

## Relazione PROGETTO FINALE – M1W4-D4

Simulazione, in un ambiente di laboratorio virtuale, di un'architettura Client – Server, in cui un client con indirizzo IP statico 192.168.32.101 (Windows) richiede tramite web browser una risorsa al hostname **epicode.internal**, che risponde all'indirizzo IP statico 192.168.32.100 (Kali Linux). Sniffare successivamente lo scambio di informazioni tra le due macchine virtuali in esame, tramite il tool professionale Wireshark, delineando i MAC Address e il tipo di informazioni scambiate tramite protocollo di comunicazione HTTPS (7° livello teorico ISO/OSI – Applicazione).

### Virtual Machine (VM) in esame (macchine virtualizzate sul software VirtualBox):

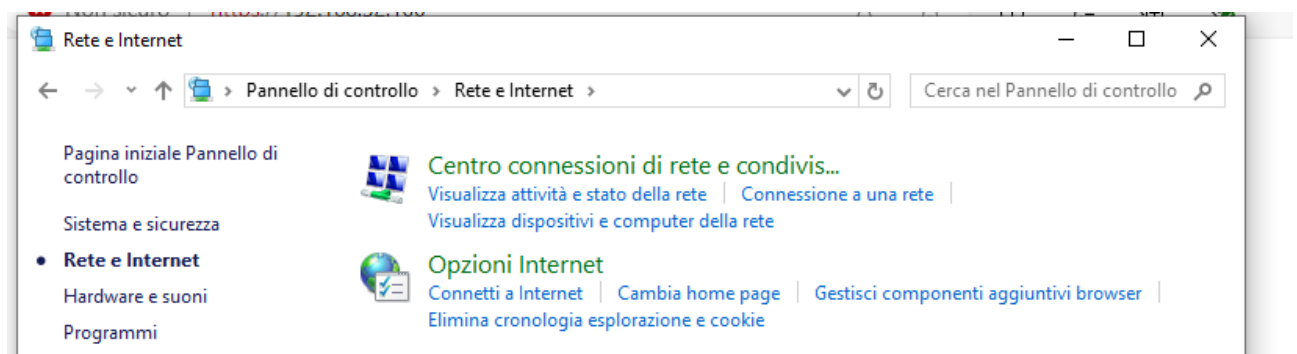
- **Windows10**  
IP statico: 192.168.32.101  
Netmask: 255.255.255.0  
Gateway: 198.168.32.1
- **Kali Linux**  
IP statico: 192.168.32.100  
Netmask: 255.255.255.0  
Gateway: 192.168.32.1

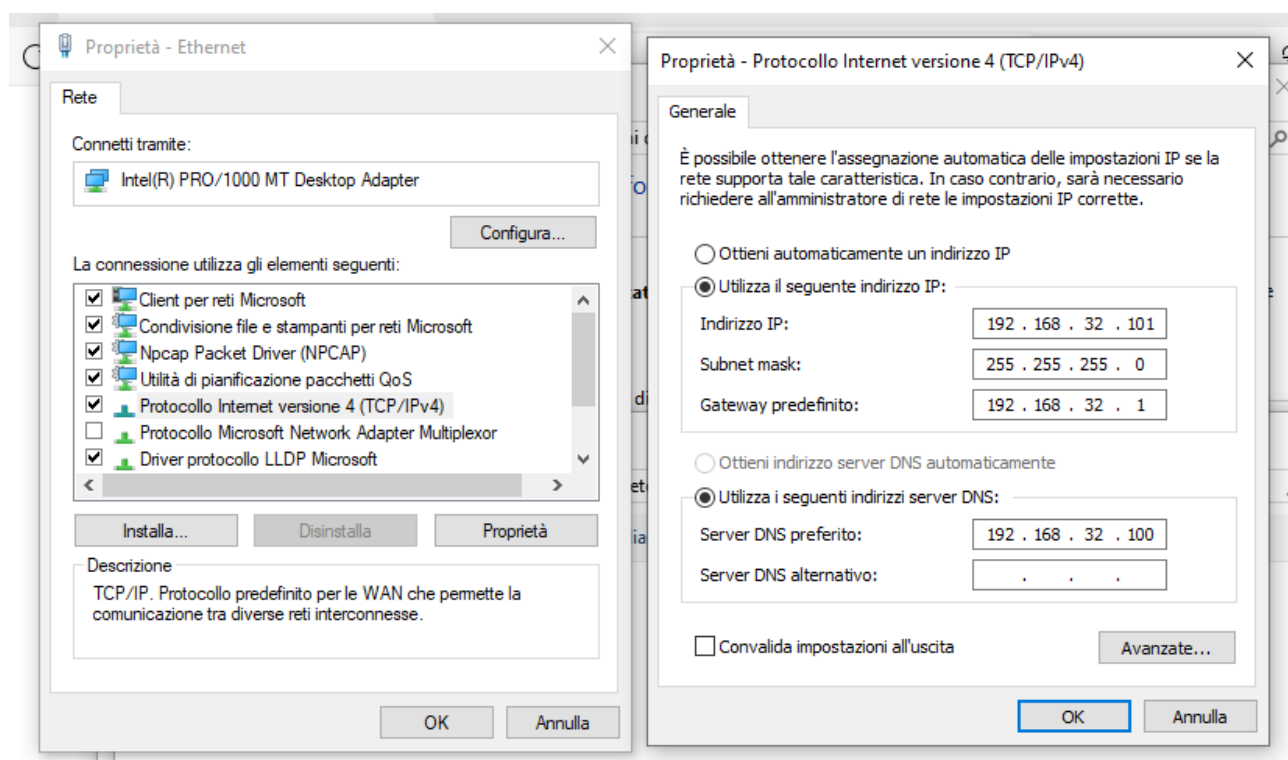
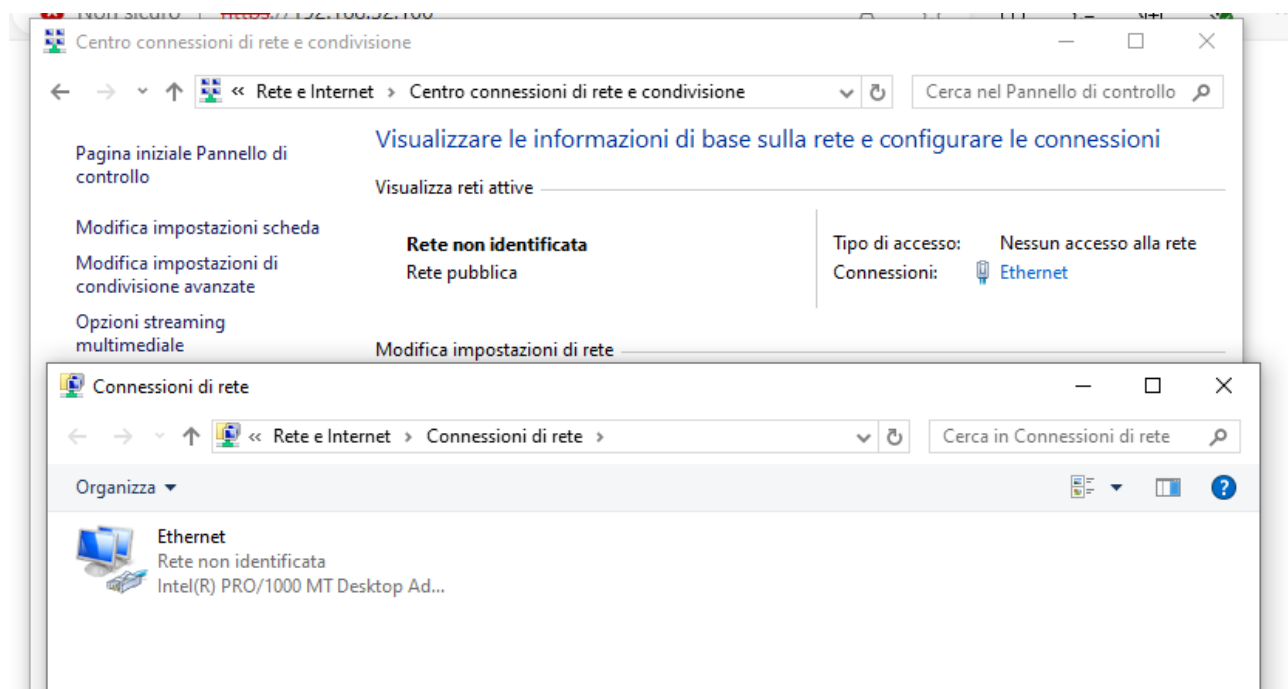
### Attività preliminari alla preparazione del laboratorio virtuale:

1. Creazione delle macchine virtuali su VirtualBox, assegnando tutte le impostazioni necessarie alla creazione delle macchine, RAM, CPU ecc.
2. Per ogni macchina creata, entrare nel menù impostazioni
3. Nella sezione “rete”, abilitare scheda di rete connessa a “internal”
4. Nella sezione “sistema”, ordine di avvio: ottico, disco fisso, rete -> defleggare Floppy

### Avvio Macchine Virtuali e impostazioni delle schede di rete:

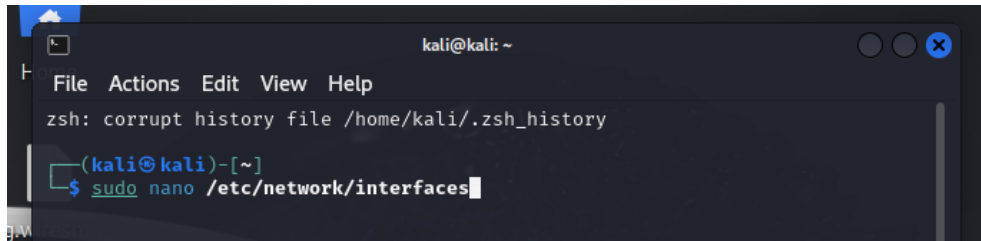
1. Dal menù di VirtualBox, avviare le VM Kali Linux e Windows10
2. Seguendo le procedure apprese, assegnare alla scheda di rete della VM Windows, un indirizzo di IP statico, una Netmask e un indirizzo Gateway



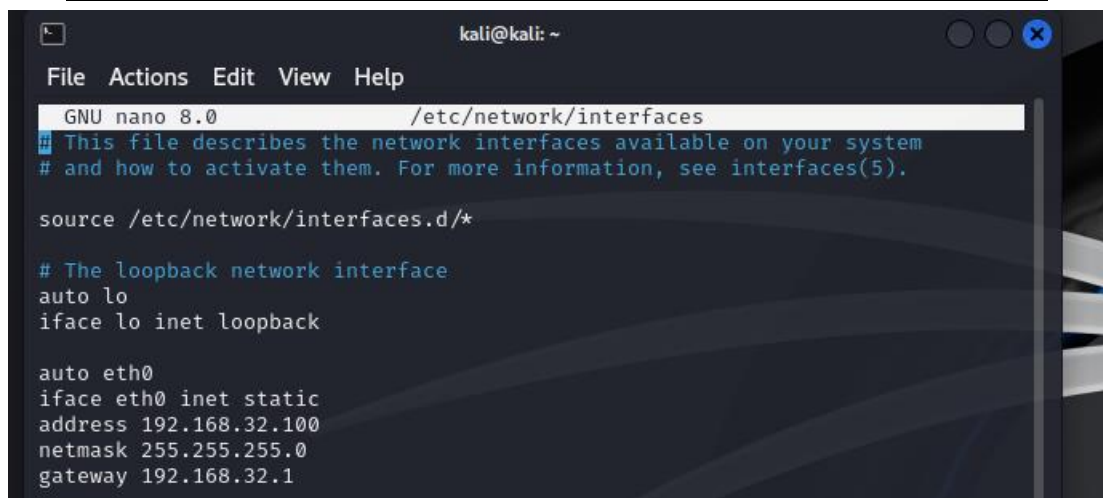


Nota bene: è stato impostato anche l'indirizzo del server DNS 192.168.32.100, ovvero quello che risponderà al dominio epicode.internal, sul server Kali 192.168.32.100

3. Seguendo le procedure apprese, assegnare alla scheda di rete della VM Kali, un indirizzo di IP statico, una Netmask e un indirizzo Gateway, digitando il comando su terminale: `sudo nano /etc/network/interfaces`; confermare ed eseguire un `sudo reboot` della macchina; vedi immagini sotto

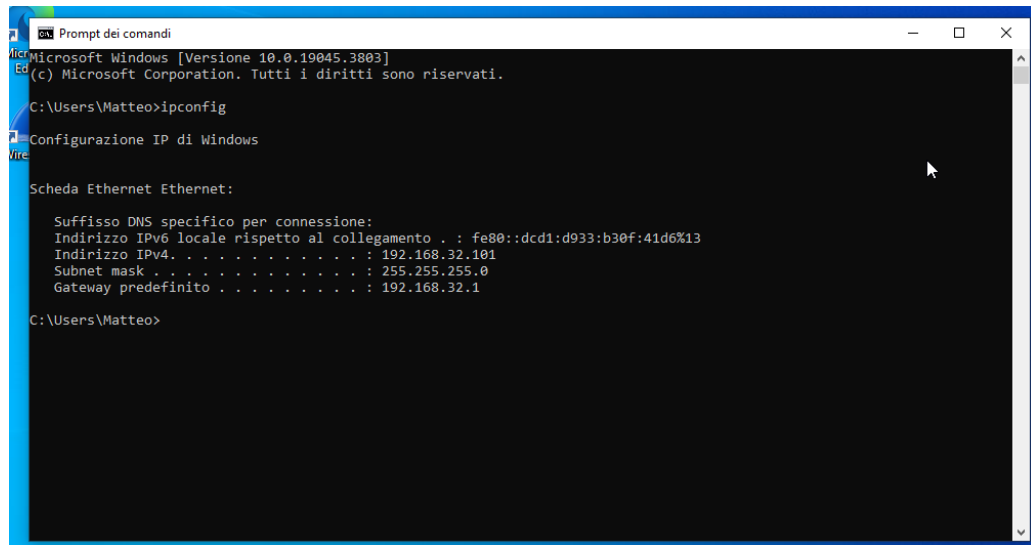


```
kali@kali: ~  
File Actions Edit View Help  
zsh: corrupt history file /home/kali/.zsh_history  
(kali@kali)-[~]  
$ sudo nano /etc/network/interfaces
```



```
kali@kali: ~  
File Actions Edit View Help  
GNU nano 8.0 /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  
netmask 255.255.255.0  
gateway 192.168.32.1
```

4. Sul cmd di Windows, eseguire un ipconfig e verificare che la macchina abbia appreso correttamente le impostazioni



```
Microsoft Windows [Versione 10.0.19045.3803]
(c) Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\Matteo>ipconfig

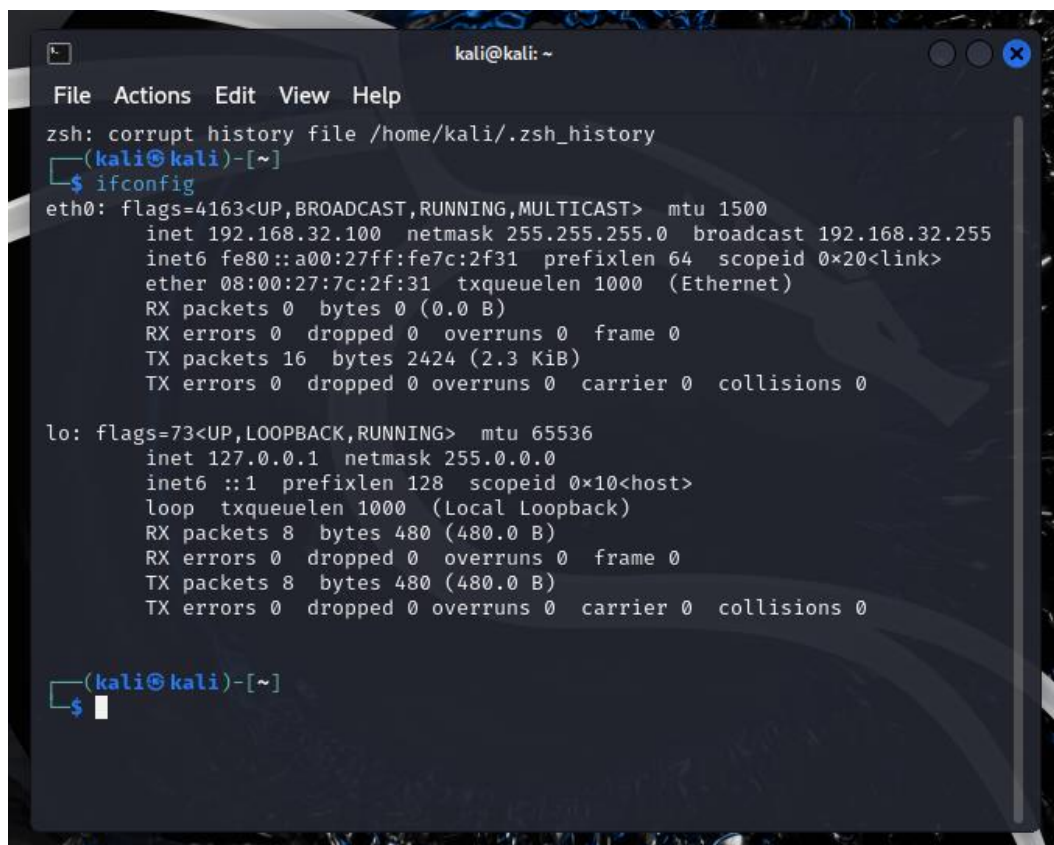
Configurazione IP di Windows

Scheda Ethernet Ethernet:

    Suffisso DNS specifico per connessione:
    Indirizzo IPv6 locale rispetto al collegamento . : fe80::dcd1:d933:b30f:41d6%13
    Indirizzo IPv4. . . . . : 192.168.32.101
    Subnet mask . . . . . : 255.255.255.0
    Gateway predefinito . . . . . : 192.168.32.1

C:\Users\Matteo>
```

5. Sul prompt dei comandi di Kali, eseguire un ifconfig e verificare che la macchina abbia appreso correttamente le impostazioni

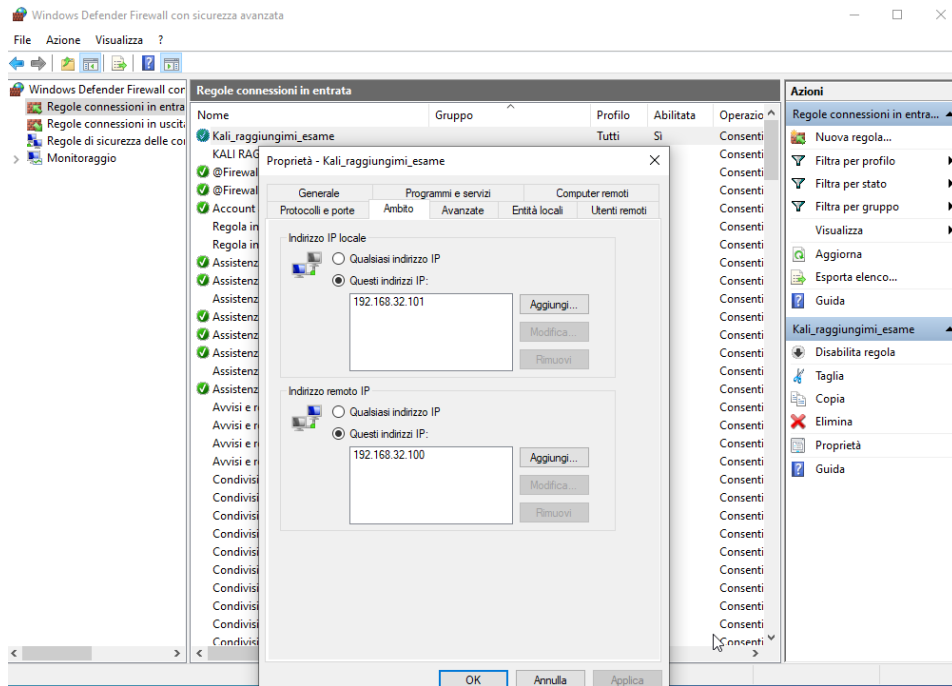
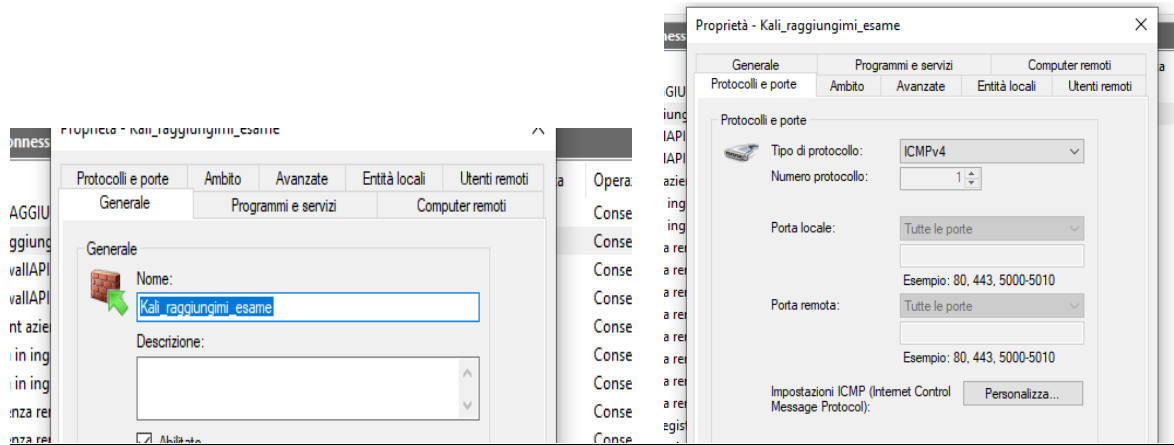


```
kali@kali: ~
File Actions Edit View Help
zsh: corrupt history file /home/kali/.zsh_history
(kali@kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.32.100 netmask 255.255.255.0 broadcast 192.168.32.255
    inet6 fe80::a00:27ff:fe7c:2f31 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:7c:2f:31 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 16 bytes 2424 (2.3 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 8 bytes 480 (480.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 8 bytes 480 (480.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

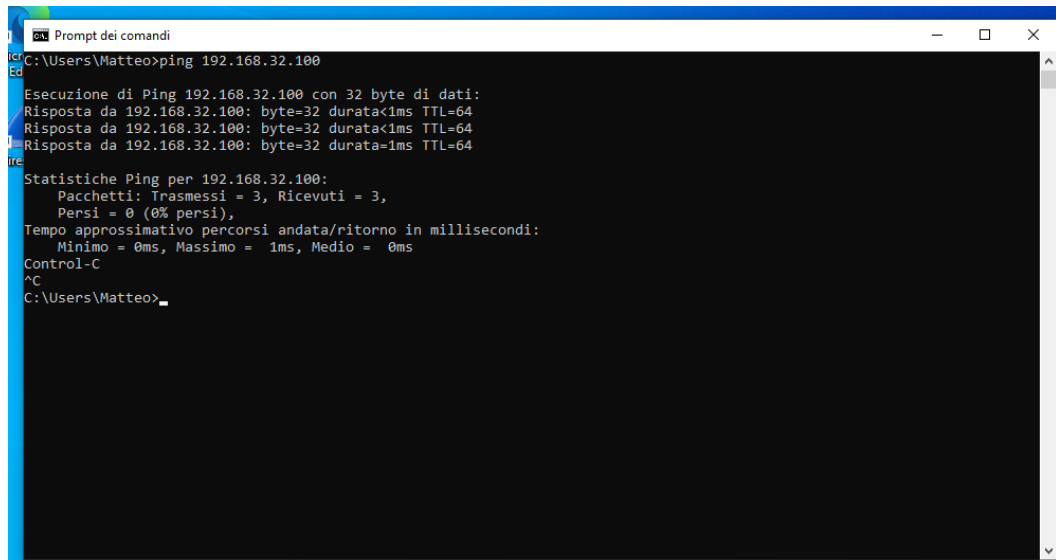
(kali@kali)-[~]
$
```

6. Per permettere alla macchina Kali Linux di raggiungere Windows10, ho creato una nuova Policy Firewall inbound, su windows firewall, impostando protocolli e porte "ICMPv4" e impostando in Ambito, l'indirizzo IP locale d'interesse e l'indirizzo IP della macchina remota, attribuire un nome alla policy creata -> abilitare la Policy; di seguito screenshot



7. Eseguire infine da ambedue le macchine, un ping che confermi la comunicazione reciproca delle stesse.
8. Le macchine comunicano, confermando l'invio dei pacchetti

Conferma trasmissione dei pacchetti da Windows10 a Kali Linux, tramite comando ping:

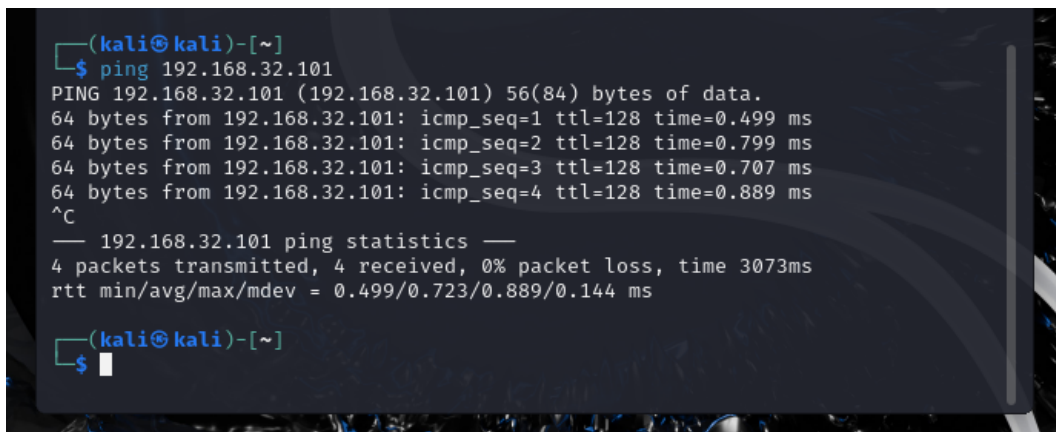


```
Prompt dei comandi
C:\Users\Matteo>ping 192.168.32.100

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

Statistiche Ping per 192.168.32.100:
    Pacchetti: Trasmessi = 3, Ricevuti = 3,
    Persi = 0 (0% persi),
    Tempo approssimativo percorsi andata/ritorno in millisecondi:
        Minimo = 0ms, Massimo = 1ms, Medio = 0ms
Control-C
^C
C:\Users\Matteo>
```

Conferma trasmissione dei pacchetti da Kali Linux a Windows10, tramite comando ping:

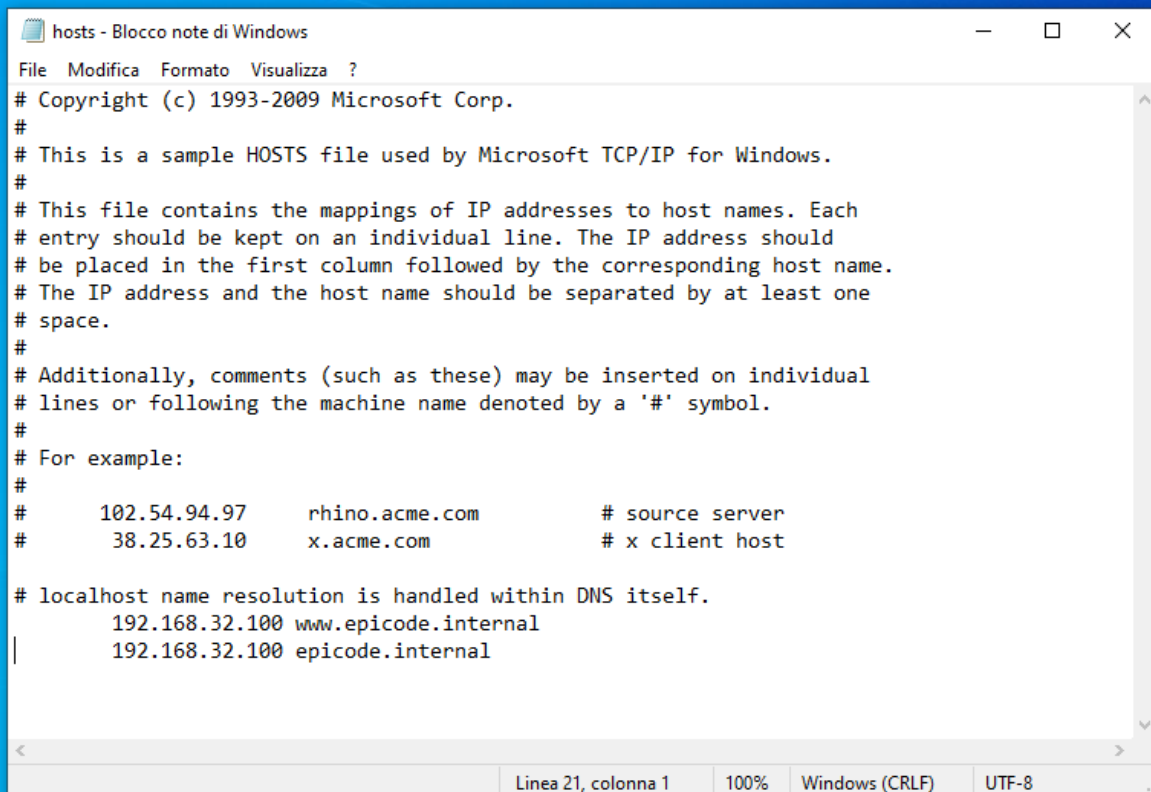


```
(kali㉿kali)-[~]
$ ping 192.168.32.101
PING 192.168.32.101 (192.168.32.101) 56(84) bytes of data.
64 bytes from 192.168.32.101: icmp_seq=1 ttl=128 time=0.499 ms
64 bytes from 192.168.32.101: icmp_seq=2 ttl=128 time=0.799 ms
64 bytes from 192.168.32.101: icmp_seq=3 ttl=128 time=0.707 ms
64 bytes from 192.168.32.101: icmp_seq=4 ttl=128 time=0.889 ms
^C
— 192.168.32.101 ping statistics —
4 packets transmitted, 4 received, 0% packet loss, time 3073ms
rtt min/avg/max/mdev = 0.499/0.723/0.889/0.144 ms
(kali㉿kali)-[~]
$
```

**Configurazione e modifica file hosts di Windows10, per il rilascio dell'hostname epicode.internal su indirizzo IP 192.168.32.100 (kali);** operazione importante, in quanto, se non eseguita, durante la prova di raggiungimento del server con hostname epicode.internal dal browser windows10, la pagina browser non ci mostrerà nulla, risultando irraggiungibile.

Seguendo il seguente percorso per il raggiungimento del file: disco c – windows – system32 – drivers – etc – hosts (file)

Di seguito modifiche apportate



```
hosts - Blocco note di Windows
File  Modifica  Formato  Visualizza  ?
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#       102.54.94.97       rhino.acme.com          # source server
#       38.25.63.10       x.acme.com              # x client host

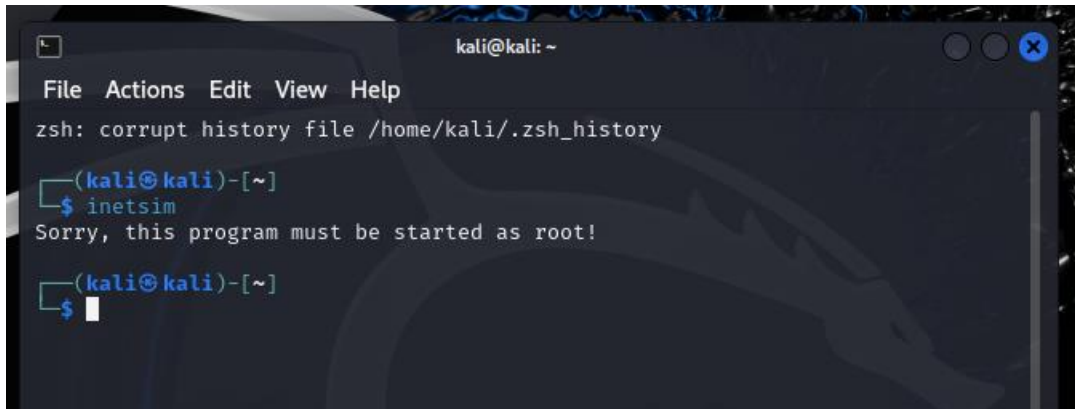
# localhost name resolution is handled within DNS itself.
#       192.168.32.100   www.epicode.internal
#       192.168.32.100   epicode.internal
```

**NOTA BENE:** il file hosts, dovrà essere eseguito come amministratore, per poi essere modificato come da immagine qui sopra; le modifiche al file hosts posso essere apportate e salvate solo se eseguite da un amministratore.




## Configurazione e creazione, con il tool “Inetsim” offerto da Kali Linux, di un servizio internet simulato, HTTPS, nel laboratorio virtuale

1. Pre attivare Inetsim come segue da riga di comando: inetsim  
P.S. in questo caso io lo avevo già pre attivato sulla macchina



```
kali@kali: ~  
File Actions Edit View Help  
zsh: corrupt history file /home/kali/.zsh_history  
(kali@kali)-[~]  
$ inetsim  
Sorry, this program must be started as root!  
(kali@kali)-[~]  
$
```

2. Configurazione di Inetsim come richiesto, per la creazione di un servizio internet simulato, basato sullo scambio di informazioni su protocollo HTTPS, tramite riga di comando: `sudo nano /etc/inetsim/inetsim.conf`



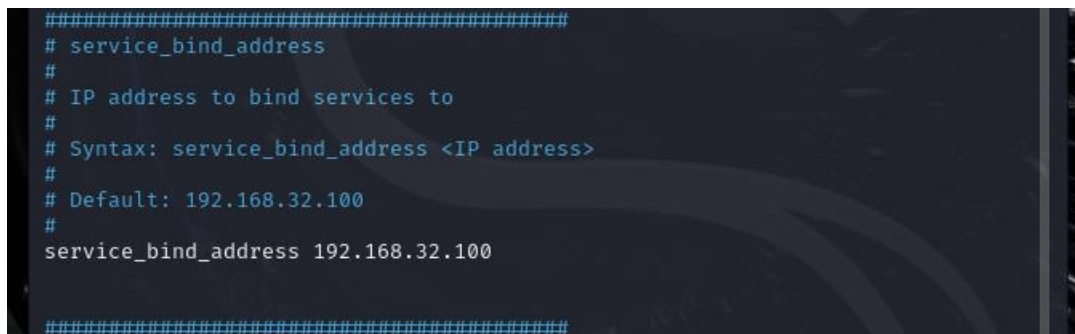
```
(kali@kali)-[~]  
$ sudo nano /etc/inetsim/inetsim.conf  
[sudo] password for kali:
```

Indichiamo senza il simbolo #, i servizi che vogliamo attivare: HTTPS e DNS



```
# 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
```

Impostiamo l'IP Address del service bind, sul quale sarà in ascolto la comunicazione



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



Nella sezione DNS, configuriamo il DNS, come segue: inserendo nella sezione DNS default IP, l'indirizzo IP 192.168.32.100 (Kali), idem nella sezione dns\_static

```
#####  
# Service DNS  
#####  
  
#####  
# dns_bind_port  
#  
# Port number to bind DNS service to  
#  
# Syntax: dns_bind_port <port number>  
#  
# Default: 53  
#  
#dns_bind_port 53  
  
#####  
# dns_default_ip  
#  
# Default IP address to return with DNS replies  
#  
# Syntax: dns_default_ip <IP address>  
# Default: 192.168.32.100  
#  
dns_default_ip 192.168.32.100
```

```
#####  
# 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  
  
#####
```

Salviamo le configurazioni (ctrl+o – enter – ctrl+x) e ritorniamo alla schermata principale della riga di comando, attiviamo quindi il servizio Inetsim appena configurato, come segue:

```
9.4  
(kali@kali)-[~]  
$ sudo inetsim
```

Una volta inserita la password “kali”, il servizio si attiverà come segue:

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

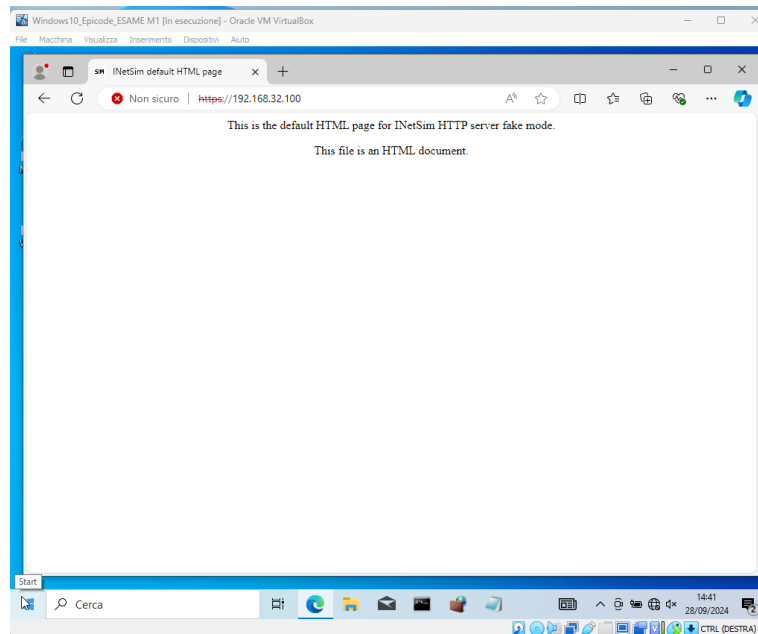
(kali㉿kali)-[~]
$ sudo inetsim
[sudo] password for kali:
INetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
Using log directory: /var/log/inetsim/
Using data directory: /var/lib/inetsim/
Using report directory: /var/log/inetsim/report/
Using configuration file: /etc/inetsim/inetsim.conf
Parsing configuration file.
Configuration file parsed successfully.
== INetSim main process started (PID 23872) ==
Session ID: 23872
Listening on: 192.168.32.100
Real Date/Time: 2024-09-28 08:34:40
Fake Date/Time: 2024-09-28 08:34:40 (Delta: 0 seconds)
Forking services...
* dns_53_tcp_udp - started (PID 23874)
  deprecated method; prefer start_server() at /usr/share/perl5/INetSim/DNS.pm l
  ine 69.
  Attempt to start Net::DNS::Nameserver in a subprocess at /usr/share/perl5/INe
  tSim/DNS.pm line 69.
* https_443_tcp - started (PID 23875)
  done.
Simulation running.
```

Possiamo notare che il servizio HTTPS è in ascolto sulla porta 443, sull'indirizzo IP 192.168.32.100 (che corrisponde con epicode.internal – Kali)

Torniamo ora sulla Macchina Virtuale Windows10

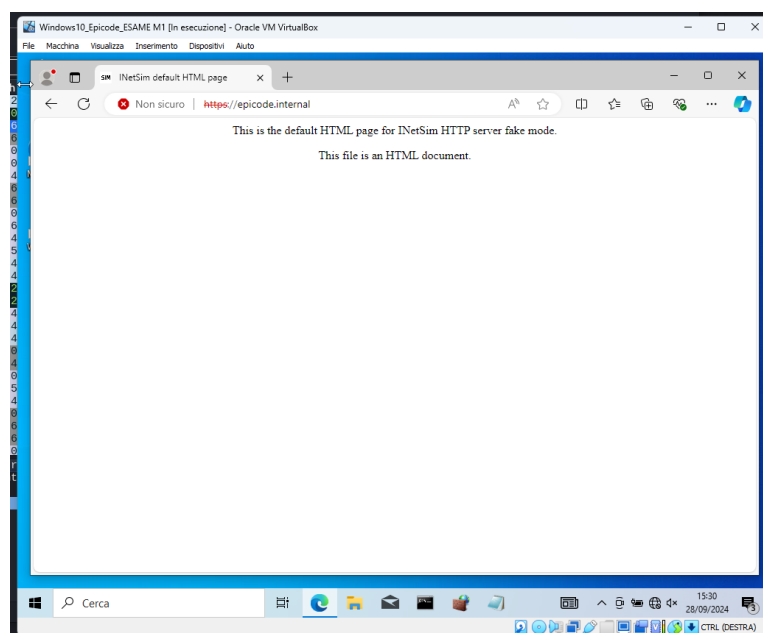
**Apertura del browser Edge, su Windows10, raggiungere l'indirizzo IP 192.168.32.100:443 (kali):**

scrivere nel browser, nella barra del URL, l'indirizzo IP 192.168.32.100:443 (in ascolto sulla porta 443); in caso positivo comparirà la schermata sotto riportata:



Oppure scrivere nel browser, nella barra del URL, l'hostname “epicode.internal” (corrispondente all'indirizzo IP 192.168.32.100:443 - in ascolto sulla porta 443); in caso positivo comparirà la schermata sotto riportata:

Scrittura URL: <https://epicode.internal>



## Apertura del tool professionale Wireshark, per sniffare la comunicazione e trasmissione delle informazioni su protocollo **HTTPS**

Attiviamo Wireshark, quando sul browser di windows10, procediamo al raggiungimento dell'indirizzo IP 192.168.32.100 (kali) / hostname epicode.internal, possiamo benissimo notare lo scambio di informazioni su protocollo TCP, con l'apertura del three-way handshake (SYN, SYN-ACK, ACK)

Time	Source	Destination	Protocol	Length	Info
16.2.022342860	192.168.32.101	192.168.32.100	DNS	72	Standard query 0x44ef A www.bing.com
17.2.022352604	192.168.32.100	192.168.32.101	ICMP	100	Destination unreachable (Port unreachable)
18.4.144650829	192.168.32.101	192.168.32.100	TCP	60	60566 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
19.4.144687954	192.168.32.100	192.168.32.101	TCP	66	443 → 60566 [SYN, ACK] Seq=0 Ack=1 Win=32120 Len=0 MSS=1460 SACK_PERM WS=128
20.4.144944307	192.168.32.101	192.168.32.100	TCP	60	60566 → 443 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
21.4.145453418	192.168.32.101	192.168.32.100	TLSv1.3	2090	Client Hello
22.4.145462369	192.168.32.100	192.168.32.101	TCP	54	443 → 60566 [ACK] Seq=1 Ack=2037 Win=31872 Len=0
23.4.155001607	192.168.32.100	192.168.32.100	TCP	66	60567 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
24.4.155032939	192.168.32.100	192.168.32.101	TCP	66	443 → 60567 [SYN, ACK] Seq=0 Ack=1 Win=32120 Len=0 MSS=1460 SACK_PERM WS=128
25.4.155244882	192.168.32.101	192.168.32.100	TCP	60	60567 → 443 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
26.4.155688892	192.168.32.101	192.168.32.100	TLSv1.3	2026	Client Hello
27.4.155698206	192.168.32.100	192.168.32.101	TCP	54	443 → 60567 [ACK] Seq=1 Ack=1973 Win=31872 Len=0
28.4.158819401	192.168.32.100	192.168.32.101	TLSv1.3	1475	Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data, Applica...
29.4.160159585	192.168.32.101	192.168.32.100	DNS	94	Standard query 0xeac5 A nav-edge.smartscreen.microsoft.com
30.4.160159836	192.168.32.101	192.168.32.100	DNS	94	Standard query 0x4d8d HTTPS nav-edge.smartscreen.microsoft.com
31.4.160182941	192.168.32.100	192.168.32.101	ICMP	122	Destination unreachable (Port unreachable)
32.4.160192953	192.168.32.100	192.168.32.101	ICMP	122	Destination unreachable (Port unreachable)
33.4.160537594	192.168.32.101	192.168.32.100	TLSv1.3	84	Change Cipher Spec, Application Data
34.4.160687413	192.168.32.101	192.168.32.100	DNS	94	Standard query 0xd6d8 HTTPS nav-edge.smartscreen.microsoft.com
35.4.160845294	192.168.32.101	192.168.32.100	DNS	94	Standard query 0x609b A nav-edge.smartscreen.microsoft.com
36.4.160845418	192.168.32.101	192.168.32.100	TCP	60	60566 → 443 [FIN, ACK] Seq=2067 Ack=1422 Win=2100736 Len=0
37.4.164587377	192.168.32.100	192.168.32.101	TCP	54	443 → 60566 [FIN, ACK] Seq=1422 Ack=2068 Win=31872 Len=0
38.4.164754871	192.168.32.101	192.168.32.100	TCP	60	60566 → 443 [ACK] Seq=2068 Ack=1423 Win=2100736 Len=0

## Evidenziamo i Mac Address nelle varie fasi di SYN, SYN-ACK, ACK

Nella fase di SYN, riportiamo i seguenti dati, nei quali possiamo notare i Mac Address nella sezione Ethernet II

32.101	192.168.32.100	TCP	66	60566 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
32.100	192.168.32.101	TCP	66	443 → 60566 [SYN, ACK] Seq=0 Ack=1 Win=32120 Len=0 MSS=1460 SACK_PERM
32.101	192.168.32.100	TCP	60	60566 → 443 [ACK] Seq=1 Ack=1 Win=2102272 Len=0

Wireshark - Packet 18 - eth0	
▶ Frame 18: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface eth0, id 0	
▶ Ethernet II, Src: PCSSystemtec_8e:b0:eb (08:00:27:8e:b0:eb), Dst: PCSSystemtec_92:31:24 (08:00:27:92:31:24)	
▶ Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100	
▶ Transmission Control Protocol, Src Port: 60566, Dst Port: 443, Seq: 0, Len: 0	

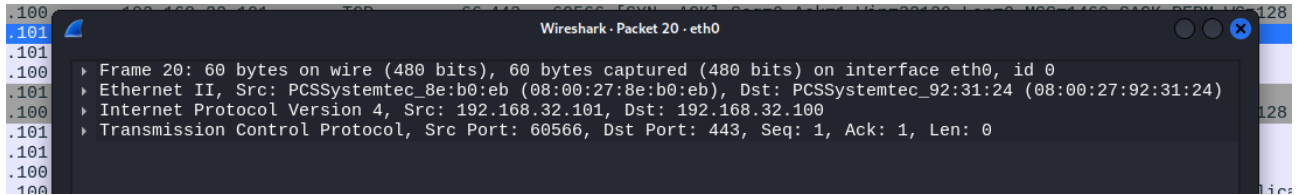
Nella fase di SYN-ACK, riportiamo i seguenti dati, nei quali possiamo notare i Mac Address nella sezione Ethernet II, facendo particolare attenzione all'inversione degli stessi da sorgente a destinatario e viceversa; il Mac che prima era sorgente, ora è il destinatario e viceversa

32.101	192.168.32.100	TCP	60	60566 → 443 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
--------	----------------	-----	----	---

Wireshark - Packet 19 - eth0	
▶ Frame 19: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface eth0, id 0	
▶ Ethernet II, Src: PCSSystemtec_92:31:24 (08:00:27:92:31:24), Dst: PCSSystemtec_8e:b0:eb (08:00:27:8e:b0:eb)	
▶ Internet Protocol Version 4, Src: 192.168.32.100, Dst: 192.168.32.101	
▶ Transmission Control Protocol, Src Port: 443, Dst Port: 60566, Seq: 0, Ack: 1, Len: 0	

Nella fase di ACK, riportiamo i seguenti dati, nei quali possiamo notare i Mac Address nella sezione Ethernet II, facendo particolare attenzione alla nuova inversione degli stessi da sorgente a destinatario e viceversa; il Mac che prima era sorgente, ora è il destinatario e viceversa.



**Nota bene:** il medesimo procedimento di inversione da sorgente a destinatario e viceversa, avviene anche per gli indirizzi IP delle macchine (windows10 e Kali)

## Evidenziamo ora le informazioni trasmesse su protocollo HTTPS nelle varie fasi di SYN, SYN-ACK, ACK e FIN-ACK

SYN:

```
Wireshark - Packet 14 - eth0
> Frame 14: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface eth0, id 0
> Ethernet II, Src: PCSSystemtec_8e:b0:eb (08:00:27:8e:b0:eb), Dst: PCSSystemtec_92:31:24 (08:00:27:92:31:24)
> Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100
> Transmission Control Protocol, Src Port: 60579, Dst Port: 443, Seq: 0, Len: 0
  Source Port: 60579
  Destination Port: 443
  [Stream index: 0]
  [Conversation completeness: Complete, WITH_DATA (31)]
    ..0. .... = RST: Absent
    ...1 .... = FIN: Present
    .... 1... = Data: Present
    .... .1.. = ACK: Present
    .... ..1 = SYN-ACK: Present
    .... ...1 = SYN: Present
  [Completeness Flags: -FDASS]
  [TCP Segment Len: 0]
  Sequence Number: 0 (relative sequence number)
  Sequence Number (raw): 3418264270
  [Next Sequence Number: 1 (relative sequence number)]
  Acknowledgment Number: 0
  Acknowledgment number (raw): 0
  1000 .... = Header Length: 32 bytes (8)
  [Flags: 0x002 (SYN)]
    000. .... = Reserved: Not set
    ...0 .... = Accurate ECN: Not set
    .... 0... = Congestion Window Reduced: Not set
    .... .0.. = ECN-Echo: Not set
    .... ..0 = Urgent: Not set
    .... ...0 = Acknowledgment: Not set
    .... ....0 = Push: Not set
    .... .....0 = Reset: Not set
    > .... ..1 = Syn: Set
    .... ....0 = Fin: Not set
    [TCP Flags: .....S.]
  Window: 64240
  [Calculated window size: 64240]
  Checksum: 0x6519 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  > Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP), SACK permitted
  [Timestamps]
    [Time since first frame in this TCP stream: 0.000000000 seconds]
    [Time since previous frame in this TCP stream: 0.000000000 seconds]
0000 08 00 27 92 31 24 08 00 27 8e b0 eb 08 00 45 00 ..'1$.. '....E.
0010 00 34 49 01 40 00 80 06 ef a8 c0 a8 20 65 c0 a8 ..4I @... ..e.

No. 14 - Time: 1.530150128 - Source: 192.168.32.101 - Destination: 192.168.32.100 - Protocol: TCP - Length: 66 - Info: 60579 -> 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
```

SYN-ACK:

```
> Frame 15: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface eth0, id 0
> Ethernet II, Src: PCSSystemtec_8e:b0:eb (08:00:27:8e:b0:eb), Dst: PCSSystemtec_8e:b0:eb (08:00:27:8e:b0:eb)
> Internet Protocol Version 4, Src: 192.168.32.100, Dst: 192.168.32.101
> Transmission Control Protocol, Src Port: 443, Dst Port: 60579, Seq: 0, Ack: 1, Len: 0
  Source Port: 443
  Destination Port: 60579
  [Stream index: 0]
  [Conversation completeness: Complete, WITH_DATA (31)]
    ..0. .... = RST: Absent
    ...1 .... = FIN: Present
    .... 1... = Data: Present
    .... .1.. = ACK: Present
    .... ..1 = SYN-ACK: Present
    .... ...1 = SYN: Present
  [Completeness Flags: -FDASS]
  [TCP Segment Len: 0]
  Sequence Number: 0 (relative sequence number)
  Sequence Number (raw): 1242831228
  [Next Sequence Number: 1 (relative sequence number)]
  Acknowledgment Number: 1 (relative ack number)
  Acknowledgment number (raw): 3418264271
  1000 .... = Header Length: 32 bytes (8)
  [Flags: 0x012 (SYN, ACK)]
    000. .... = Reserved: Not set
    ...0 .... = Accurate ECN: Not set
    .... 0... = Congestion Window Reduced: Not set
    .... .0.. = ECN-Echo: Not set
    .... ..0 = Urgent: Not set
    .... ...1 = Acknowledgment: Set
    .... ....0 = Push: Not set
    .... .....0 = Reset: Not set
    > .... ..1 = Syn: Set
    .... ....0 = Fin: Not set
    [TCP Flags: .....A..S.]
  Window: 32120
  [Calculated window size: 32120]
  Checksum: 0xc240 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  > Options: (12 bytes), Maximum segment size, No-Operation (NOP), No-Operation (NOP), SACK permitted, No-Operation (NOP), Window scale
  [Timestamps]
    [Time since first frame in this TCP stream: 0.000055083 seconds]
    [Time since previous frame in this TCP stream: 0.000055083 seconds]
  [SEQ/ACK analysis]
0000 08 00 27 8e b0 eb 08 00 27 92 31 24 08 00 45 00 ..'1$.. '....E.
No. 15 - Time: 1.530205271 - Source: 192.168.32.100 - Destination: 192.168.32.101 - Protocol: TCP - Length: 66 - Info: 443 -> 60579 [SYN, ACK] Seq=0 Ack=1 Win=32120 Len=0 MSS=1460 SACK_PERM WS=128
Show packet bytes
```

## ACK:

```
Wireshark - Packet 16 - eth0
▼ Frame 16: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface eth0, id 0
▼ Ethernet II, Src: PCSSystemtec_8e:b0:eb (08:00:27:8e:b0:eb), Dst: PCSSystemtec_92:31:24 (08:00:27:92:31:24)
▼ Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100
▼ Transmission Control Protocol, Src Port: 60579, Dst Port: 443, Seq: 1, Ack: 1, Len: 0
  Source Port: 60579
  Destination Port: 443
  [Stream index: 0]
  ▼ [Conversation completeness: Complete, WITH_DATA (31)]
    ..0. .... = RST: Absent
    ...1 .... = FIN: Present
    .... 1... = Data: Present
    .... .1.. = ACK: Present
    .... ..1. = SYN-ACK: Present
    .... ...1 = SYN: Present
    [Completeness Flags: -FDASS]
    [TCP Segment Len: 0]
    Sequence Number: 1 (relative sequence number)
    Sequence Number (raw): 3418264271
    [Next Sequence Number: 1 (relative sequence number)]
    Acknowledgment Number: 1 (relative ack number)
    Acknowledgment number (raw): 1242831229
    0101 .... = Header Length: 20 bytes (5)
  ▼ Flags: 0x010 (ACK)
    000. .... = Reserved: Not set
    ...0 .... = Accurate ECN: Not set
    .... 0... = Congestion Window Reduced: Not set
    .... .0.. = ECN-Echo: Not set
    .... ..0. = Urgent: Not set
    .... ...1 = Acknowledgment: Set
    .... .... = Push: Not set
    .... ..0. = Reset: Not set
    .... ...0. = Syn: Not set
    .... .... = Fin: Not set
    [TCP Flags: .....A....]
    Window: 8212
    [Calculated window size: 2102272]
    [Window size scaling factor: 256]
    Checksum: 0xd28 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  ▼ [Timestamps]
    [Time since first frame in this TCP stream: 0.000427991 seconds]
    [Time since previous frame in this TCP stream: 0.000372908 seconds]
0000 08 00 27 92 31 24 08 00 27 8e b0 eb 08 00 45 00 ..'1$'....E.
0010 00 28 49 02 40 00 00 06 ef b3 c0 a8 20 65 c0 a8 -(I.@... ..e..
0020 20 64 ec a3 01 bb cb be 92 cf 4a 14 19 7d 50 10 d.....J..}P.

No.: 16 - Time: 1.530578119 - Source: 192.168.32.101 - Destination: 192.168.32.100 - Protocol: TCP - Length: 60 - Info: 60579 → 443 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
▼ Show packet bytes
```

## FIN-ACK:

```
Wireshark - Packet 26 - eth0
▼ Frame 26: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface eth0, id 0
▼ Ethernet II, Src: PCSSystemtec_8e:b0:eb (08:00:27:8e:b0:eb), Dst: PCSSystemtec_92:31:24 (08:00:27:92:31:24)
▼ Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100
▼ Transmission Control Protocol, Src Port: 60579, Dst Port: 443, Seq: 2092, Ack: 1422, Len: 0
  Source Port: 60579
  Destination Port: 443
  [Stream index: 0]
  ▼ [Conversation completeness: Complete, WITH_DATA (31)]
    ..0. .... = RST: Absent
    ...1 .... = FIN: Present
    .... 1... = Data: Present
    .... .1.. = ACK: Present
    .... ..1. = SYN-ACK: Present
    .... ...1 = SYN: Present
    [Completeness Flags: -FDASS]
    [TCP Segment Len: 0]
    Sequence Number: 2092 (relative sequence number)
    Sequence Number (raw): 3418266362
    [Next Sequence Number: 2093 (relative sequence number)]
    Acknowledgment Number: 1422 (relative ack number)
    Acknowledgment number (raw): 1242832650
    0101 .... = Header Length: 20 bytes (5)
  ▼ Flags: 0x011 (FIN, ACK)
    000. .... = Reserved: Not set
    ...0 .... = Accurate ECN: Not set
    .... 0... = Congestion Window Reduced: Not set
    .... .0.. = ECN-Echo: Not set
    .... ..0. = Urgent: Not set
    .... ...1 = Acknowledgment: Set
    .... .... = Push: Not set
    .... ..0. = Reset: Not set
    .... ...0. = Syn: Not set
    .... ...1 = Fin: Set
    [TCP Flags: .....A..F]
    Window: 8206
    [Calculated window size: 2100736]
    [Window size scaling factor: 256]
    Checksum: 0xf75 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  ▼ [Timestamps]
    [Time since first frame in this TCP stream: 0.017776510 seconds]
    [Time since previous frame in this TCP stream: 0.00043421 seconds]
0000 08 00 27 92 31 24 08 00 27 8e b0 eb 08 00 45 00 ..'1$'....E.

No.: 26 - Time: 1.547926638 - Source: 192.168.32.101 - Destination: 192.168.32.100 - Protocol: TCP - Length: 60 - Info: 60579 → 443 [FIN, ACK] Seq=2092 Ack=1422 Win=2100736 Len=0
▼ Show packet bytes
```

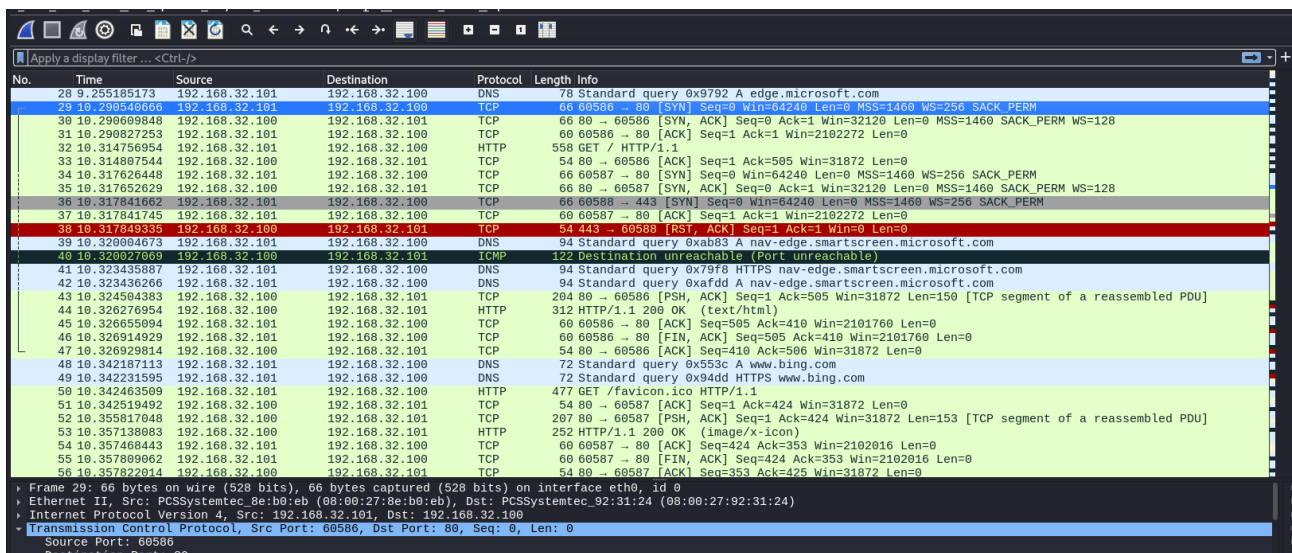


## Apertura del tool professionale Wireshark, per sniffare la comunicazione e trasmissione delle informazioni su protocollo **HTTP**

**Premessa:** per verificare e osservare lo scambio di informazioni su protocollo http, senza apportare particolari modifiche nel menù di configurazione di InetSim, entrando nel menù di configurazione da riga di comando, utilizzando il medesimo comando, disattiviamo il servizio HTTPS in precedenza attivato (quindi apponiamo il simbolo # prima della dicitura) e cancelliamo invece il simbolo # presente davanti alla dicitura del servizio http (la scritta diventa bianca), vedi immagine di seguito:

```
# dns, dhcp, dhcpv6, pop3, tcp, tcp, tcp, 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
```

Il service dns, rimane attivo e con impostazioni invariate, lasciare le impostazioni di default per il servizio http (così come è stato per HTTPS)



No.	Time	Source	Destination	Protocol	Length	Info
28	9.255185173	192.168.32.101	192.168.32.100	DNS	78	Standard query 0x9792 A edge.microsoft.com
29	10.290800000	192.168.32.101	192.168.32.100	TCP	60	60586 → 80 [SYN] Seq=0 Win=2102272 Len=0 MSS=1460 WS=256 SACK_PERM
30	10.290809848	192.168.32.100	192.168.32.101	TCP	66	80 → 60586 [SYN, ACK] Seq=0 Ack=1 Win=32120 Len=0 MSS=1460 SACK_PERM WS=128
31	10.290827253	192.168.32.101	192.168.32.100	TCP	60	60586 → 80 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
32	10.314756954	192.168.32.101	192.168.32.100	HTTP	558	GET / HTTP/1.1
33	10.314807544	192.168.32.100	192.168.32.101	TCP	54	80 → 60586 [ACK] Seq=1 Ack=505 Win=31872 Len=0
34	10.317626448	192.168.32.101	192.168.32.100	TCP	66	60587 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
35	10.317652629	192.168.32.100	192.168.32.101	TCP	66	80 → 60587 [SYN, ACK] Seq=0 Ack=1 Win=32120 Len=0 MSS=1460 SACK_PERM WS=128
36	10.317841662	192.168.32.101	192.168.32.100	TCP	66	60588 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
37	10.317841745	192.168.32.101	192.168.32.100	TCP	66	60587 → 80 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
38	10.317849335	192.168.32.100	192.168.32.101	TCP	54	443 → 60588 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
39	10.320804673	192.168.32.101	192.168.32.100	DNS	94	Standard query 0xab83 A nav-edge.smartscreen.microsoft.com
40	10.320827069	192.168.32.100	192.168.32.101	ICMP	122	Destination unreachable (Port unreachable)
41	10.323435887	192.168.32.101	192.168.32.100	DNS	94	Standard query 0x79f8 HTTPS nav-edge.smartscreen.microsoft.com
42	10.323436266	192.168.32.101	192.168.32.100	DNS	94	Standard query 0xafdd A nav-edge.smartscreen.microsoft.com
43	10.324504383	192.168.32.100	192.168.32.101	TCP	204	80 → 60586 [PSH, ACK] Seq=1 Ack=505 Win=31872 Len=150 [TCP segment of a reassembled PDU]
44	10.326276954	192.168.32.100	192.168.32.101	HTTP	312	HTTP/1.1 200 OK (text/html)
45	10.326655094	192.168.32.101	192.168.32.100	TCP	60	60586 → 80 [ACK] Seq=505 Ack=410 Win=2101760 Len=0
46	10.326914929	192.168.32.101	192.168.32.100	TCP	60	60586 → 80 [FIN, ACK] Seq=505 Ack=410 Win=2101760 Len=0
47	10.326929814	192.168.32.100	192.168.32.101	TCP	54	80 → 60586 [ACK] Seq=410 Ack=506 Win=31872 Len=0
48	10.342187113	192.168.32.101	192.168.32.100	DNS	72	Standard query 0x553c A www.bing.com
49	10.342231595	192.168.32.101	192.168.32.100	DNS	72	Standard query 0x94dd HTTPS www.bing.com
50	10.342463509	192.168.32.101	192.168.32.100	HTTP	477	GET /favicon.ico HTTP/1.1
51	10.342519492	192.168.32.100	192.168.32.101	TCP	54	80 → 60587 [ACK] Seq=1 Ack=424 Win=31872 Len=0
52	10.355817048	192.168.32.100	192.168.32.101	TCP	207	80 → 60587 [PSH, ACK] Seq=1 Ack=424 Win=31872 Len=153 [TCP segment of a reassembled PDU]
53	10.357138083	192.168.32.100	192.168.32.101	HTTP	252	HTTP/1.1 200 OK (image/x-icon)
54	10.357468443	192.168.32.101	192.168.32.100	TCP	60	60587 → 80 [ACK] Seq=424 Ack=353 Win=2102816 Len=0
55	10.357899062	192.168.32.101	192.168.32.100	TCP	60	60587 → 80 [FIN, ACK] Seq=424 Ack=353 Win=2102816 Len=0
56	10.357922014	192.168.32.100	192.168.32.101	TCP	54	80 → 60587 [ACK] Seq=353 Ack=425 Win=31872 Len=0

Frame 29: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface eth0, id 0  
Ethernet II, Src: PCSSystemtec\_8e:b0:eb (08:00:27:8e:b0:eb), Dst: PCSSystemtec\_92:31:24 (08:00:27:92:31:24)  
Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100  
Transmission Control Protocol, Src Port: 60586, Dst Port: 80, Seq: 0, Len: 0  
Source Port: 60586  
Destination Port: 80

Come avvenuto nel procedimento in HTTPS, andiamo ora ad analizzare le informazioni scambiate nelle varie fasi, SYN, SYN-ACK, ACK, su protocollo http:

SYN:

```
Wireshark - Packet 29 - eth0
> Frame 29: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface eth0, id 0
> Ethernet II, Src: PCSSystemtec_8e:b0:eb (08:00:27:8e:b0:eb), Dst: PCSSystemtec_92:31:24 (08:00:27:92:31:24)
> Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100
> Transmission Control Protocol, Src Port: 60586, Dst Port: 80, Seq: 0, Len: 0
  Source Port: 60586
  Destination Port: 80
  [Stream index: 0]
```

SYN-ACK:

```
> Frame 30: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface eth0, id 0
> Ethernet II, Src: PCSSystemtec_92:31:24 (08:00:27:92:31:24), Dst: PCSSystemtec_8e:b0:eb (08:00:27:8e:b0:eb)
> Internet Protocol Version 4, Src: 192.168.32.100, Dst: 192.168.32.101
> Transmission Control Protocol, Src Port: 80, Dst Port: 60586, Seq: 0, Ack: 1, Len: 0
  Source Port: 80
  Destination Port: 60586
  [Stream index: 0]
  [Conversation completeness: Complete, WITH DATA (31)]
```

ACK:

```
> Frame 31: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface eth0, id 0
> Ethernet II, Src: PCSSystemtec_8e:b0:eb (08:00:27:8e:b0:eb), Dst: PCSSystemtec_92:31:24 (08:00:27:92:31:24)
> Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100
> Transmission Control Protocol, Src Port: 60586, Dst Port: 80, Seq: 1, Ack: 1, Len: 0
  Source Port: 60586
  Destination Port: 80
  [Stream index: 0]
  [Conversation completeness: Complete, WITH DATA (31)]
```

Nota bene: come avvenuto durante lo scambio di informazioni su protocollo HTTPS, anche in questo caso, durante le fasi di SYN, SYN-ACK, ACK, si verificano l'inversione dei Mac address delle macchine e i corrispettivi indirizzi IP, da sorgente a destinatario e viceversa

HTTPS

# HTTP

- Porte di comunicazione: HTTPS porta 443, http porta 80
- Tipologia di informazioni ricavate differenti
- Nel HTTPS non mi è possibile osservare il tipo di protocollo di comunicazione e il tipo di informazioni trasmesse
- Nel HTTP posso osservare le informazioni trasmesse: vedi es. la richiesta “GET”
- Nel HTTPS non riesco a ricavare la richiesta appunto HTTPS, ma è soltanto visibile il trasporto delle informazioni su protocollo TCP (livello trasporto ISO/OSI)

Grazie per l'attenzione