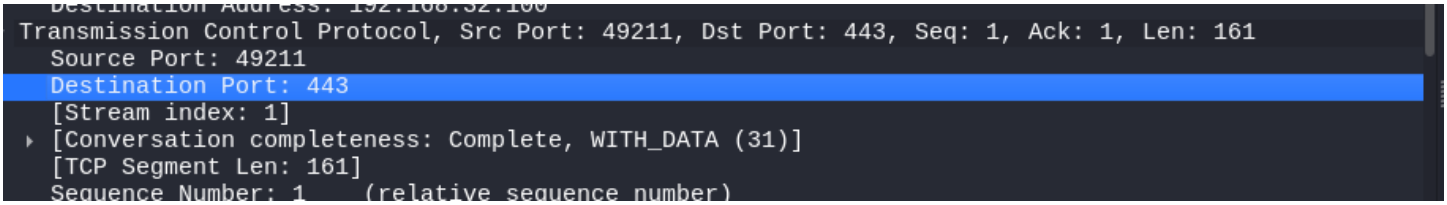
Desination (**08:00:27:21:b1:d0**) corrispondente alla macchina Kali

E i loro relativi IP



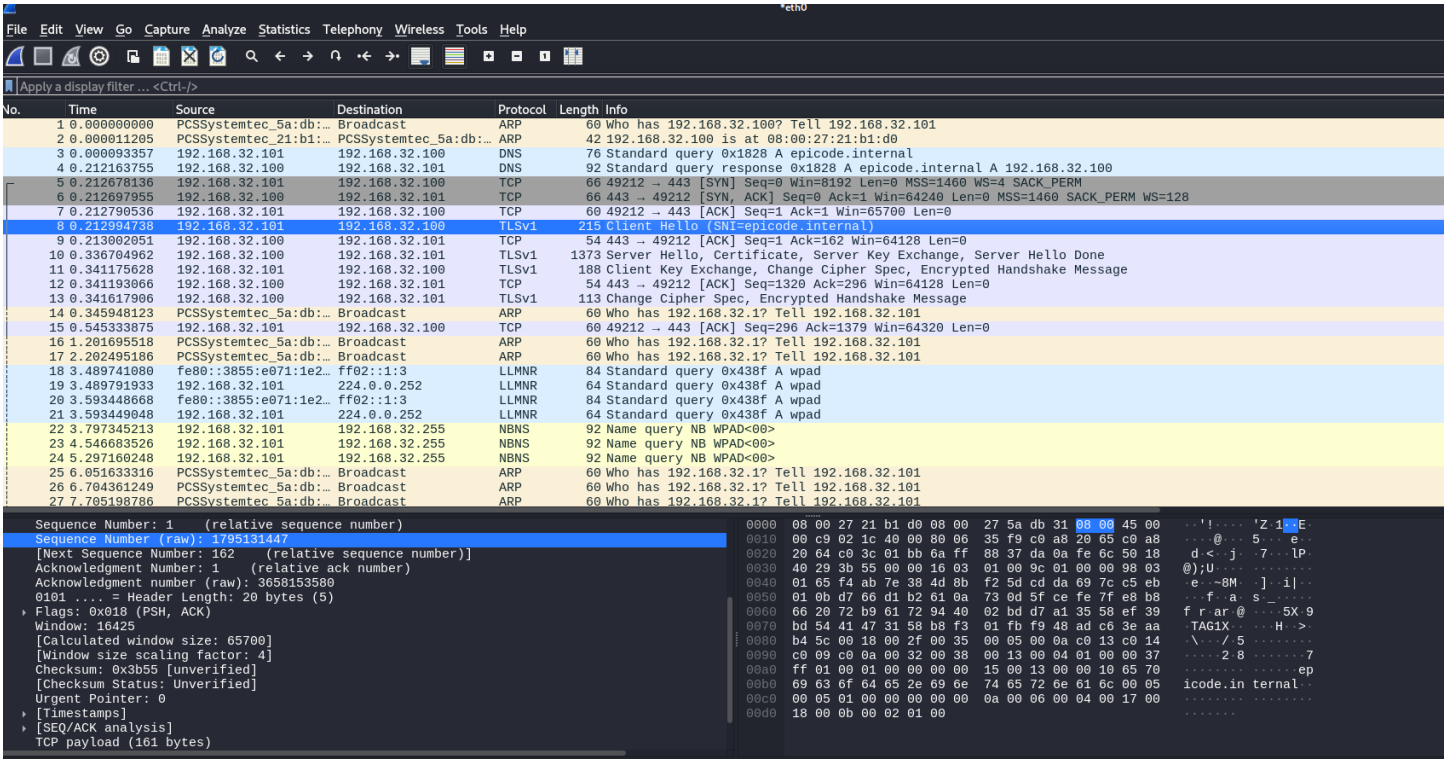
Tipo di protocollo:

Si nota a differenza dell'HTTP, l'utilizzo del TLSv1 che provvede a criptare i dati del pacchetto. La porta utilizzata è la default 443

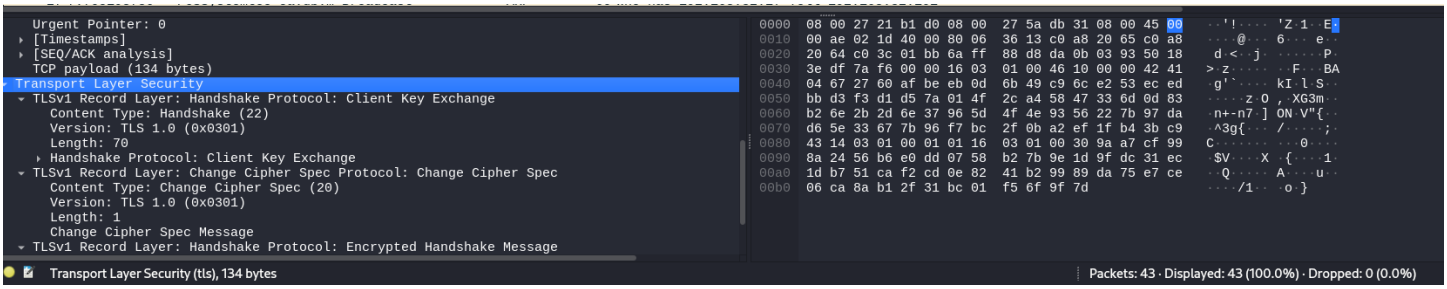


Contenuto del pacchetto:

Come si nota, il pacchetto risulta tutto cifrato



e come si vede il pacchetto continene il TLS che previene la visualizzazione del contenuto del pacchetto in chiaro, ma risultano solo stringe di testo “casuali”



CONCLUSIONI:

HTTP (Hypertext Transfer Protocol) è il protocollo di comunicazione utilizzato per trasferire dati tra un browser web e un sito web. È standard, ma non crittografa i dati, rendendoli vulnerabili agli attacchi.

Un esempio è quando un utente visita una pagina web tramite il proprio browser. Il browser invia una richiesta HTTP al server web per ottenere la pagina richiesta, e il server risponde inviando il contenuto della pagina. Questo avviene senza crittografia, quindi i dati sono inviati in chiaro e potrebbero essere intercettati da terze parti.

HTTPS (HTTP Secure) aggiunge un livello di crittografia SSL o TLS, garantendo una comunicazione sicura e proteggendo i dati degli utenti durante la navigazione online.

Un esempio è quando un utente si connette al proprio home banking. Durante questa connessione, il browser e il server della banca negoziano un protocollo TLS per crittografare tutti i dati scambiati, come informazioni di accesso e transazioni finanziarie, proteggendo così le informazioni sensibili degli utenti durante il trasferimento.