

## 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 con 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.

### Requisiti:

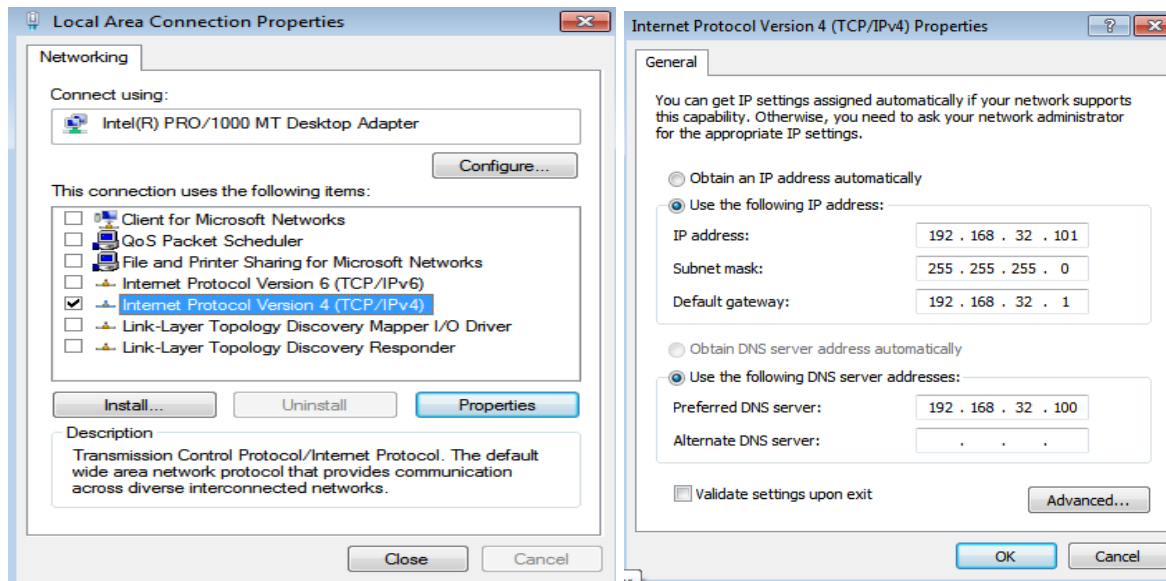
- Kali Linux - IP 192.168.32.100
- Windows 7 - IP 192.168.32.101
- HTTPS server: attivo
- Servizio DNS per risoluzione nomi di dominio: attivo

## 1- impostazione indirizzo IP statico di Windows 7

Per far comunicare le 2 macchine Kali e Windows su una rete interna dobbiamo impostare degli IP statici.

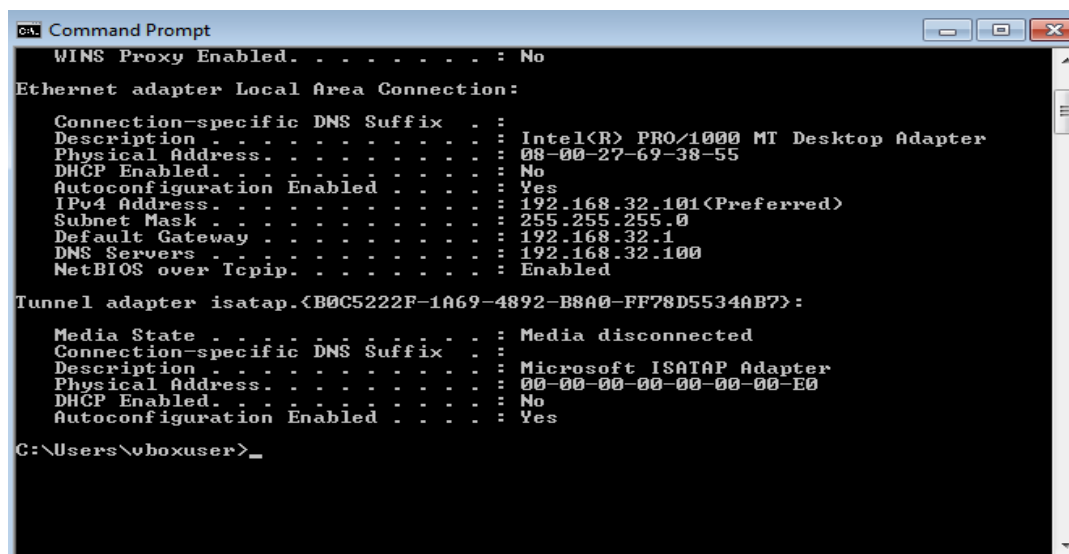
Per quanto riguarda Windows dobbiamo entrare dal pannello di controllo alle impostazioni avanzate di rete. Qui possiamo impostare l'indirizzo IP statico e anche l'indirizzo del server DNS che ci aiuterà a risolvere la chiamata `epicode.internal`.

Di seguito i passaggi fatti:



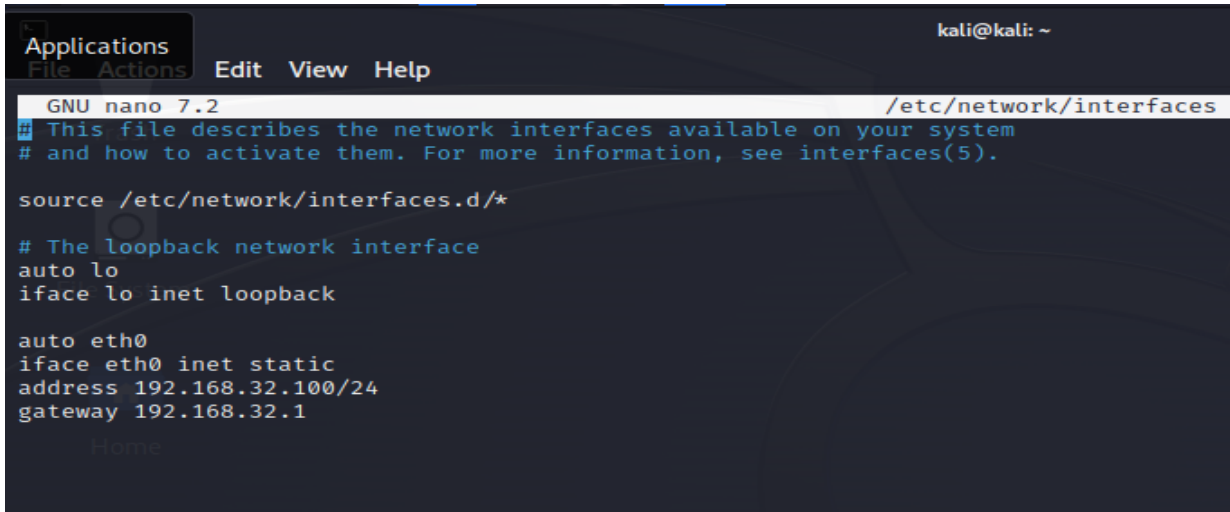
Come mostra lo screen abbiamo assegnato come indirizzo del server dns 192.168.32.100 (Ip di Kali)

Dopo aver fatto la seguente configurazione tramite command prompt con il comando "ipconfig /all" andiamo a verificare che le impostazioni siano salvate con successo.



## 2- Configurazione indirizzo IP di Kali Linux

Una volta aperto il terminale con il comando “sudo nano /etc/network/interfaces” vado a modificare l’IP e inserisco l’IP 192.168.32.100/24 e come gateway 192.168.32.1.



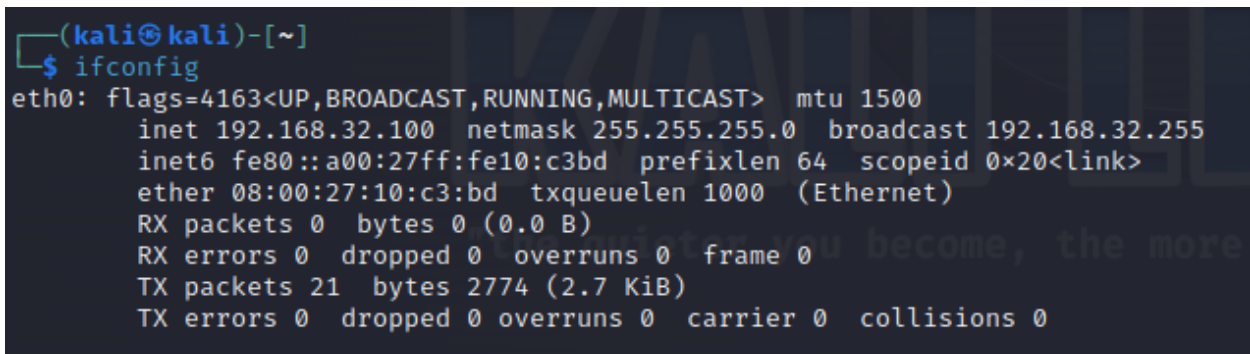
```
Applications
File Actions Edit View Help
GNU nano 7.2 /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

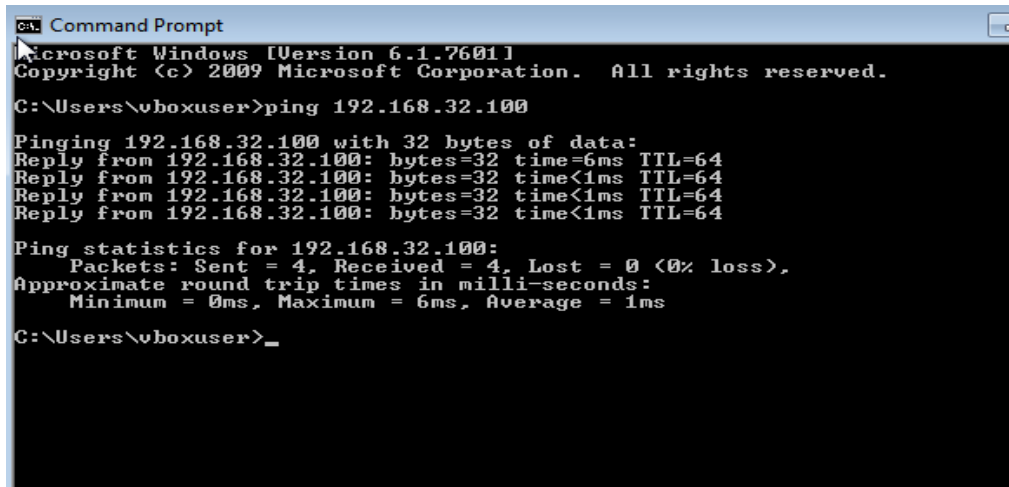
auto eth0
iface eth0 inet static
address 192.168.32.100/24
gateway 192.168.32.1
```

Come passo successivo andiamo a verificare che le impostazioni siano salvate correttamente con il comando ifconfig.



```
(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:fe10:c3bd prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:10:c3:bd txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 21 bytes 2774 (2.7 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

**3- Come ulteriore test effettuiamo un ping tra le due macchine per verificare che possano effettivamente comunicare.**



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

Pinging 192.168.32.100 with 32 bytes of data:
Reply from 192.168.32.100: bytes=32 time=6ms TTL=64
Reply from 192.168.32.100: bytes=32 time<1ms TTL=64
Reply from 192.168.32.100: bytes=32 time<1ms TTL=64
Reply from 192.168.32.100: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.32.100:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 6ms, Average = 1ms

C:\Users\vboxuser>_
```

#### **4- Configurazione del server HTTPS, HTTP e DNS.**

In Kali troviamo un simulatore di servizi internet “inetsim” che utilizzeremo per configurare ed attivare i server richiesti (https, http, DNS).

I comandi da utilizzare sono:

1. `cd /etc/inetsim`
2. `ls`
3. `sudo nano inetsim.conf`

Commentiamo tutti i service attivi tranne quelli che ci interessano ovvero HTTPS, HTTP, DNS.

```

# 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

```

Impostiamo un nuovo indirizzo IP rispetto a quello di default.

```

# 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

```

Andiamo a questo punto ad inserire un nuovo DNS statico 192.168.32.100 epicode.internal che è associato all'IP che abbiamo configurato a Kali.

```

#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

```

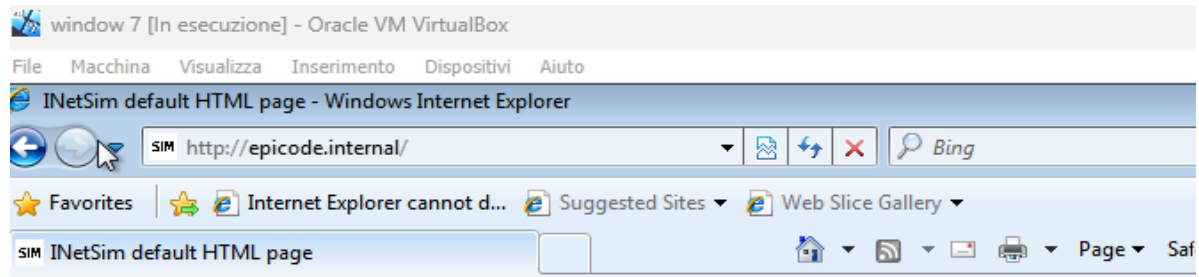
Facciamo partire la simulazione con “sudo inetsim” per verificare che i 3 servizi richiesti siano stati attivati correttamente.

```
(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 23394) ==
Session ID:      23394
Listening on:    0.0.0.0
Real Date/Time:  2023-11-17 20:25:35
Fake Date/Time: 2023-11-17 20:25:35 (Delta: 0 seconds)
Forking services ...
 * dns_53_tcp_udp - started (PID 23404)
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 23405)
 * https_443_tcp - started (PID 23406)
done.
Simulation running.
█
```

## 5- Richiedere tramite web browser una risorsa all'hostname epicode.internal

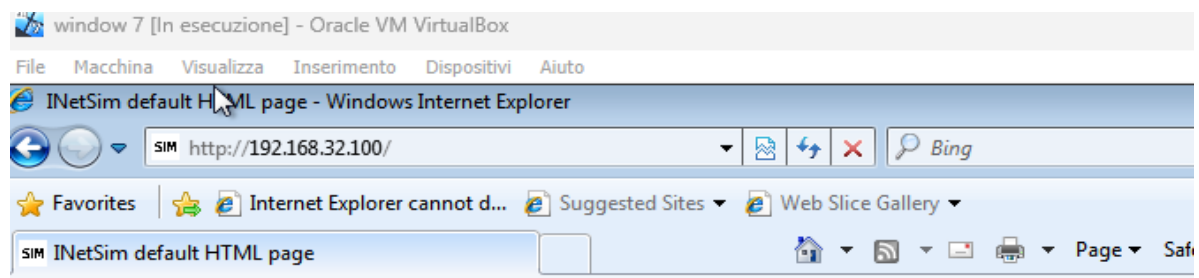
Da Windows 7 richiamiamo tramite internet explorer una richiesta HTTP e HTTPS all'hostname epicode.internal e anche tramite il suo indirizzo IP per verificare di aver configurato il DNS correttamente.

## Richiesta http:



This is the default HTML page for INetSim HTTP server fake mode.

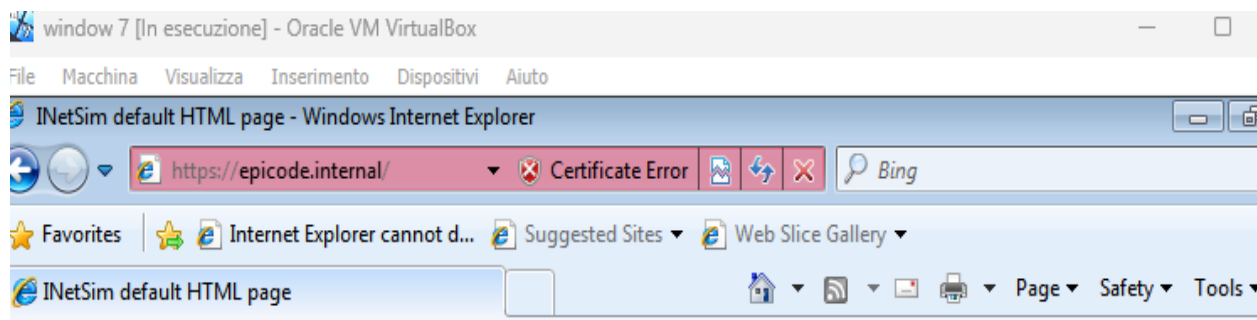
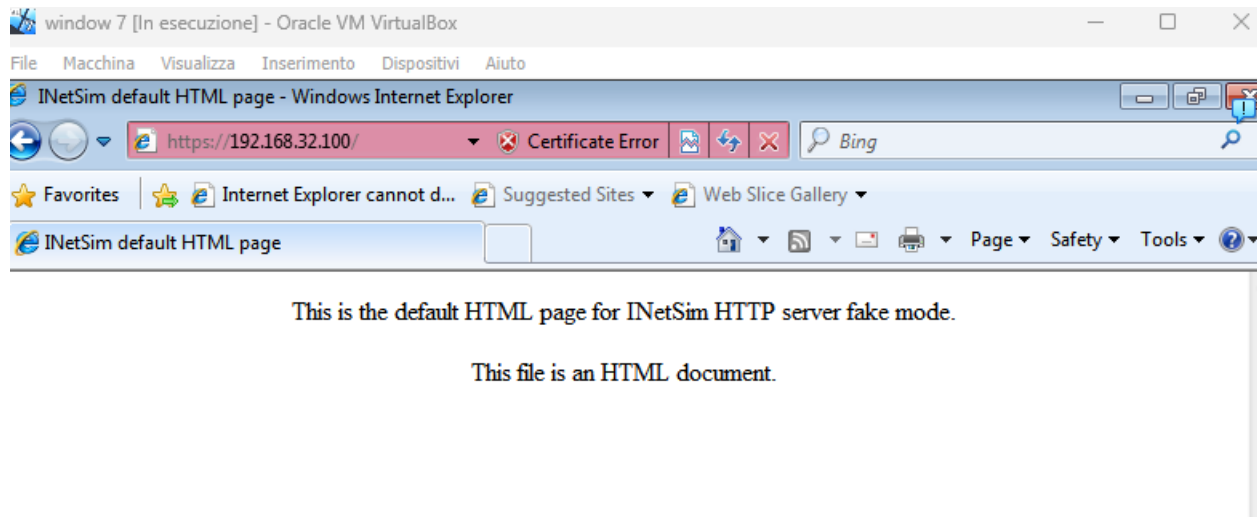
This file is an HTML document.



This is the default HTML page for INetSim HTTP server fake mode.

This file is an HTML document.

## Richiesta https:



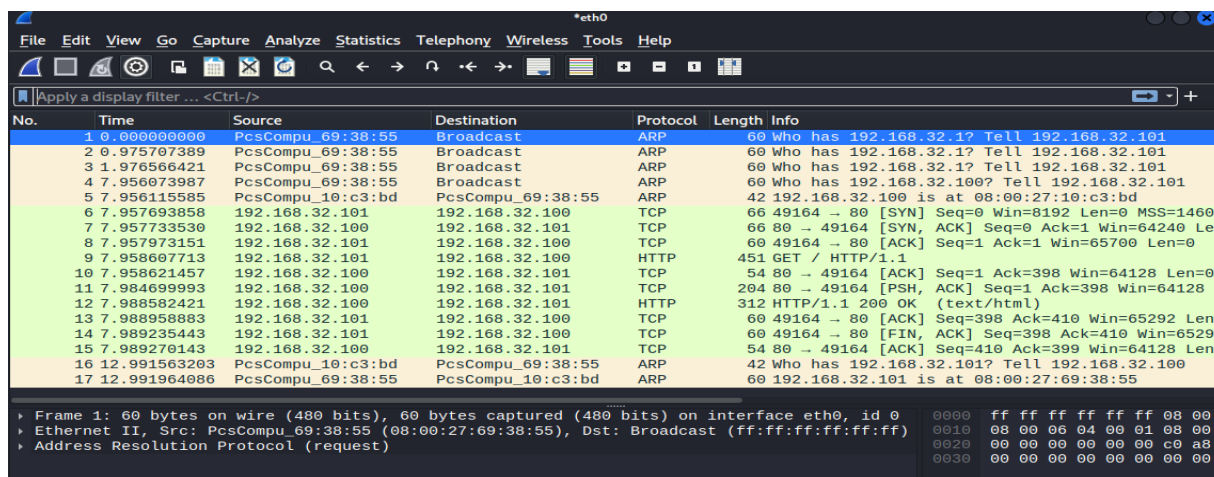
Possiamo quindi notare che il server DNS risolve correttamente epicode.internal.



## 6- Cattura e analisi pacchetti con Wireshark

Wireshark è uno strumento di analisi della rete ampiamente utilizzato per monitorare e analizzare il traffico di rete in tempo reale. Noi andremo ad utilizzarlo per intercettare la chiamata da windows 7 con l'indirizzo epicode.internal così da poterla analizzare nello specifico.

- **Traffico HTTP**



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu_69:38:55	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101
2	0.975707389	PcsCompu_69:38:55	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101
3	1.976566421	PcsCompu_69:38:55	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101
4	7.956073987	PcsCompu_69:38:55	Broadcast	ARP	60	Who has 192.168.32.100? Tell 192.168.32.101
5	7.956115585	PcsCompu_10:c3:bd	PcsCompu_69:38:55	ARP	42	192.168.32.100 is at 08:00:27:10:c3:bd
6	7.957693858	192.168.32.101	192.168.32.100	TCP	66	49164 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460
7	7.957733530	192.168.32.100	192.168.32.101	TCP	66	80 → 49164 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0
8	7.957973151	192.168.32.101	192.168.32.100	TCP	60	49164 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
9	7.958607713	192.168.32.101	192.168.32.100	HTTP	451	GET / HTTP/1.1
10	7.958621457	192.168.32.100	192.168.32.101	TCP	54	80 → 49164 [ACK] Seq=1 Ack=398 Win=64128 Len=0
11	7.984699993	192.168.32.100	192.168.32.101	TCP	204	80 → 49164 [PSH, ACK] Seq=1 Ack=398 Win=64128 Len=0
12	7.988582421	192.168.32.100	192.168.32.101	HTTP	312	HTTP/1.1 200 OK (text/html)
13	7.988958883	192.168.32.101	192.168.32.100	TCP	60	49164 → 80 [ACK] Seq=398 Ack=410 Win=65292 Len=0
14	7.989235443	192.168.32.101	192.168.32.100	TCP	60	49164 → 80 [FIN, ACK] Seq=398 Ack=410 Win=65292 Len=0
15	7.989270143	192.168.32.100	192.168.32.101	TCP	54	80 → 49164 [ACK] Seq=410 Ack=399 Win=64128 Len=0
16	12.991563203	PcsCompu_10:c3:bd	PcsCompu_69:38:55	ARP	42	Who has 192.168.32.101? Tell 192.168.32.100
17	12.991964086	PcsCompu_69:38:55	PcsCompu_10:c3:bd	ARP	60	192.168.32.101 is at 08:00:27:69:38:55

- Nel primo pacchetto possiamo notare uno scambio di pacchetti con protocollo ARP dove la macchina Windows (MAC Address: 08:00:69:38:55) chiede con un messaggio Broadcast a quale indirizzo MAC corrisponde 192.168.32.101 (IP di Kali);
- Nel quinto pacchetto Kali risponde confermando di avere l'indirizzo IP richiesto;

```
> Frame 5: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface eth0, id 0
> Ethernet II, Src: PcsCompu_10:c3:bd (08:00:27:10:c3:bd), Dst: PcsCompu_69:38:55 (08:00:27:69:38:55)
- Address Resolution Protocol (reply)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: reply (2)
  Sender MAC address: PcsCompu_10:c3:bd (08:00:27:10:c3:bd)
  Sender IP address: 192.168.32.100
  Target MAC address: PcsCompu_69:38:55 (08:00:27:69:38:55)
  Target IP address: 192.168.32.101
```

- Nel pacchetto 9 abbiamo la richiesta get da Windows 7 al server http;

9	7.958607713	192.168.32.101	192.168.32.100	HTTP	451 GET / HTTP/1.1
10	7.958621457	192.168.32.100	192.168.32.101	TCP	54 80 → 49164 [ACK
11	7.984699993	192.168.32.100	192.168.32.101	TCP	204 80 → 49164 [PSH
12	7.988582421	192.168.32.100	192.168.32.101	HTTP	312 HTTP/1.1 200 OK
13	7.988958883	192.168.32.101	192.168.32.100	TCP	60 49164 → 80 [ACK
14	7.989235443	192.168.32.101	192.168.32.100	TCP	60 49164 → 80 [FIN
15	7.989270143	192.168.32.100	192.168.32.101	TCP	54 80 → 49164 [ACK
16	12.991563203	PcsCompu_19:c3:bd	PcsCompu_69:c3:85	ARP	42 Who has 192.168.
17	12.991964086	PcsCompu_69:c3:85	PcsCompu_19:c3:bd	ARP	60 192.168.32.101

```
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: 397
  Hypertext Transfer Protocol
    GET / HTTP/1.1\r\n
    Accept: application/x-ms-application, image/jpeg, application/xaml+xml, image/gif, image/
    Accept-Language: it-IT\r\n
    User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2;
    Accept-Encoding: gzip, deflate\r\n
    Host: epicode.internal\r\n
    Connection: Keep-Alive\r\n
    \r\n
    [Full request URI: http://epicode.internal/]
    [HTTP request 1/1]
    [Response in frame: 12]
```

- Nel pacchetto 12 la risposta positiva che il server trasmette a Windows 7;

11	7.984099993	192.168.32.100	192.168.32.101	TCP	204 80 → 49104 [PSH, ACK] Seq=1
12	7.988582421	192.168.32.100	192.168.32.101	HTTP	312 HTTP/1.1 200 OK (text/html)
13	7.988958883	192.168.32.101	192.168.32.100	TCP	60 49164 → 80 [ACK] Seq=398 Ack=
14	7.989235443	192.168.32.101	192.168.32.100	TCP	60 49164 → 80 [FIN, ACK] Seq=398
15	7.989270143	192.168.32.100	192.168.32.101	TCP	54 80 → 49164 [ACK] Seq=410 Ack=
16	12.991563203	PcsCompu_10:c3:bd	PcsCompu_69:38:55	ARP	42 Who has 192.168.32.101? Tell
17	12.991964086	PcsCompu_69:38:55	PcsCompu_10:c3:bd	ARP	60 192.168.32.101 is at 08:00:2

```

[2 Reassembled TCP Segments (408 bytes): #11(150), #12(258)]
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
    Content-Type: text/html\r\n
    Connection: Close\r\n
    Date: Fri, 17 Nov 2023 21:13:47 GMT\r\n
    Server: INetSim HTTP Server\r\n
    Content-Length: 258\r\n
    \r\n
  [HTTP response 1/1]
  [Time since request: 0.029974708 seconds]
  [Request in frame: 9]
  [Request URI: http://epicode.internal/]
  File Data: 258 bytes
  line-based text data: text/html (10 lines)

```

- **Traffico HTTPS**

Time	Source	Destination	Protocol	Length	Info
1 0.000000000	PcsCompu_69:38:55	Broadcast	ARP	60	Who has 192.168.32.100? Tell 192.168.32.101
2 0.000022065	PcsCompu_10:c3:bd	PcsCompu_69:38:55	ARP	42	192.168.32.100 is at 08:00:27:10:c3:bd
3 0.005359734	192.168.32.101	192.168.32.100	TCP	66	49173 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460
4 0.005388639	192.168.32.100	192.168.32.101	TCP	66	443 → 49173 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0
5 0.005739358	192.168.32.101	192.168.32.100	TCP	60	49173 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0
6 0.010066696	192.168.32.101	192.168.32.100	TLSv1	215	Client Hello
7 0.010092167	192.168.32.100	192.168.32.101	TCP	54	443 → 49173 [ACK] Seq=1 Ack=162 Win=64128 Len=0
8 0.108391299	192.168.32.100	192.168.32.101	TLSv1	1373	Server Hello, Certificate, Server Key Exchange
9 0.116398779	192.168.32.101	192.168.32.100	TLSv1	188	Client Key Exchange, Change Cipher Spec, Encry
10 0.116428247	192.168.32.100	192.168.32.101	TCP	54	443 → 49173 [ACK] Seq=1320 Ack=296 Win=64128 Len=0
11 0.116960926	192.168.32.100	192.168.32.101	TLSv1	113	Change Cipher Spec, Encrypted Handshake Messag
12 0.124606626	PcsCompu_69:38:55	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101
13 0.323075060	192.168.32.101	192.168.32.100	TCP	60	49173 → 443 [ACK] Seq=296 Ack=1379 Win=64320 Len=0
14 0.962062972	PcsCompu_69:38:55	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101
15 1.963274839	PcsCompu_69:38:55	Broadcast	ARP	60	Who has 192.168.32.1? Tell 192.168.32.101
16 3.263739715	192.168.32.101	224.0.0.252	LLMNR	64	Standard query 0xa6a7 A wpa
17 3.368703439	192.168.32.101	224.0.0.252	LLMNR	64	Standard query 0xa6a7 A wpa

Nel secondo pacchetto possiamo notare gli indirizzi MAC di sorgente e destinazione ed i relativi indirizzi IP:

Sender MAC Address: 08:00:27:c3:bd - Sender IP 192.168.32.100 (Kali)

Sender MAC Address: 08:00:27:69:38:55 - Sender IP 192.168.32.101 (Windows)

Le informazioni trasmesse sono criptate infatti si può notare la presenza di pacchetti con protocollo TLS che garantisce la protezione di dati sensibili durante la comunicazione Client e server.

- **Differenze tra traffico HTTP e HTTPS**

HTTP (Hypertext Transfer Protocol) e HTTPS (Hypertext Transfer Protocol Secure) sono due protocolli di comunicazione utilizzati per trasferire dati su una rete, come ad esempio Internet. Le differenze tra HTTP e HTTPS riguardano la sicurezza e la crittografia dei dati.

Le principali differenze del traffico catturato che possiamo notare sono:

1. HTTP trasmette i dati in forma non crittografata, il che significa che le informazioni sono vulnerabili agli attacchi di intercettazione;
2. Nella richiesta HTTPS i dati sono criptati e quindi al momento della chiamata al server è utilizzato il protocollo TLS per una maggiore sicurezza e protezione dei dati;
3. L'HTTP invece non ha protocolli TLS;
4. Nel traffico HTTP al momento della chiamata è utilizzato il protocollo http che può, a differenza del TLS, mostrarci il contenuto della richiesta e della risposta.
5. La porta di default per HTTPS è la 443;
6. La porta di default per HTTP è la 80;

In conclusione, mentre entrambi i protocolli consentono la comunicazione tra il browser e il server, HTTPS offre un livello aggiuntivo di sicurezza attraverso la crittografia dei dati, contribuendo a proteggere la privacy e l'integrità delle informazioni scambiate tra l'utente e il sito web.

LEONARDO MARGHERI