

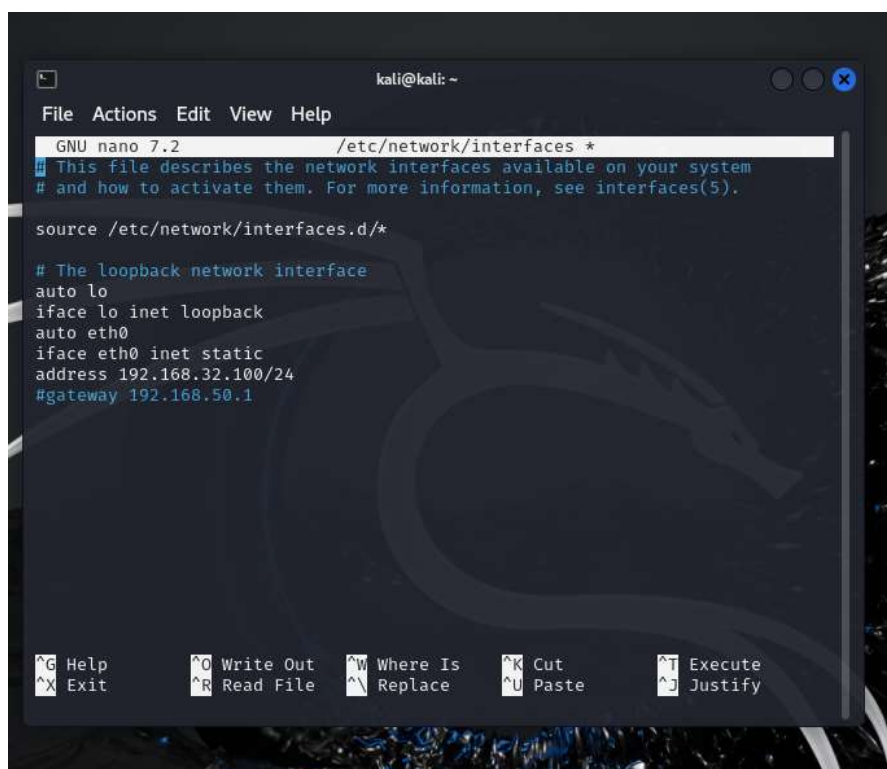
Nell'esercizio di oggi metteremo insieme le competenze acquisite finora. Lo studente verrà valutato sulla base della risoluzione al problema seguente.

Requisiti e servizi:

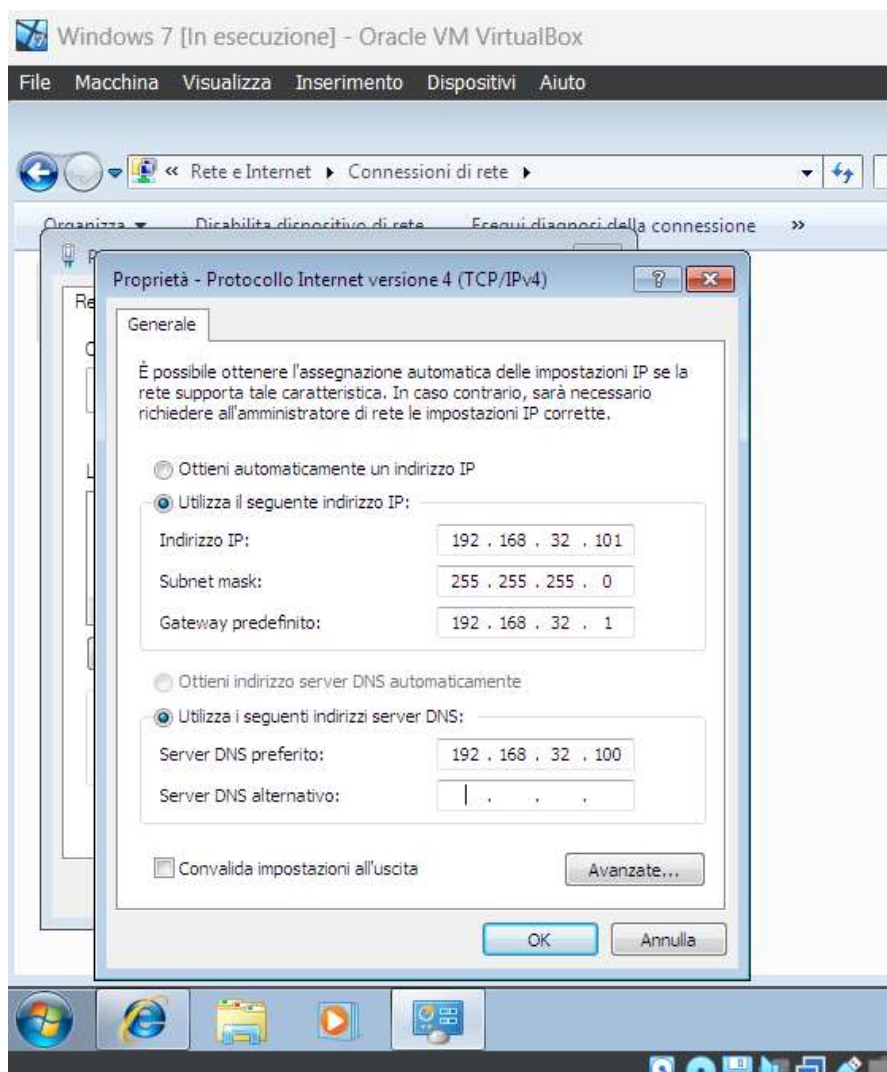
- Kali Linux: IP 192.168.32.100
- Windows 7: IP 192.168.32.101
- HTTPS server: attivo
- Servizio DNS per risoluzione nomi di dominio: attivo

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

Come prima cosa bisogna configurare le nostre macchine con gli indirizzi IP citati sopra, per Kali andremo da terminale e tramite il comando `< sudo nano /etc/network/interfaces >` potremo avviare le nostre modifiche mentre per windows bisogna andare su centro connessione di rete e condivisione - modifica impostazioni scheda, con il destro cliccare su connessione rete locale - proprietà - protocollo internet 4 (TCP/IPv4) - proprietà e inserisci l'indirizzo IP, qui bisogna inserire anche l'indirizzo del DNS che sarà lo stesso 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 192.168.50.1  
  
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute  
^X Exit      ^R Read File  ^_ Replace    ^U Paste      ^J Justify
```



Una volta impostati gli indirizzi bisogna attivare il server Https e il servizio DNS per fare ciò digitare il comando `< sudo nano /etc/inetsim/inetsim.conf >` per entrare nella configurazione di inetsim e apportare le modifiche necessarie

```

GNU nano 8.0
# time_udp, daytime_tcp, d
# echo_udp, discard_tcp, d
# quotd_udp, chargen_tcp,
# ident, syslog, dummy_tcp
# 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

#
# Default: 127.0.0.1
#
service_bind_address 192.168.32.100

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

#
# Default: inetsim.org
#
dns_default_domainname epicode.internal

#####
# dns_static
#
# Static mappings for DNS
#
#Syntax: dns_static <fqdn hostname> <IP address>

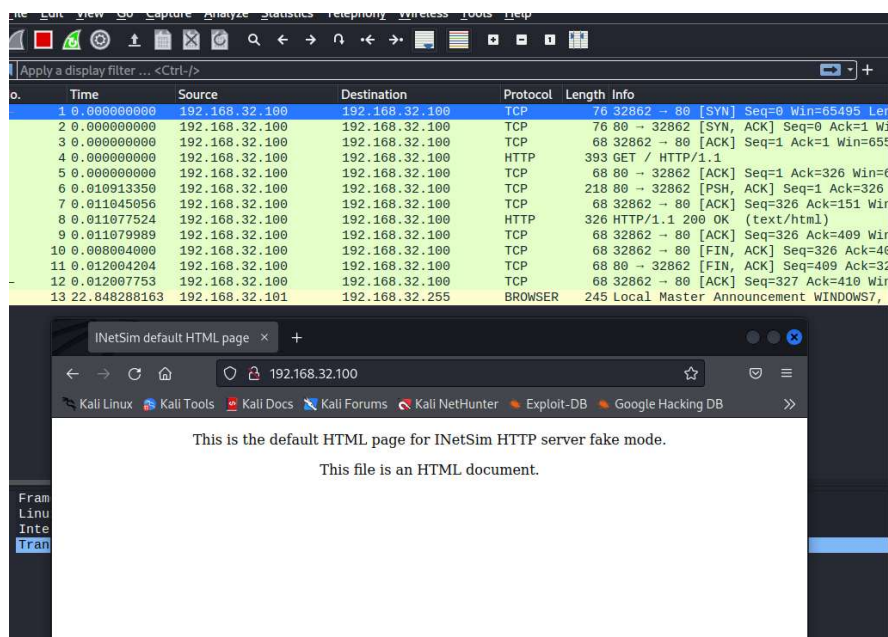
```

```
#####
# dns_default_hostname
#
# Default hostname to return with DNS replies
#####

# http_default_fakefile
#
# The default fake file returned in fake mode
# in the HTTP request does not match any of the
# defined above.
#
# The default fake file must be placed in <data
#
# Syntax: http_default_fakefile <filename> <mime
#
# Default: none
#
http_default_fakefile sample.html text/html

#####
# http_static_fakefile
#
# Static mappings for DNS
#
# Syntax: dns_static <fqdn hostname> <IP address>
#
# Default: yes
#
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
```

Ora che il servizio DNS e il server HTTPS sono attivi avviando la simulazione di rete con il comando `<sudo inetsim>` e apriamo il nostro Wireshark per intercettare il traffico, ora se apriamo il browser di kali e digitiamo il suo indirizzo IP sulla barra di ricerca si aprirà una pagina HTML di INetSim di un finto server HTTP e Wireshark inizierà a catturare il traffico.



Ciò che a noi interessa per lo svolgimento dell'esercizio però è cercare tramite il browser di Windows7 la risorsa `epicode.internal` e intercettare la comunicazione che avviene tra Windows7 e Kali evidenziando MAC address e il contenuto della richiesta HTTPS.

Per farlo aprire il browser da Windows7 e scrivere `<https://epicode.internal>` ti dovrà aprire la stessa pagina HTML che abbiamo visto sopra, poi fare lo stesso con `http`.

HTTPS:

The image shows a Wireshark capture of HTTPS traffic and a corresponding Internet Explorer window. The Wireshark interface displays a list of packets with columns for Time, Source, Destination, Protocol, and Length. The selected packet (No. 12) is a TCP segment from 192.168.32.101 to 192.168.32.100, port 443. The packet details pane shows the TCP header and the application data, which is an HTML document. The Internet Explorer window shows the default HTML page for the DNetSim HTTP server fake mode.

Time	Source	Destination	Protocol	Length	Info
0.000000000	PcsCompu_c8:d5:74	192.168.32.101	ARP	62	Who has 192.168.32.100? Tell 192.168.32.101
0.000000000	PcsCompu_b7:1f:c6	192.168.32.101	ARP	44	192.168.32.100 is at 08:00:27:b7:1f:c6
0.000000000	192.168.32.101	192.168.32.100	TCP	68	49191 → 443 [SYN] Seq=0 Win=0 MSS=1460 WS=4 SACK_PER
0.000000000	192.168.32.101	192.168.32.101	TCP	68	443 → 49191 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 S
0.000000000	192.168.32.101	192.168.32.100	TCP	62	49191 → 443 [ACK] Seq=1 Ack=1 Win=0 Len=0
0.000000000	192.168.32.101	192.168.32.100	TLSv1	217	Client Hello
0.000000000	192.168.32.100	192.168.32.101	TCP	56	443 → 49191 [ACK] Seq=1 Ack=102 Win=64128 Len=0
0.000000000	192.168.32.100	192.168.32.101	TLSv1	1375	Server Hello, Certificate, Server Key Exchange, Server Hello ...
0.011488186	192.168.32.101	192.168.32.100	TLSv1	198	Client Key Exchange, Change Cipher Spec, Encrypted Handshake ...
0.011496576	192.168.32.100	192.168.32.101	TCP	56	443 → 49191 [ACK] Seq=1320 Ack=296 Win=64128 Len=0
0.012007850	192.168.32.100	192.168.32.101	TLSv1	115	Change Cipher Spec, Encrypted Handshake Message
0.019477344	PcsCompu_c8:d5:74	192.168.32.101	ARP	62	Who has 192.168.32.17? Tell 192.168.32.101
0.021490630	192.168.32.101	192.168.32.100	TCP	62	49191 → 443 [ACK] Seq=296 Ack=1379 Win=64320 Len=0
0.021490630	PcsCompu_c8:d5:74	192.168.32.101	ARP	62	Who has 192.168.32.17? Tell 192.168.32.101
0.021490630	PcsCompu_c8:d5:74	192.168.32.101	ARP	62	Who has 192.168.32.17? Tell 192.168.32.101
0.021490630	192.168.32.101	192.168.32.255	NBNS	94	Name query NB WPAD>00>
0.021490630	192.168.32.101	192.168.32.255	NBNS	94	Name query NB WPAD>00>
0.021490630	192.168.32.101	192.168.32.255	NBNS	94	Name query NB WPAD>00>
0.021490630	PcsCompu_b7:1f:c6	192.168.32.101	ARP	44	Who has 192.168.32.101? Tell 192.168.32.100
0.021490630	PcsCompu_c8:d5:74	192.168.32.101	ARP	62	192.168.32.101 is at 08:00:27:b7:1f:c6
0.021490630	PcsCompu_c8:d5:74	192.168.32.101	ARP	62	Who has 192.168.32.17? Tell 192.168.32.101
0.021490630	PcsCompu_c8:d5:74	192.168.32.101	ARP	62	Who has 192.168.32.17? Tell 192.168.32.101
0.021490630	PcsCompu_c8:d5:74	192.168.32.101	ARP	62	Who has 192.168.32.17? Tell 192.168.32.101
0.021490630	192.168.32.101	192.168.32.255	NBNS	94	Name query NB WPAD>00>

Protocol: TCP (6)
Header checksum: 0x56e [validation disabled]
[Header checksum status: Unverified]
Source Address: 192.168.32.101
Destination Address: 192.168.32.100
Transmission Control Protocol, Src Port: 49191, Dst Port: 443, Seq: 162, Ack: 1320, Len: 134
Source Port: 49191
Destination Port: 443
[Stream index: 0]
[Conversation completeness: Incomplete, DATA (15)]
[TCP Segment Len: 134]
Sequence Number: 162 (relative sequence number)
Sequence Number (raw): 4132938684
[Next Sequence Number: 296 (relative sequence number)]
0000 00 00 00 01 00 00 00 00 27 c8 d5 74 00 00 00 00

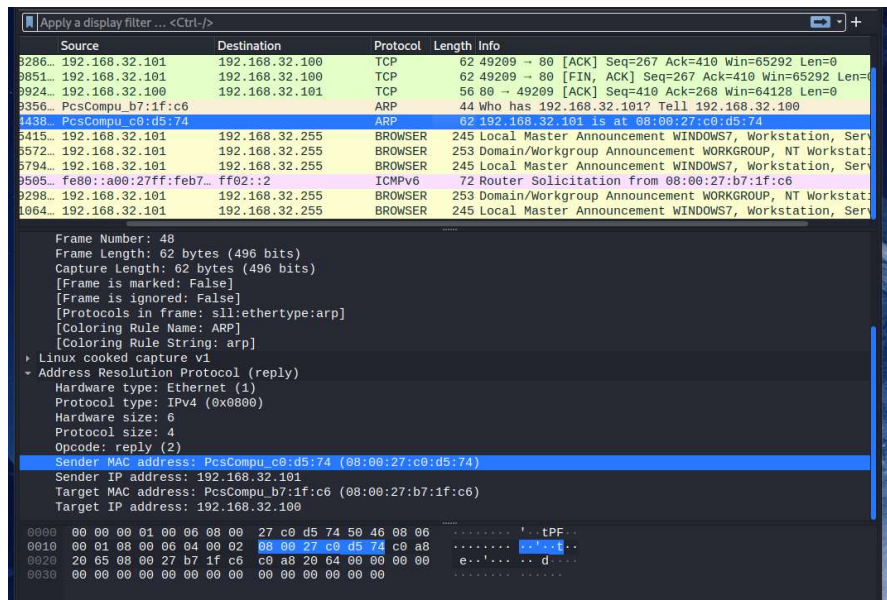
HTTP:

The image shows a Wireshark capture of HTTP traffic and a corresponding Internet Explorer window. The Wireshark interface displays a list of packets with columns for Source, Destination, Protocol, and Length. The selected packet (No. 12) is a TCP segment from 192.168.32.101 to 192.168.32.100, port 443. The packet details pane shows the TCP header and the application data, which is an HTML document. The Internet Explorer window shows the default HTML page for the DNetSim HTTP server fake mode.

Source	Destination	Protocol	Length	Info
192.168.32.101	192.168.32.255	NBNS	94	Name query NB WPAD>00>
192.168.32.101	192.168.32.255	NBNS	94	Name query NB WPAD>00>
192.168.32.101	192.168.32.100	TCP	68	49190 → 80 [SYN] Seq=0 Win=8192 L
192.168.32.100	192.168.32.101	TCP	68	80 → 49190 [SYN, ACK] Seq=0 Ack=1
192.168.32.101	192.168.32.100	TCP	62	49190 → 80 [ACK] Seq=1 Ack=1 Win=
192.168.32.101	192.168.32.100	HTTP	273	GET /msdownload/update/v3/static/
192.168.32.100	192.168.32.101	TCP	56	80 → 49190 [ACK] Seq=1 Ack=218 W
192.168.32.100	192.168.32.101	TCP	200	80 → 49190 [PSH, ACK] Seq=1 Ack=2
192.168.32.101	192.168.32.100	TCP	62	49190 → 80 [ACK] Seq=218 Ack=410
192.168.32.101	192.168.32.100	TCP	62	49190 → 80 [FIN, ACK] Seq=218 Ack
192.168.32.100	192.168.32.101	TCP	56	80 → 49190 [ACK] Seq=410 Ack=219
192.168.32.101	192.168.32.100	TCP	62	49185 → 443 [FIN, ACK] Seq=264 A
192.168.32.100	192.168.32.101	TLSv1	93	Encrypted Alert
192.168.32.100	192.168.32.101	TCP	56	443 → 49185 [FIN, ACK] Seq=1416 A
192.168.32.101	192.168.32.100	TCP	62	49185 → 443 [RST, ACK] Seq=265 A
08:00:27ff:feb7..ff02::2	72	Router Solicitation from 08:00:27		

Protocol: TCP (6)
Header checksum: 0x584c [validation disabled]
[Header checksum status: Unverified]
Source Address: 192.168.32.100
Destination Address: 192.168.32.101
Transmission Control Protocol, Src Port: 80, Dst Port: 49190, Seq: 161, Ack: 218, Len: 1
[2 Reassembled TCP Segments (408 bytes): #630(150), #631(258)]
Hypertext Transfer Protocol
Line-based text data: Text/html (10 lines)
Frame (314 bytes) Reassembled TCP (408 bytes)
Header Checksum (ip.checksum), 2 bytes | Packets: 639 - Displayed: 639 (100.0%) | Profile: Default

Comparirà una richiesta ARP con scritto < Who has e l'indirizzo IP in questione > qui si posso individuare gli indirizzi Mac di sorgente e ricevente



Adesso passiamo all'ultima parte dell'esercizio dove chiede se abbiamo notato differenze tra il traffico dei dati del server HTTPS e quello HTTP.

Sì, tra i due Server possiamo notare diverse differenze tra cui:

Cifratura dei dati:

HTTP: il traffico non è cifrato, di conseguenza possiamo leggere tutto il contenuto delle richieste e delle risposte in chiaro

HTTPS: il traffico è cifrato e come possiamo vedere dall'immagine sopra utilizza TLS (Transport Layer Security), di conseguenza il contenuto delle richieste e delle risposte non è leggibile

Porte di comunicazione:

HTTP: utilizza la porta 80

HTTPS: utilizza la porta 443

Three Way Handshake:

HTTP: non ha fasi di negazione, qui la comunicazione inizia immediatamente con l'invio della richiesta HTTP

HTTPS: utilizza una fase di handshake iniziale per stabilire la connessione sicura. Qui vengono negoziati i parametri di sicurezza e vengono scambiati certificati per autenticare server e client